Visão computacional rápida utilizando rede neural implementada em processamento paralelo

Roberto Alves de Oliveira

USP/IFSC/SBI

8-2-001660

Tese apresentada ao Instituto de Física de São Carlos, da Universidade de São Paulo, para obtenção do título de Doutor em Ciências: Física Aplicada – opção: Física Computacional

Orientador: Prof. Dr. Valentim Obac Roda

São Carlos – 2002

IFSC-USP SERVIÇO DE BIBLIOTECA INFORMAÇÃO

Oliveira, Roberto Alves de

"Visão computacional rápida utilizando rede neural implementada em processamento paralelo."
Roberto Alves de Oliveira – São Carlos, 2002

Tese (Doutorado) – Área de Física Aplicada-opção Física Computacional do Instituto de Física de São Carlos da Universidade de São Paulo
2002 - Páginas: 254

Orientador: Prof. Dr. Valentim Obac Roda
Visão computacional; rede neural; processamento paralelo.
I. Título

IFSC-USP SERVIÇO DE BIBLIOTECA E INFORMAÇÃO

UNIVERSIDADE
DE SÃO PAULO
Instituto de Física de São Carlos

Av. Trabalhador São-carlense, 400
Caixa Postal 369
13560-970 - São Carlos - SP - Brasil

MEMBROS DA COMISSÃO JULGADORA DA TESE DE DOUTORADO DE ROBERTO ALVES DE OLIVEIRA, APRESENTADA AO INSTITUTO DE FÍSICA DE SÃO CARLOS, UNIVERSIDADE DE SÃO PAULO, EM 20-9-2002.

COMISSÃO JULGADORA:

Prof. Dr. Valentin Obac Roda (Orientador e Presidente) – EESC-USP

Prof. Dr. Sergio Takeo Kofuji – EP-USP

Prof. Dr. José Hiroki Saito – UFSCar

Prof. Dr. André C Ponce de Leon F de Carvalho – ICMC-USP

Prof. Dr. Guilherme Matos Sipahi – IFSC-USP

AGRADECIMENTOS

Agradeço a Deus, por me ter dado força e saúde para conseguir atingir este objetivo.

Agradeço a minha esposa e filhas, pela compreensão e estímulo para o término deste trabalho.

À Wladerez, que sempre atendeu-me com tanta paciência e boa vontade.

Agradeço ao Prof. Dr. Onofre, pela colaboração e oportunidade de uso do Laboratório de Computação de Alto Desempenho.

Ao meu orientador, Prof. Dr. Valentin Obac Roda, pela imensa paciência e pelas oportunidades oferecidas.

SUMÁRIO

LISTA DE FIGURAS i

LISTA DE TABELAS iii

RESUMO iv

ABSTRACT v

1 - INTRODUÇÃO 1
1 – INTRODUÇÃO 1
1.1 - *Objetivo* 1
1.2 – *Constituição do Sistema* 2
1.3 – *Organização da Dissertação* 3

2 - COMPUTAÇÃO PARALELA 5
2.1 – *Introdução* 5
2.2 – *Redes de Comunicação de Alta Velocidade* 5
2.2.1 – *Atomic* 5
2.2.2 –*Myrinet* 6
2.2.3 – *ATM – Asynchronou Transfer Mode* 7
2.2.4 – *SCI – Scalable Coherent Interface* 7

2.2.5 – *HIPPI – High Performance Parallel Interface* 7
2.2.6 – *Fiber Channel* 8
2.2.7 – *Características Principais das Redes Apresentadas* 8

3 – PROCESSAMENTO PARALELO 12
3.1 - *Introdução* 12
3.2 - *Tipos de Paralelismo* 13
3.3 – *Concorrência ou Paralelismo?* 18
3.4 - *Granularidade ou Nível de Paralelismo* 18
3.5 – *Speedup e Eficiencia* 19
3.6 – Arquiteturas para Processamento Paralelo 20
3.6.1 – *Taxonomia de Flynn* 22
3.6.1.1 – *SISD* 23
3.6.1.2 – *SIMD* 24
3.6.1.3 – *MISD* 24
3.6.1.4 - *MIMD* 24
3.6.2 – Taxonomia Estrutural 26
3.6.3 – Taxonomia de Duncam 28
3.6.3.1 – Arquiteturas Síncronas 30
3.6.3.1.1 – Processadores Vetoriais Pipeline 30
3.6.3.1.2 – Arquiteturas SIMD 30
3.6.3.1.3 – Arquitetura Sistólicas 31
3.6.3.2 – Arquiteturas Assíncronas 32
3.6.3.2.1 – Arquiteturas MIND 32
3.6.3.2.1.1 – Arquitetura MIND com Memória Compartilhada 33
3.6.3.2.1.2 – Arquitetura MIND com Memória Distribuída 38
3.6.3.2.2 – Paradígma MIND 41
3.6.4 – *A arquitetura do SPP3* 42
3.6.4.1 – *Organização e Implementação do SPP3* 43

4 – PROGRAMAÇÃO PARALELA 45
4.1 – Algoritmo Paralelo 46
4.2 – Ativação e término de processos paralelos 46
4.3 – Comunicação e Sincronismo 48
4.3.1 – Comunicação e Sincronismo em Memória Compartilhada 50
4.3.2 – Comunicação e Sincronismo em Memória Distribuída 51
4.4 – Ferramentas para desenvolvimento de aplicações paralelas 56
4.4.1 – Ferramentas de Programação 56
4.4.2 – Ferramentas de Auxílio à Paralelização 57
4.4.3 – Ferramentas de Depuração 57
4.4.4 – Ferramentas de Análise de Desempenho 58

5 – VISÃO COMPUTACIONAL 59
5.1 – Visão Computacional e Robótica 59
5.2 – Processamento Digital de Imagens 64
5.2.1 – Operações Básicas 65
5.2.2 – Operações Estruturais 66
5.3 – Reconhecimento de Padrões 67
5.3.1 – Características 70
5.3.1.1 – Características radiométricas de uma imagem 70
5.3.1.2 – Características de textura 71
5.3.2 – Classificação 77
5.3.3 – Métodos para reconhecimento de padrões 81
5.4 – Redes Neurais 82
5.4.1 – Introdução 82
5.4.2 – Neurônio 83
5.4.2.1 – Neurônio humano 83
5.4.2.2 – Neurônio artificial 84
5.4.3 – Classificação das redes neurais 89
5.4.3.1 – Operações das unidade de processamento 89
5.4.3.2 – Topologia das redes 90
5.4.3.3 – Relação entre entrada e saída de dados 91
5.4.3.4 – Algoritmo de aprendizagem 91
5.4.3.5 – Taxonomia as redes neurais 92
5.4.3.6 – Escolha de modelo de rede neural 93
5.4.3.7 – Perceptron Múltiplas Camadas 93
5.4.3.8 – Aprendizado 95
5.4.4 – Reconhecimento de padrões usando redes neurais 96
5.4.4.1 – Normalização de padrões 96
5.4.4.2 – Invariância pelo treinamento 97
5.4.4.3 – Utilização de atributos invariantes 98
5.4.4.4 – Invariância pela estrutura de rede 99

6 – SISTEMA DE VISÃO COMPUTACIONAL RÁPIDA UTILIZANDO REDE NEURAL IMPLEMENTADA EM PROCESSAMENTO PARALELO 100
6.1 – Introdução 100
6.2 – Implementação do sistema de visão 100
6.2.1 – Aquisição da Imagem 101
6.2.1.1 – Iluminação 101
6.2.1.1.1 – Iluminação em sistemas para visão de máquina 103
6.2.1.2 – Focalização da imagem 106
6.2.1.2.1 – Lente 106
6.2.1.2.2 – Distância focal (F) e abertura da lente (f) 107
6.2.1.2.3 – Ângulo de Visão ou de Cobertura 110

6.2.1.2.4 – Profundidade de Campo 110
6.2.1.2.4.1 – Controle da profundidade de campo 114
6.2.1.2.5 - Supernitidez 117
6.2.1.3 - Detecção da Imagem 117
6.2.1.4 - Câmera 117
6.2.1.5 - Placa de Captura de Vídeo 120
6.2.2 – Pré-Processamento 122
6.2.2.1 - Filtragem 122
6.2.2.2 - Falsos Contornos 124
6.2.2.3 - Limiarização (Thresholding) 125
6.2.3 – Extração de Características 126
6.2.4 – Reconhecimento de padrões 130
6.2.5 – Aplicação 134

7 – DESENVOLVIMENTO EXPERIMENTAL 135
7.1 – Sistema de Iluminação 135
7.2 – Suporte para fixação da câmera 135
7.3 – Aquisição da imagem 135
7.4 – Processamento paralelo 137
7.4.1 – Pré-processamento, Treinamento e Teste 137
7.4.2 – Utilização do sistema para reconhecimento e Aplicação 139
7.4.3 – Programação 141
7.4.4 – Análise dos Resultados 144

8 – CONCLUSÕES 157
8.1 – Considerações finais 157
8.2 – Contribuições 157
8.3 – Sugestões para trabalhos futuros 158

BIBLIOGRAFIA 161

APÊNDICE A – IMAGENS DE OBJETOS UTILIZADOS 170

APÊNDICE B – TABELAS DE CARACTERÍSTICAS E RESULTADOS EXPERIMENTAIS 176

APÊNDICE C – DETECÇÃO DA IMAGEM (Complemento) 203

APÊNDICE D - ALGORITMOS E CÓDIGOS FONTE 230

LISTA DE FIGURAS

Figura 1 - Utilização básica para o Sistema de Visão Implementado 2
Figura 3.1 - Paralelismo lógico 13
Figura 3.2 – Paralelismo físico 14
Figura 3.3 – Paralelismo misto 14
Figura 3.4 – Estágio pipeline 15
Figura 3.5 – Execução das tarefas em relação ao tempo 15
Figura 3.6 – Classificação de Processamento Paralelo 16
Figura 3.7 – Execução das operações de instruções com pipeline 16
Figura 3.8 – Execuções executadas em super-pipeline 17
Figura 3.9 – Instruções executadas com super-escalonamento 17
Figura 3.1 – Classificação de Flyn 21
Figura 3.2 – Diagrama da arquitetura SISD 22
Figura 3.3 – Diagrama da arquitetura SIMD 23
Figura 3.4 – Diagrama da arquitetura MISD 24
Figura 3.5 – Diagrama da arquitetura MIMD 25
Figura 3.6 – Diagrama da Classificação estrutural 26
Figura 3.7 – Arquiteturas matriciais 27
Figura 3.8 – Classificação estrutural para MIMD 28
Figura 3.9 – Classificação de Duncam 29
Figura 3.10 – Arquitetura MIMD com memória compartilhada 33
Figura 3.11 – Arquitetura MIMD com memória distribuída 38
Figura 3.12 – Arquitetura MIMD com memória compartilhada e com memória cache 34
Figura 3.13 – Redes de interconexão para memória compartilhada 37
Figura 3.14 – Redes de interconexão para memória distribuída 40
Figura 3.15 – Arquitetura do SPP3 42
Figura 3.16 – Arquitetura do SPP3 44
Figura 4.1 – Uso de carotinas 47
Figura 4.2 – Exemplo de utilização fork / join 48
Figura 4.3 – Exemplo de utilização Cobegin/Coend 48
Figura 4.4 – Exemplo de utilização doall 49
Figura 4.5 – Funcionamento de uma comunicação síncrona 52
Figura 4.6 – Funcionamento de uma comunicação assíncrona 52
Figura 4.7 – Funcionamento da comunicação não bloqueante 53
Figura 4.8 – Mecanismo de comunicação Ponto-a-Ponto 54
Figura 4.9 – Mecanismo de comunicação rendezvouz 55
Figura 4.10 – Mecanismo de comunicação RPC 55
Figura 5.1 – Estrutura básica de um robô 60
Figura 5.2 – Sistema básico de visão 62
Figura 5.3 – Sistema de Reconhecimento de padrões 68
Figura 5.4 – Escolha das características de entrada 70
Figura 5.5 – Código cadeia. (a) orientação para 4-conectado. (b) orientação para 8-conectado 75
Figura 5.6 – Exemplo da aplicação do código cadeia 75
Figura 5.7 – Figura aproximada por polígonos 76
Figura 5.8 – Classificação pelo método do vizinho mais próximo. 77
Figura 5.9 – Exemplo de classificador linear. (a) duas funções discriminantes para Separar as duas classes. (b) resultado da classificação através das Funções discriminantes. 78
Figura 5.10 – Classificador Bayesiano. 80
Figura 5.11 – Neurônio 83
Figura 5.12 – Modelo de Neurônio artificial de McCulloc & Pitts 85

IFSC-USP SERVIÇO DE BIBLIOTECA INFORMAÇÃO

Figura 5.13 – Funções de ativação ou de transferência 86
Figura 5.14 – Modelo básico do neurônio 87
Figura 5.15 – Função Hard Limit 87
Figura 5.16 – Problema do ou-exclusivo 89
Figura 5.17 – Estrutura de redes neurais. (a) direta, (b) recorrente, (c) treliça e (d) pirâmide. 90
Figura 5.18 – Perceptron múltiplas camadas 94
Figura 5.19 – Função sigmoidal 95
Figura 5.20 – Função de energia de uma rede neural 95
Figura 5.21 – RNA utilizando normalização de padrões par reconhecimento de padrões 97
Figura 5.22 – RNA com utilização de invariância pelo treinamento para reconhecimento de padrões 97
Figura 5.23 – RNA utilizando a abordagem e atributos invariantes para recolhecimento de padrões 98
Figura 6.1 - Sistema de aquisição/iluminação de imagens utilizado nesta tese 107
Figura 6.2 - Marcações típicas de abertura da lente no anel de abertura da lente 108
Figura 6.3 - O tamanho da abertura do diafragma íris para diferentes regulagens da lente 108
Figura 6.4 - Com a abertura ajustada em f/4, o diâmetro da abertura do diafragma íris de uma objetiva de 50 mm mede 12,5 109
Figura 6.5 - Escala de distância em pés (ft) e metros (m). O losango indica a distância em que a câmera está focalizada. 110
Figura 6.6 - Profundidade de campo 113
Figura 6.7 - Profundidade de campo de uma objetiva focalizada a distância fixa. 113
Figura 6.8 – Profundidade de campo em função da variação da distância em que a câmera é focalizada. 114
Figura 6.9 - Profundidade de campo quando varia a distância focal 115
Figura 6.10 - Escala típica de profundidade de campo 116
Figura 6.11 - Leitura da profundidade de campo com abertura f/11 quando a objetiva da câmera está a 5 m. 116
Figura 6.12 - Ajuste da objetiva numa distância hiperfocal (profundidade de campo máxima) com abertura de f/16 117
Figura 6.13 – Princípio básico de um dispositivo de carga acoplada. 119
Figura 6.14 - Um dispositivo de imagem a carga acoplada 120
Figura 6.15 - Os tipos primários de sensores de imagem, definidos pela combina - ção de fotosensores 120
Figura 7.1 - Utilização de paralelismo através da estratégia processo "FARM". 139
Figura 7.2 - Paralelismo utilizando a divisão do módulo de extração de característica. 141
Figura 7.3 – Primitivas básicas do MPI 142
Figura 7.4 - Rede Neural *Perceptron* Múltiplas Camadas (MLP 145

LISTA DE TABELAS

Tabela 5.1 - Taxonomia de redes neurais 91
Tabela 6.1 - Técnicas de Iluminação 104
Tabela 6.2 - Profundidade de campo de uma objetiva focalizada a distância fixa 111
Tabela 6.3 - Profundidade de campo em função da variação da distância focalizada 112
Tabela 6.4 - Profundidade de campo em função da variação da distância focal 112
Tabela 7.1 - Resultado do desempenho encontrado para o teste de treinamento e validação dos conjuntos (a) A, (b) B e (c) C, utilizando a função de aprendizagem **Backpropagation Padrão** com coeficiente de aprendizagem 0,4 após o cálculo da média de dez (10) testes. 146
Tabela 7.2 - Resultado do desempenho encontrado para o teste de treinamento e validação dos conjuntos (a) A, (b) B e (c) C, utilizando a função de aprendizagem **Backpropagation Momentum** com coeficiente de aprendizagem 0,4 e fator de momento 0,5 após o cálculo da média de dez (10) treinamentos e validação. 148
Tabela 7.3 - Resultado do desempenho encontrado para o teste de treinamento e validação dos conjuntos (a) A, (b) B e (c) C, utilizando a função de aprendizagem **Backpropagation Momentum,** coeficiente de aprendizagem 0,2 e fator de momento 0,5 após o cálculo da média de dez (10) treinamentos e validação. 150
Tabela 7.4 -. Desempenho encontrado durante os testes de treinamento, validação e testes realizados utilizando os conjuntos de testes (a) A, (b) B e (c) C com função de aprendizagem **Backpropagation Padrão** com coeficiente de aprendizagem 0,4 após o cálculo da média de dez (10) testes. 152
Tabela 7.5 - Desempenho encontrado durante os testes de treinamento, validação e testes realizados utilizando os conjuntos de testes (a) A, (b) B e (c) C com função de aprendizagem **Backpropagation Momentum,** coeficiente de aprendizagem 0,4 e fator de momento 0,5 após o cálculo da média de dez (10) testes. 154
Tabela 7.6 - Desempenho encontrado durante os testes de treinamento, validação e testes realizados utilizando os conjuntos de testes (a) A, (b) B e (c) C com função de aprendizagem **Backpropagation Momentum,** coeficiente de aprendizagem 0,2 e fator de momento 0,5 após o cálculo da média de dez (10) testes. 155
Tabela 7.7 - Resultado do speedup para os testes de treinamento variando-se o número de amostras e número de nós de processamento. 157

RESUMO

O presente trabalho apresenta o desenvolvimento de um sistema de Visão Artificial inteligente visando uma maior velocidade de processamento, um menor custo e aumento na produtividade industrial. Para o desenvolvimento do sistema foi utilizado o computador paralelo SPP3 desenvolvido no LCAD (Laboratório de Computação de Alto Desempenho) do ICMSC (Instituto de Ciências e Matemática de São Carlos) que utiliza uma arquitetura paralela MIMD com memória distribuída e a uma rede de comunicação de alta velocidade do tipo Myrinet [TRINDADE, 1994].

Este trabalho de tese teve como proposta desenvolver um sistema de visão em tempo real. Para atender os objetivos propostos citados, realizou-se de forma inédita a utilização de métodos estatísticos na extração do mínimo de características naturais (textura) e artificiais (histograma), invariantes à transformações geométricas, que definam a cena (ou objeto), para formar os vetores de atributos destinados ao treinamento e aprendizagem de redes neurais, utilizando ainda a técnica de invariância pelo treinamento. E para melhorar ainda mais a eficiência recorreu-se a utilização do paralelismo de hardware e software, proporcionando uma aplicação para multicomputadores.

ABSTRACT

The present work presents the development of an intelligent Artificial Vision system seeking larger processing speed, smaller cost and increase in the industrial productivity. For the development of the system, a parallel computer was used, the SPP3 developed in LCAD (Laboratory of Computation of High Performance computing) of the ICMSC (Institute of Sciences and Mathematics of São Carlos). The SPP3 uses a parallel architecture MIMD with distributed memory and the a high-speed Myrinet communication network [TRINDADE, 1994].

This Thesis's work has with plan the development of a vision's sistem in real time. To attend the objetives cited, it was realized of original form, the utilization of statistical methods in extraction of less natural feature (texture) and artificial (histogram) invariants for geometrics transformations, that define the scene (or objets), to form the atribut's vectors destinated for training and learning of neural networks, still using the invariants technical for training. And to improve more the eficient have recourse to use parallelism of hardware and software, giving a aplication to multicomputers.

SERVIÇO DE BIBLIOTECA
INFORMAÇÃO

CAPÍTULO 1

1 INTRODUÇÃO

1.1 OBJETIVO:

Desenvolver um sistema de Visão Artificial inteligente para o reconhecimento de objetos, atendendo a sistemas mecânicos de montagem. Em tais sistemas, a utilização da visão artificial torna-se necessária para trabalhar com problemas relacionados à tolerância das peças, peças defeituosas, alem de erros de posicionamento das mesmas [DAVIES & GIIL, 1993].

O sistema de Visão Artificial servirá para implementação de robô inteligente, fornecendo a descrição de uma imagem após processamento e análise da mesma, que será utilizado em robô para aplicações envolvendo tarefas que incluem a manipulação de peças em operações repetitivas, montagem de produtos e tarefas de inspeção visual em Manufatura Integrada por Computador – CIM (Computer Intregraded Manufacturing). Este sistema poderá ter aplicações em: soldagem, pintura, análise de cenas, furação, controle de trajetória, medicina, etc.

Através do processo de inspeção visual determina-se se um produto ou peça, desvia-se de um conjunto de especificações, envolvendo normalmente medidas de características específicas, tais como integridade de montagem, acabamento superficial e dimensões geométricas.

A aplicação deste projeto deverá proporcionar um aumento na produtividade, qualidade do produto, redução dos custos de produção e tempo de entrega de um produto, tornando-o mais competitivo.

Fazer um estudo sobre o uso de técnicas de Redes Neurais na utilização de aplicação do sistema.

Implementar um Sistema de Visão Inteligente de Alto Desempenho, com características de invariância geométrica ao tratar com as imagens de peças transladadas, escalonadas e rotacionadas em relação à posição original, integrando hardware e software, para se obter Processamento em Tempo Real, utilizando uma câmera, um multicomputador (SPP3), redes neurais e processamento paralelo.

O sistema deverá trabalhar com diversas classes de peças, de diversos tamanhos e formas.

O sistema final, em função de sua flexibilidade, poderá ser adequado a outras aplicações industriais, além de montagem automatizada envolvendo classificação e análise de imagem, tais

como área médica (diagnóstico por imagem), e de segurança (identificação baseada em padrões da íris, impressões digitais, entre outros).

1.2 CONSTITUIÇÃO DO SISTEMA

O sistema completo foi implementado utilizando um Multicomputador Paralelo (SPP3), uma Câmera CCD, Sistema de Iluminação, uma Placa de Captura de Imagem, a ser ligado futuramente com um controlador do robô, e o robô.

A Arquitetura Paralela que foi utilizada é um Multicomputador de Alto Desempenho (SPP3) desenvolvido pelo LCAD – ICMSC – USP, que consiste em uma arquitetura paralela MIND com memória distribuída.

A versão inicial do SPP3 suporta até 256 nós processadores distribuídos e módulos de 16 processadores cada um. Uma máquina com um único módulo pode conter até 32 processadores que serão interligados por duas redes de comunicação: uma rede Ethernet, utilizada para controle e gerenciamento do sistema e uma rede de comunicação de alto desempenho.

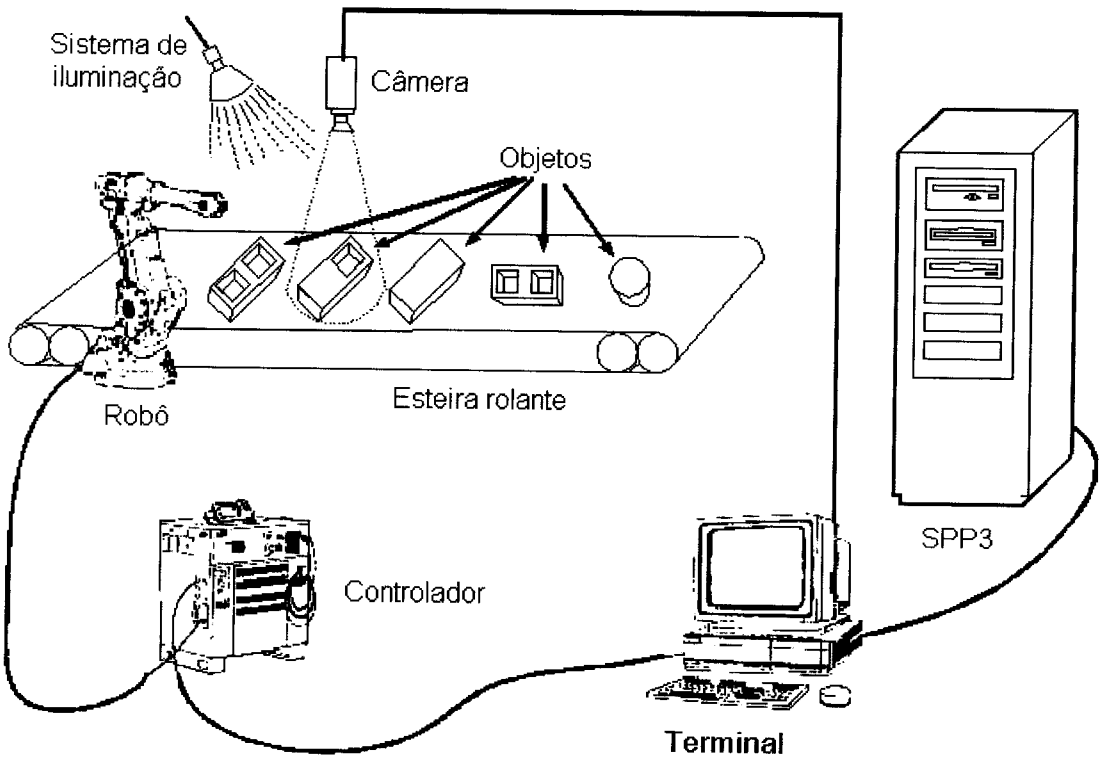


Fig. 1 – Utilização básica para o Sistema de Visão Implementado

O Sistema de iluminação foi montado, com a finalidade de se obter imagens bem definidas, homogêneas e imunes a variações luminosas do meio ambiente.

A Câmera é utilizada para capturar uma imagem que será armazenada e acessada pelo sistema de processamento e análise.
Depois que a imagem é capturada pela câmera e armazenada, ela deve ser processada para extração da informação nela contida, envolvendo um estágio de pré-processamento, para correção de distorções em um estágio de extração de características, no qual características determinantes da informação de imagem são extraídas e utilizadas para propósitos de classificação.
Para a extração de informação tri-dimensional de um objeto, o sistema deve ser capaz de medir algumas dimensões específicas de uma classe determinada de objetos, na qual deverão ser utilizadas as técnicas de visão estéreo para que a informação de profundidade possa ser obtida.

A Placa de Captura de Imagens, deverá digitalizar imagens a um ritmo de trinta quadros por segundo. Os quadros consistem em uma matriz de dados que representam projeções de cena captada pela câmera. Os elementos da matriz são chamados de elementos de imagem ou pixels. O número de pixels é determinado por um processo de amostragem realizado em cada quadro de imagem. Um pixel é a projeção de uma pequena porção da cena que reduz essa porção a um valor unitário. O valor é a medida da intensidade de luz para esse elemento. Cada intensidade de pixel é convertida num valor digital.

O Robô a ser utilizado futuramente será o **IRB2400** da ABB Robotics AB, com seis graus de liberdade e que permite comunicação com PC remoto, PLC ou manualmente no local.

1.3 ORGANIZAÇÃO DA DISSERTAÇÃO

Esta tese inicia-se com uma introdução, contendo o objetivo e uma breve descrição do sistema de visão computacional utilizado.

No capítulo 2 serão apresentados os conceitos relacionados à aplicação e construção de sistemas paralelos, descrevendo as arquiteturas e suas principais características e ferramentas utilizadas para a construção de algoritmos paralelos.

No terceiro capítulo é apresentado o conceito de processamento paralelo, uma descrição dos tipos de paralelismo, um breve comentário sobre concorrência e paralelismo, a definição de

granularidade, os conceitos de *speed-up* e eficiência e uma revisão bibliográfica sobre as arquiteturas para processamento paralelo, bem como uma descrição que focaliza e descreve a organização e implementação do multicomputador SPP3, utilizado neste projeto.

O capítulo 4 apresenta uma revisão bibliográfica sobre programação paralela.

No capítulo 5 são apresentados os conceitos sobre Visão Computacional. Neste capítulo é feita uma abordagem sobre o reconhecimento de padrões, dando-se ênfase à utilização de redes neurais.

O capítulo 6 apresenta o sistema implementado de visão computacional, detalhando o sistema de aquisição da imagem (sistema de iluminação, câmera e placa de captura de vídeo), utilizando redes neurais implementadas em processamento paralelo e sua aplicação.

O capítulo 7 apresenta a realização prática do desenvolvimento experimental e análise dos resultados.

No capítulo 8 são apresentadas as considerações finais, contribuições e sugestões para futuros trabalhos.

O apêndice A, apresenta algumas imagens utilizadas, pois para treinamento, validação e teste da rede neural, utiliza-se um número elevado de padrões (imagens), para se ter um desempenho eficiente da mesma, justificando assim, a necessidade de se utilizar do processamento paralelo, para que se reduza o tempo de pré-processamento e realização dos referidos testes.

No apêndice B encontram-se algumas tabelas de características utilizadas e alguns resultados experimentais.

O apêndice C apresenta um complemento sobre a detecção da imagem.

E no final, temos o apêndice D com os algoritmos e fragmentos de códigos fonte, utilizados nesta tese.

CAPÍTULO 2

2 COMPUTAÇÃO PARALELA

2.1 INTRODUÇÃO

Nestes últimos anos o aumento do poder computacional tem ocorrido em grande parte devido à utilização de técnicas de processamento paralelo aplicadas á arquitetura interna do processador, o qual possui funções que trabalham de modo concorrente. No entanto, tem-se explorado técnicas de processamento na construção de computadores paralelos que em vez de uma única UCP (unidade central de processamento) de alta velocidade, utilizam várias UCPs completas e mais lentas, obtendo-se um poder computacional equivalente a supercomputadores, a um custo menor.

2.2 REDES DE COMUNICAÇÃO DE ALTA VELOCIDADE

As redes de alta velocidade têm evoluído no aspecto desempenho e à queda de custo, tornando-as mais acessíveis, permitindo sua utilização em muitas aplicações, como no caso da computação de alto desempenho utilizando-se máquinas MIMD com memória distribuída. Nos tópicos seguintes temos uma breve descrição das redes ATOMIC, MYRINET, ATM, SCI, HIPPI, FIBER CHANEL, e justificativas que levam a escolha do padrão de rede Myrinet no SPP3.

2.2.1 ATOMIC

A rede *Atomic* é uma rede local de altíssima velocidade, de baixo custo, com capacidade de transmitir dados de vídeo digital, multimídia e aplicações distribuídas com grandes taxas de transmissão de dados.

A rede *Atomic* utiliza técnicas de computação paralela atingindo desempenho bem maior do que o apresentado por redes locais convencionais. Existe um protótipo operacional na USC/ISI (University of Southern California / Information Sciences Institute) desde outubro de 1991, com tecnologia baseada no projeto Mosaic, sendo um computador de granularidade fina que se comunica através de uma rede de troca de mensagens [FLORISI, 1994].

Quanto ao desempenho, em pacotes de 1500 bytes (tamanhos típicos de pacotes FTP) são transferidos através de um canal *Mosaic*, obtém-se uma taxa de transferência de 480 Mbits/s, e

IFSC-USP SERVIÇO DE BIBLIOTECA INFORMAÇÃO

para pacotes de 54 bytes, tais como pacotes ATM, a taxa de transferência é de 205 Mbits/s [FELDERMAN, 1994].

2.2.2 MYRINET

A rede local *Myrinet* trata-se de uma rede local de alta velocidade, de baixo custo, baseada na tecnologia para comunicação de pacotes e *switching* (chaveamento) em MPPs (Massively Parallel Processors – Processadores Maciçamente Paralelos), resultante dos projetos de pesquisa MOSAIC CALTECH (Multicomputador de granularidade fina) [SEITZ, 1993] e Rede Local ATOMIC [FELDERMAN, 1994] de responsabilidade da ARPA (Advanced Research Projects Agency).

A rede *Myrinet* atende às necessidades das aplicações de computação distribuída, cluster de computadores, servidores de arquivos de imagens, atualmente concectados por redes de 10 ou 100 Mbits/s.

A rede *Myrinet* é também uma rede em nível de sistema e pode-se agrupar computadores em um único gabinete com o mesmo desempenho de uma rede local, e a um custo mais baixo.

Utilizando-se softwares especializados tem-se obtido tempos de latência na ordem de 5 μs para pequenas mensagens entre processos de usuários no sistema Unix. Esta rede apresenta um desempenho melhor do que a maioria dos MPPs de memória distribuída, tais como o Intel Paragon e o IBM SP-1/2, com taxas que podem exceder a 1 Gbits/s [WHY,1998]. Podemos encontrar várias interfaces de software de MPPs de memória distribuída, tais como PVM (Parallel Virtual Machine) e MPI (Message Passing Interface)[MPI, 1995], disponíveis para o uso em uma rede *Myrinet*.

Pode-se obter alto desempenho, utilizando-se de APIs (Application Programming Interface) que evitam as chamadas ao sistema operacional. O MCP (Myrinet Control Processors - programa carregado no chip da LAN na placa de interface do *host*, cuja função é receber mensagens da *workstation* e enviá-las para a *Myrinet* ou receber mensagens da *Myrinet* e enviá-las para a *workstation*, podendo, alem disso, mapear e monitorar a rede continuamente) ativa a interface que passa a interagir diretamente com o processo do usuário, atingindo, taxas de transferências acima de 550 Mbits/s.

2.2.3 ATM – ASYNCHRONOUS TRANSFER MODE

A rede ATM (Asynchronous Transfer Mode), veio a atender principalmente a novas aplicações, como as de multimídia e as de tempo real, que exigem cada vez mais velocidade e alta capacidade de transmissão com garantia de resposta, não atendidas pelas redes tradicionais [LIEIRA, 1995]. Esta rede possui como característica principal em apresentar flexibilidade em adaptar-se a diversas plataformas e protocolos existentes [KARVÉ, 1997][LIBOUDEC, 1992].

A velocidade de 155,52 Mbits/s possibilita a transmissão de imagens de vídeo em alta resolução e proporciona compatibilidade com sistema de transmissão SONET [TANENBAUM, 1997], sendo utilizada a velocidade de 622 Mbits/s que possibilita a sua utilização com quatro canais de 155 Mbits/s.

2.2.4 SCI – SCALABLE COHERENT INTERFACE

A interface de comunicação SCI (Scalable Coherent Interface), visa três aspectos principais:

a) Escalabilidade: exibir alto desempenho para sistemas com pequeno ou grande número de processadores.
b) Coerência de cache: em sistemas distribuídos com memória compartilhada, a coerência deve ser mantida pela rede de forma transparente para o processador, a qual garante confiabilidade, possibilitando sua implementação em sistemas para processamento paralelo.
c) Interface: elas não devem ser vinculadas a uma tecnologia em particular, o que possibilita a implementação de interfaces de acordo com cada necessidade.

2.2.5 HIPPI – HIGH PERFORMANCE PARALLEL INTERFACE

A HIPPI (High Performance Parallel Interface) surgiu a partir de 1987 em Los Alamos National Laboratory, com a necessidade de se buscar um padrão para interligação de supercomputadores. Em princípio, buscava-se uma taxa de transferência de dados de 800 Mbits/s para visualizar cenas com quadros de 1.024 x 1.024 pontos, com 24 bits por ponto e 30 quadros/s, utilizando-se barramento de 32 bits, representando uma taxa de dados agregada de 750 Mbits/s. Posteriormente, foi proposto um padrão com uma taxa de dados de 1.600 Mbits/s que passou a ser chamado de padrão HIPPI [TANENBAUM, 1997], fazendo uso de um

barramento de dados de 64 bits. Este padrão foi originalmente projetado para ser um canal de dados, sendo utilizado posteriormente em redes locais.

2.2.6 FIBER CHANNEL

O FIBER CHANNEL (canal de fibra), surgiu através da utilização de fibra ótica que possibilitou a substituição de 50 ou 100 fios do tipo par trançado por uma única fibra, passando a ser a sucessora do padrão HIPPI.

Este padrão é aplicado em conexão serial de alto desempenho suportando protocolos de camadas mais altas, incluindo o HIPPI, IPI (Intelligente Peripheral Interface), SCSI (Small Computer System Interface) e o canal multiplexador usado nos mainframes da IBM [TANENBAUM, 1997]. O FIBER CHANNEL, pode se utilizado para transportar pacotes do tipo IEEE 802, IP e ATM, possibilitando classes de serviço de entrega de pacotes na ordem em que foram transmitidos, com ou sem entrega garantida.

2.2.7 CARACTERÍSTICAS PRINCIPAIS DA REDES APRESENTADAS

	MYRINET	ATM	SCI	HIPPI
Meios Físicos	Cabos elétricos Fibra Ótica	Cabos elétricos Fibra Ótica	Cabos elétricos Fibra Ótica	Cabos elétricos Fibra Ótica
Distância Máxima para cabos elétricos	Menor ou igual a 10 metros	200 metros	Menor que 20 Metros	Menor ou igual a 25 metros
Distância Máxima para fibra óptica	Em projeto	2 Quilômetros	-	300 metros
Modo de Trans-missão	Paralela, 18 bits, 9 Bits em cada direção	Serial	Paralela de 18 bits	Paralela de 50/100 bits ou serial
Topologia	Ponto-a-ponto	Ponto-a-ponto	Ponto-a-ponto	Ponto-a-ponto
Velocidade	1.28 + 1.28 Gibs/s	622 Mbits/s	500 Mbits/s	800 Mbits/s
Latência	7.2 µs		4 µs	160 ns
Disponibilidade de Interface PCI	Sim	Sim	Sim	Sim
Custo aproximado por nó	$1.800	$4.200		>$4.000

Tabela : características principais das redes Myrinet, ATM, SCI e Hippi

Como a velocidade de transmissão de dados e a latência de transmissão são as duas características mais importantes para redes de alto desempenho ao se tratar de máquinas paralelas, pode-se verificar no quadro acima, que para as redes analisadas, a rede *Myrinet* se destaca entre elas como uma rede de maior velocidade e menor custo por nó, apesar de existir no mercado redes ATM operando a 2.5 Gbits/s, porém com um preço bem superior.

Em 1998, John Rynearson, diretor técnico da VITA Standards Organization, anunciou a transformação de rede *Myrinet* em um padrão nacional americano, reconhecido como ANSI/VITA 26, sendo o protocolo Myrinet-on-VME oficialmente aprovado em 02 de novembro de 1998 [ANSI, 1998].

O empenho da *Myricom* e diversas instituições de pesquisa para aprimorar e desenvolver novos componentes e protocolos para aumentar o desempenho da rede *Myrinet*, tem levado esta rede a ocupar lugar de destaque no meio das redes de comunicação de alta velocidade [MYRINET, 1998], sendo utilizada até o início de 1997, por mais de 150 instituições. A seguir temos alguns sistemas que utilizam esse padrão:

- Projeto NOW (Network of Workstation) da University of California at Berkeley, que consiste em um conjunto de workstations (100 SUN Ultrasparc e 40 Sparcstation executando Solaris, 35 PCs Intel com Windows NT ou UNIX) interligados por uma rede comutada de alta velocidade, a Myrinet, envolvendo pesquisas em áreas como redes de comunicação de alta velocidade, sistemas operacionais distribuídos, sistemas de arquivos escaláveis, web sobre NOW, ambientes de programação e aplicações, tendo como objetivo alcançar melhor custo/desempenho para aplicações paralelas que as oferecidas pelas arquiteturas MPPs e melhor desempenho para aplicações seqüenciais que as workstations individuais podem oferecer [BERKELEY, 1998].

- Projeto Netstation, da University of Southern California/Information Sciences Institute (ISI), que utiliza a rede Myrinet para criar um malha E/S para uma estação de trabalho de alto desempenho. O ISI implementou um protocolo de transferência confiável sobre a API Myrinet, que utiliza PVM sobre o padrão Myrinet trabalhando com um protocolo de alta velocidade para se conseguir uma maior velocidade do referido padrão [NETSTATION, 1997].

- Projeto DAS (Distributed ASCI Supercomputer) da ASCI (Advanced School for Computing and Imaging), constituido de quatro clusters compostos por 200 nós de processamento, sendo a interconexão de cada cluster efetuada através do padrão

Myrinet/Fast Ethernet, localizados em diferentes universidades, situadas em uma área geográfica ampla, utilizando o sistema operacional BSD/OS (v.3.0) [DAS, 1998].

- Projeto HPVM (High Performance Virtual Machines) da University of Illinois, consistindo em conjuntos de processadores Pentium executando os sistemas operacionais *Linux* ou Windows NT, conectados por uma rede *Myrinet* de comunicação de dados em alta velocidade. Neste projeto utilizam-se as interfaces MPI, SHMEM, Global *Arrays* e *Fast Messages* para o desenvolvimento de aplicações paralelas, bem como devendo ser incorporado a versão HPF (High Performance Fortran) [ILLINOIS, 1997].

- Projeto MOSIX – *Multicomputer Operating System for UNIX*, desenvolvido pela Hebrew University of Jerusalem, que tem como objetivo modificar o UNIX para que a rede de *workstation* (NOW) [BERKELEY, 1998] venha a funcionar como um multicomputador de recursos escalonável, sendo implementado em diferentes versões do UNIX e em diversas arquiteturas de computadores [HEBREW, 1998].

- Projeto TRAPEZE, da University Duke, que tem a finalidade de desenvolver novas técnicas para comunicação de alta velocidade e acesso rápido para dados armazenados em clusters de estações de trabalho com sistemas UNIX, utilizando como plataforma de trabalho um cluster DEC Alpha e estações de trabalho baseadas na tecnologia Intel interconectadas por uma rede *Myrinet*, desenvolvendo também drivers para sistemas DEC Alpha e um MCP completo, otimizado par obter comunicação sem cópia via RPC [ANDERSON, 1998].

- Projeto de um novo MCP de propósito geral da Mississippi State University, em desenvolvimento em seu laboratório ICDCR (Integrated Concurrent and Distributed Computation Research), disponibilizando implementações do Packet e MPI++, migrando para a rede *Myrinet*, código fonte do MPI++ baseado em objetos, já testadas sobre Suns/Solaris e LANai v.3.x e Windows NT, conjunto de tutoriais e demonstrações do MCP [INTEGRATED, 1998].

Analisando as características e descrições acima, optou-se pelo uso das redes *Myrinet* na construção do SPP3, pois, apresenta disponibilidade de interface para barramento PCI [CLARK, 1997], que consiste de um fator importante do projeto, uma vez que esse era na época o único de

IFSC-USP SERVIÇO DE BIBLIOTECA INFORMAÇÃO

alto desempenho disponível nos computadores pessoais baseados na arquitetura Intel, satisfazendo as características de velocidade e latência necessárias para um ótimo desempenho.

CAPÍTULO 3

3 PROCESSAMENTO PARALELO

3.1 - INTRODUÇÃO

O conceito de utilização do paralelismo no processo computacional surgiu no ano de 1920, quando Vanevar Bush, pesquisador do MIT (Massachussets Institute of Tecnology), apresentou um computador analógico capaz de resolver equações diferenciais em paralelo, sendo que nos anos 40, Von Neumann, criador da arquitetura von Neumann, utilizada como arquitetura fundamental da maioria dos computadores até hoje, sugeriu em seus artigos, uma grade para resolver equações diferenciais em que os pontos são atualizados em paralelo. No entanto, o computador ILLIAC IV, construído na Universidade de Illinois, composto de 64 processadores, em 1971 [AMORIM, 1988][NAVAUX, 1989], é considerado como o marco inicial do processamento paralelo.

Processamento paralelo é definido por Hwang [HWANG, 1984] como: "forma eficiente de processamento da informação com ênfase na exploração de eventos concorrentes no processo computacional".

O principal fator que justifica a necessidade do processamento paralelo é a busca por um maior desempenho [ALMASI, 1994][CORNELL, 1996][ZALUSKA, 1991], em aplicações que demandam grande poder computacional. Devido a limitações físicas como a velocidade finita da luz, o poder computacional necessário para execução dessas aplicações poderá não ser atingido pelas arquiteturas seqüenciais. A computação paralela possibilita meios mais comuns de se obter tolerância a falhas, utilizando a redundância de elementos de hardware e recursos de software embutidos na execução de aplicações intrinsecamente paralelas [ALMASI, 1994]. Nas diversas áreas de aplicação da computação paralela como por exemplo: prognóstico do tempo, controle de reatores, modelagem de reservatórios de petróleo, decisões das operações de trajetos a serem feitos por ônibus espaciais ou aviões modernos, bem como nas áreas científicas, industriais ou militares, requerem cada vez mais poder computacional, devido a complexos algoritmos que são utilizados e do tamanho do conjunto de dados a ser processado [KIRNER, 1991].

Outros fatores que podemos citar que contribuem para a necessidade e desenvolvimento da computação paralela [ALMASI, 1994][AMORIM, 1988] são:

- Surgimento da tecnologia VLSI e projeto de microprocessadores, que permitiu a construção de microprocessadores de alto desempenho, que agrupados

adequadamente, possibilitam um ganho significativo de poder computacional, possibilitando uma melhor relação preço/desempenho quando comparados ao custo elevado dos supercomputadores [ZALUSKA, 1991].

- Restrições físicas, como a velocidade finita da luz, tornam difícil o aumento de velocidade em um único processador.

- Possibilidade de configuração modular, permitindo o agrupamento de processadores em módulos de acordo com a natureza da aplicação.

- Tolerância a falhas, como por exemplo, através de redundância de hardware.

3.2 - TIPOS DE PARALELISMO

Dependendo da arquitetura e do mecanismo de escalonamento de processos do sistema operacional de um computador, o processamento de tarefas ou instruções pode ser seqüencial ou concorrente, sendo que no primeiro caso um evento somente é iniciado após o término de um outro evento, e no segundo, em um determinado instante, dois ou mais processos foram iniciados e ainda não foram terminados. A partir destas definições, temos o paralelismo lógico, físico e misto [KIRNER, 1991].

- **Paralelismo Lógico**: os processos são executados intercaladamente através de um **único elemento de processamento**, estando ativo apenas um deles a cada instante. Na figura 3.1 abaixo, temos um exemplo para a representação de três processos (P_1, P_2 e P_3), sendo executados em paralelo em função do tempo.

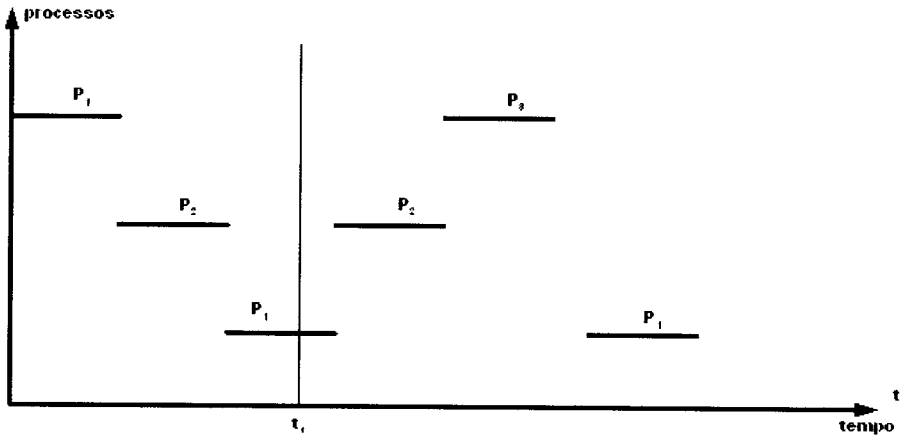


Figura 3.1 - Paralelismo Lógico

- **Paralelismo físico**: os processos são executados simultaneamente, através de **vários elementos de processamento**, sendo que existe um elemento de processamento para cada processo que está sendo executado, podendo estar ativos simultaneamente (figura 3.2).

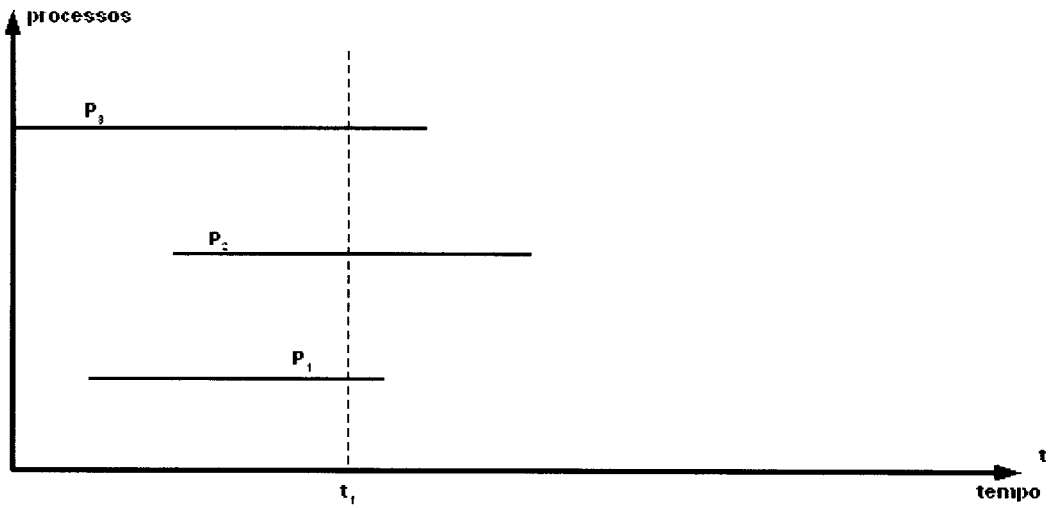


Figura 3.2 - Paralelismo Físico

- **Paralelismo misto:** os processos sendo executados de forma intercalada ou simultaneamente, realizando a combinação do paralelismo físico com o lógico, onde, por exemplo, teremos para n < m, m processos sendo executados em n elementos de processamento (figura 3.3).

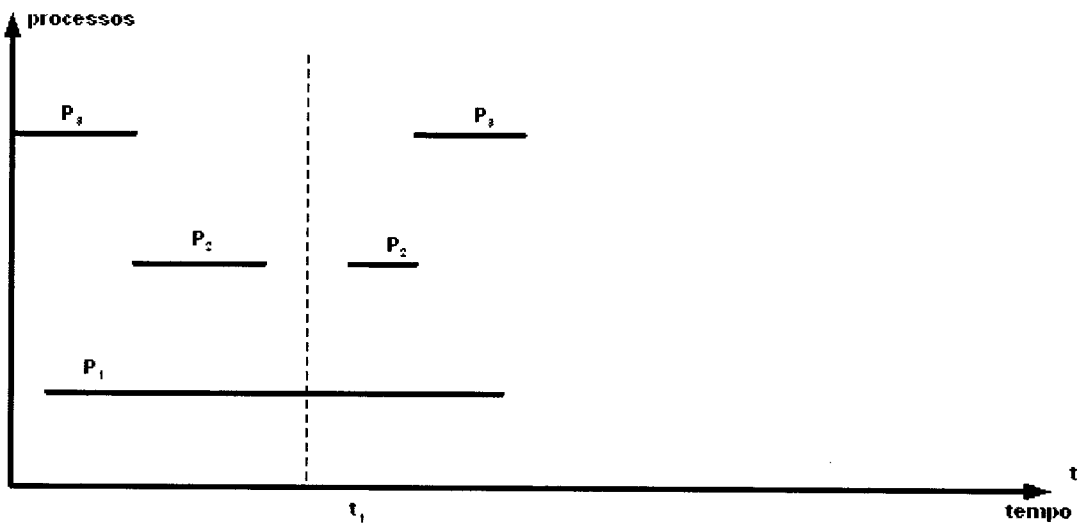


Figura 3.3 - Paralelismo misto.

Dentro desse mesmo conceito, o paralelismo físico pode ser dividido em espacial, temporal (pipeline) ou combinado, definidos a seguir:

- **Paralelismo físico espacial** – quando a execução dos processos são realizados simultaneamente em função do tempo, de forma síncrona ou assíncrona. Quando síncrono, os elementos de processamento são ligados a uma única unidade de controle, executando a mesma instrução sincronamente, e no modo assíncrono, os elementos de processamento executam instruções diferentes e interagem entre si quando necessário.

- **Paralelismo físico temporal (pipeline)** – os processos são executados de forma paralela e concorrente. Neste caso, o processo é subdividido em uma seqüência de subprocessos. Cada subprocesso e executado por um estágio especializado de hardware e software que opera paralelamente com os outros estágios do pipeline[HWANG, 1984].

  Para exemplificar o funcionamento de um pipeline, temos nas figuras 3.4 e 3.5 abaixo um processo a ser realizado em 4 estágios (E1, E2, E3 e E4), onde cada estágio é responsável pela execução de um determinado conjunto de operações sobre um certo conjunto de dados pertencentes a uma tarefa. A organização dos estágios é feita de forma seqüencial, onde o estágio posterior é alimentado pelo estágio imediatamente anterior (figura 3.5). Como pode ser visto, a cada instante de tempo, uma nova tarefa (T1, T2, T3,...., Tn) é iniciada. Após quatro (4) unidades de tempo, o fluxo de tarefas completas (tarefas que completaram os quatro (4) estágios) torna-se contínuo, e uma tarefa termina a cada unidade de tempo, resultando um paralelismo de eventos sobrepostos.

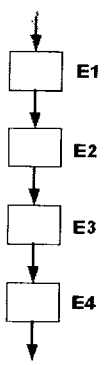


Fig. 3.4-Estágio *pipeline*

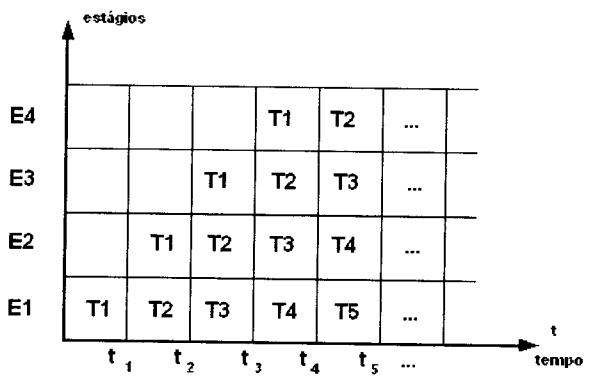


Fig. 3.5-Execução das tarefas em relação ao tempo

Podemos dividir o processamento paralelo em dois grupos o *on-chip* e o *off-chip* (figura 3.6). No paralelismo *on-chip*, a arquitetura interna do processador possui funções que trabalham de modo concorrente. No paralelismo *off-chip*, o processamento é feito com dois ou mais processadores [REIFEL, 1994].

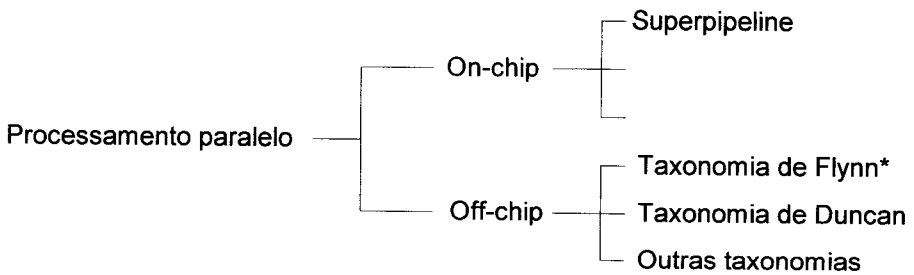


Figura 3.6 - Classificação de Processamento Paralelo

- Processamento paralelo *on-chip*: podemos, por exemplo, encontrar este tipo de processamento nos microprocessadores CISC (*Complex Instruction Set Computer)* de 32 bits, como o 68000, onde as instruções são executadas em *pipeline* (Figura 3.7), e nos microprocessadores RISC (*Reduzed Instruction Set Computer*), onde se utiliza técnicas mais avançadas agrupadas em superpipeline, super-escalonamento e integração de multi-CPU.

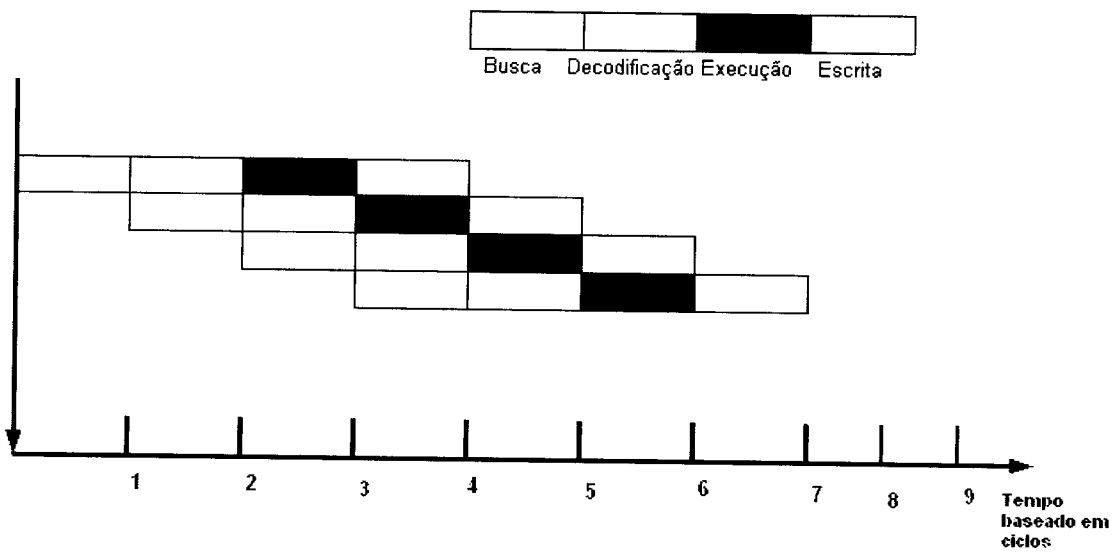


Figura 3.7 - Execução das operações de instruções com *pipeline*

No super-pipeline (figura 3.8) a instrução com pipeline é quebrada em estágios menores, podendo o processador executar muitas instruções simultaneamente em diferentes estágios, enquanto que no super-escalonamento (figura 3.9), cria-se pipelines múltiplos, permitindo que a CPU possa executar várias instruções simultaneamente.

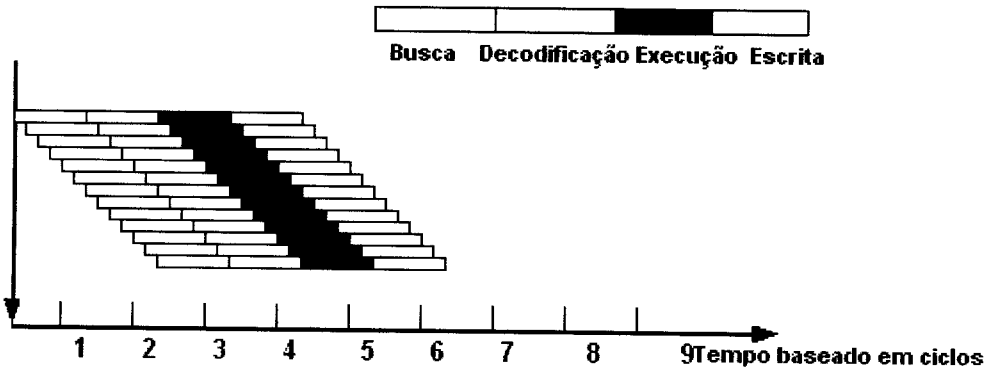


Figura 3.8 - Execuções executadas em super-*pipeline*

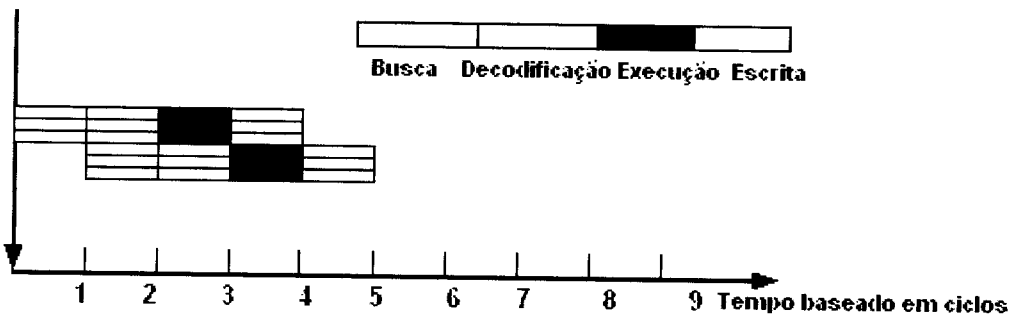


Figura 3.9 - Instruções executadas com super-escalonamento

Na integração de Multi-CPU, integram-se várias CPUs em um pedaço de silício, onde o número máximo de processadores varia de acordo com o tamanho da pastilha, dissipação de potência e número de pinos.

- Processamento paralelo *off-chip*: dois ou mais processadores executam uma ou várias instruções simultaneamente, tendo início no ano de 1920 com Vanevar Bush do MIT

(*Massachussets Institute of Tecnology*) e posteriormente em arquiteturas de computadores paralelos, proporcionando grandes benefícios, tais como: Expansibilidade (possibilidade de adicionar mais processadores no sistema para se conseguir o desempenho pretendido), Flexibilidade (pode-se implementar um grande número de topologias diferentes para aplicações dedicadas) e Upgradability (o sistema é projetado de modo modular, permitindo a atualização de processadores, para se conseguir um maior poder de processamento).

3.3 - CONCORRÊNCIA ou PARALELISMO?

O termo processamento paralelo, na literatura computacional, é normalmente utilizado para designar paralelismo físico, tornando-se importante ressaltar a diferença entre "processamento paralelo" e "processamento concorrente", pois às vezes estas expressões são utilizadas de forma inadequada. Logo, a seguir, temos a definição de cada uma delas.

- **Processamento concorrente** – em um determinado instante inicia-se dois ou mais eventos, mas ainda não são completados, não havendo necessariamente o uso de mais de um elemento de processamento (há uso do mecanismo de tempo compartilhado quando há um único elemento de processamento).

- **Processamento paralelo** – é um caso particular da execução concorrente onde existem processos sendo executados simultaneamente por elementos de processamento diferentes.

O termo "processamento paralelo" será utilizado neste trabalho para se referenciar a arquiteturas, programação e ferramentas para processamento paralelo, devido à natureza do SPP3, uma vez que em arquiteturas (inclusive a do SPP3) e ferramentas para processamento paralelo são também utilizadas em processamento concorrente.

3.4 - GRANULARIDADE ou NÍVEL DE PARALELISMO

Granuralidade ou nível de paralelismo corresponde ao tamanho da tarefa que será submetida aos processadores, medida em número de instruções executadas [ALMASI, 1994]. Dentro da computação paralela esta definição se torna importante, pois está diretamente

relacionada ao tipo de plataforma (porte e número de processadores) à qual se aplica o paralelismo [KIRNER, 1991].

A definição de granuralidade é encontrada em diversas publicações [CORNELL, 1996][ALMASI, 1994][KIRNER, 1991][NAVAUX, 1989][HWANG, 1984], podendo ser dividida em três níveis: fina, média e grossa.

A granularidade fina relaciona paralelismo ao nível de instruções ou operações implicando em um grande número de processadores pequenos e simples.

A granularidade grossa relaciona paralelismo ao nível de processos e programas, aplicados geralmente a plataformas com poucos processadores grandes e complexos [KIRNER, 1991][NAVAUX, 1989].

A granularidade média situa-se entre a fina e a grossa, com procedimentos executados em paralelo [KIRNER, 1991][NAVAUX, 1989].

3.5 - SPEEDUP e EFICIÊNCIA

A utilização do processamento paralelo é justificada pelo aumento do desempenho na execução de aplicações paralelas em relação às executadas seqüencialmente, para uma mesma tarefa. O acréscimo de desempenho é geralmente quantificado por duas métricas: speedup e eficiência [FOSTER, 1995][ALMASI, 1994].

O speedup é uma medida utilizada para quantificar o aumento de velocidade obtido na execução de um programa paralelo por p processadores em relação à sua execução seqüencial, sendo expresso pela relação:

$$Sp = T_{seq} \ / \ T_{par} \tag{3.1}$$

onde T_{seq} é o tempo de execução do algoritmo paralelo utilizando apenas um processador e T_{par} é o tempo obtido utilizando p processadores. Idealmente, Sp deveria ser igual a p ($Sp = p$), ou seja, o aumento de desempenho deveria ser diretamente proporcional ao número de processadores utilizados na execução do programa paralelo. Mas na prática, essa igualdade é difícil de ser obtida, devido a três fatores: a sobrecarga de comunicação entre os processadores e tarefas, seções de código que não podem ser paralelisadas e a inadequação do grau de paralelismo apresentado pela aplicação ao grau de paralelismo oferecido pela máquina que a executa [ALMASI, 1994]. Mas podem ocorrer casos em que $Sp > p$, e neste caso temos uma anomalia de speedup [FOSTER, 1995].

A eficiência serve para medir de forma mais acurada a qualidade de um algoritmo paralelo, quantificando a utilização dos processadores através da fração do tempo total de execução em que os processadores executam operações úteis:

$$Ep = Sp / p \tag{3.2}$$

Onde Sp é o speedup obtido com p processadores. Logo a eficiência na utilização dos processadores será tão melhor quanto mais se aproximar de 1 (100%).

3.6 - ARQUITETURAS PARA PROCESSAMENTO PARALELO

Devido à grande diversidade de arquiteturas de computadores existentes para atender às diversas necessidades de aplicações e da tecnologia de elementos computacionais (processadores e redes de interconexão, etc) disponíveis, torna-se necessário uma classificação para que se possa fazer estudo e análise comparativa sobre as mesmas. Na literatura encontramos diversas propostas para se classificar as diferentes arquiteturas e organização de computadores, das quais podemos citar as de: Flynn [Flynn, 1966], Feng [Hawang et al., 1984], Handler [Hawang et al., 1984], Gajski [Amorin et al., 1988], Shore [Hokney & Josshope, 1988], Estrutural [Hokney & Josshope, 1988], Kuk [Almasi & Gottilieb, 1994], Treleaven [Almasi & Gottlieb, 1994], Duncan [Duncan, 1990], etc.

A classificação de Flynn é a mais citada pela literatura, e devido à sua tradição e popularização, tornaram-na até então a mais utilizada e importante taxonomia de computadores, mesmo com suas falhas e dúvidas [Bogni & Marrone, 1991].

Uma classificação mais abrangente, encontramos na taxonomia estrutural de Hockney, que classifica tanto os computadores (históricos) seqüenciais, quanto os computadores paralelos, até os atuais.

Com o passar das décadas de pesquisa e com o surgimento de novas tecnologias, surgiram diversas arquiteturas de computadores paralelos que não mais se enquadravam nas classificações da taxonomia clássica. Com a finalidade de acrescentar essas novas arquiteturas, sem descartar a classificação de Flynn, Duncan [Duncan, 1990] em seu artigo "A Survey of Parallel Computer Architectures", propõe uma classificação mais completa, que permite enquadrar os estilos de organização dos computadores da atualidade.

3.6.1 - TAXONOMIA DE FLYNN

Michael J. Flynn em seus artigos [FLYNN, 1966][FLYNN, 1972], propôs um método de classificação de computadores, que consiste na possibilidade de combinação entre um fluxo de instruções atuando sobre um fluxo de dados. Um fluxo de instruções eqüivale a uma seqüência de instruções executadas (em um processador) sobre um fluxo de dados aos quais estas instruções estão relacionadas [ALMASI, 1994][CORNELL, 1996][DUNCAN, 1990][FLYNN, 1972][HWANG, 1984][NAVAUX, 1989].

Baseando-se nas possíveis unicidade e multiplicidade de fluxos de dados e instruções, FLYNN sugeriu quatro classes de computadores:

- SISD (*Single Instruction Stream / Single Data Stream)*, um fluxo de instrução e um fluxo de dado.

- SIMD (*Single Instruction Stream / Multiple Data Stream*), um fluxo de instrução e múltiplos fluxos de dados.

- MISD (*Multiple Instruction Stream / Single Data Stream*), múltiplos fluxos de instrução e um fluxo de dados.

- MIND (*Multiple Instruction Stream / Multiple Data Stream*), múltiplos fluxos de instrução e múltiplos fluxos de dados.

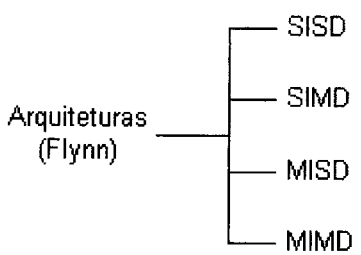


Figura 3.10 – Classificação de Flynn

A seguir, temos a representação dos diagramas correspondentes a cada tipo de arquitetura apresentada acima, contendo três espécies de componentes de sistema: Unidade de Controle (UC), responsável pela decodificação das instruções e pelo envio das instruções decodificadas ao processador; Unidade de Processamento (UP), onde se executa as instruções e armazena os

IFSC-USP SERVIÇO DE BIBLIOTECA
INFORMAÇÃO

resultados na memória e Memória (M), destinada a armazenar dados; (Figuras 3.11, 3.12, 3.13 e 3.14). Nesses diagramas aparece também a representação dos Fluxos de Dados (FD) e os Fluxos de Instruções (FI), onde podemos notar que cada fluxo de instruções é gerado por uma unidade de controle independente.

3.6.1.1 - SISD

Dentro da classificação SISD, temos os computadores seqüenciais convencionais, com uma única unidade de controle que decodifica seqüencialmente instruções que operam sobre um conjunto de dados. Cada instrução inicia uma operação sobre dados obtidos de um único fluxo de dados, como mostra a figura 3.11. A execução das instruções é realizada seqüencialmente, podendo ocorrer sobreposições nos seus estágios de execução através do uso de *pipelines*. A maioria dos microprocessadores atuais SISD possui mais de uma unidade funcional e *pipeline* de instrução [GIMARC & MILUTINOVIC, 1987], estando suas unidades funcionais sob a supervisão de uma única unidade de controle. Como exemplo de computadores SISD, seqüenciais com um único processador, podemos citar a maioria dos modelos pertencentes às famílias APLE e IBM PC.

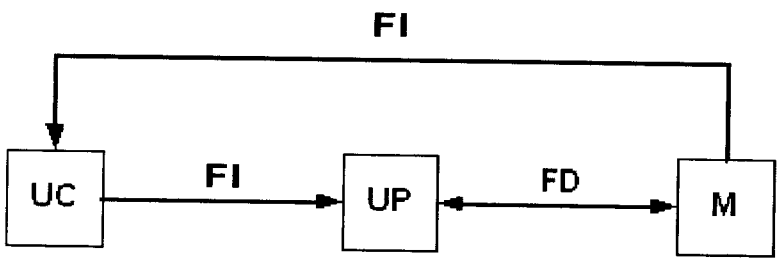


Fig. 3.11 – Diagrama da arquitetura SISD.

3.6.1.2 - SIMD

Nessa classe encontramos os computadores matriciais e vetoriais nos quais vários elementos processadores são conectados por uma única unidade de controle (figura 3.12). A unidade de controle (UC) decodifica seqüencialmente as instruções e as transmite para todos os elementos de processamento, que executam a mesma operação sobre um conjunto distinto de dados.

Podemos citar como exemplo de máquinas SIMD o ILLIAC-IV, considerado o primeiro computador dessa classe, e os mais recentes como o Thinking Machine CM-2 [HWANG, 1993] e MasPar MP [FOSTER, 1995]. Alguns autores classificam máquinas vetoriais pipeline como, por exemplo, o Cray-1 e o Cyber-205 como SIMD [ALMASI & GOTTLIEB, 1994][HOCKNEY & JOSSHOPE, 1988], enquanto outros autores as classificam como SISD [HWANG et al., 1984][BOGNI & MARRONE, 1991].

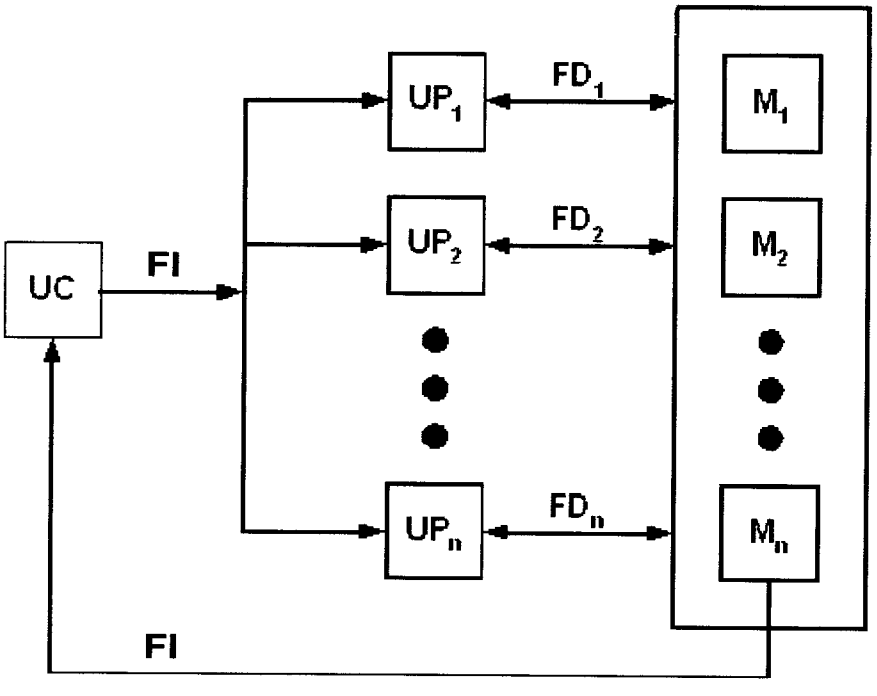


Figura 3.12 – Diagrama da Arquitetura SIMD.

3.6.1.3 - MISD

Essa classe de arquitetura de computadores é apenas teórica, pois corresponde a uma máquina com vários elementos de processamento recebendo instruções distintas das unidades de controle, e processando o mesmo fluxo de dados, onde o resultado de um processador é a entrada do próximo, conforme pode ser vista na figura 3.13. Notamos aí, uma falha de classificação do modelo de Flynn, que não apresenta nenhuma implementação prática [ALMASI & GOTTLIEB, 1994][DUNCAN, 1990][NAVE, 1989][TANENBAUM, 1992].

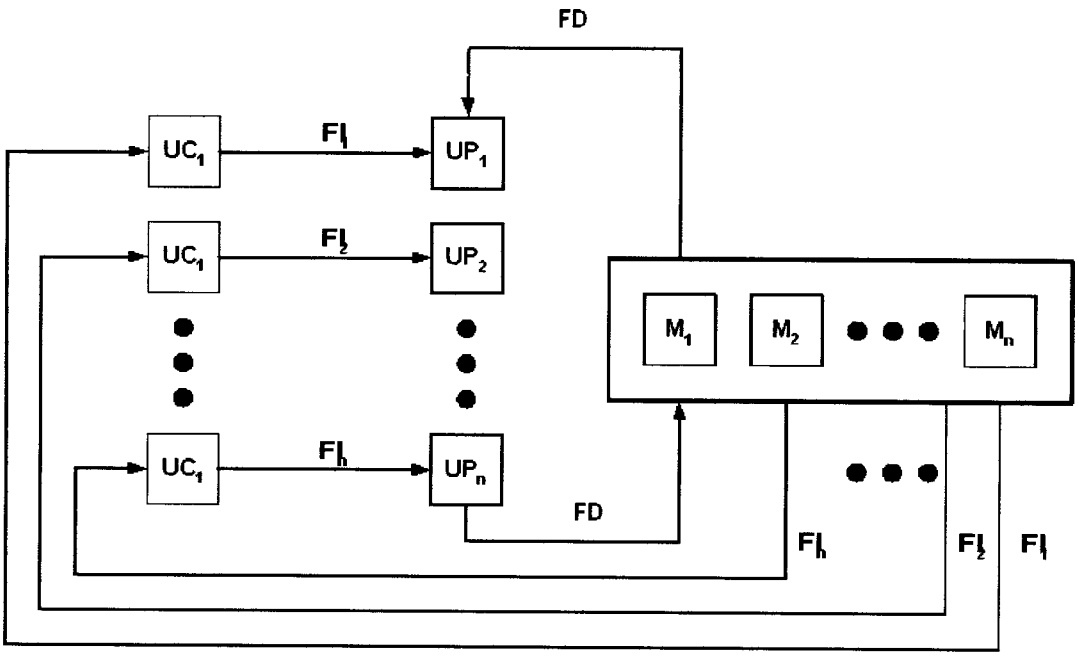


Figura 3.13 – Diagrama da arquitetura MISD.

3.6.1.4 - MIMD

Na classificação MIMD, temos múltiplos processadores executando de forma autônoma diferentes instruções em conjunto de dados distintos (figura 3.14). Pode-se enquadrar nessa classificação a maioria dos sistemas paralelos atuais, ocasionando uma sobrecarga a esta classe, que abrange tanto os multiprocessadores quanto os multicomputadores.

Podemos salientar que nessa classe inclui as máquinas com processadores e unidades de memória especificamente projetadas para serem componentes de uma arquitetura paralela, máquinas paralelas construídas com processadores comerciais, máquinas paralelas construídas com computadores simples como o IBM PC interconectados por uma rede local [FOSTER, 1995] e sistemas distribuídos [TANENBAUM, 1992].

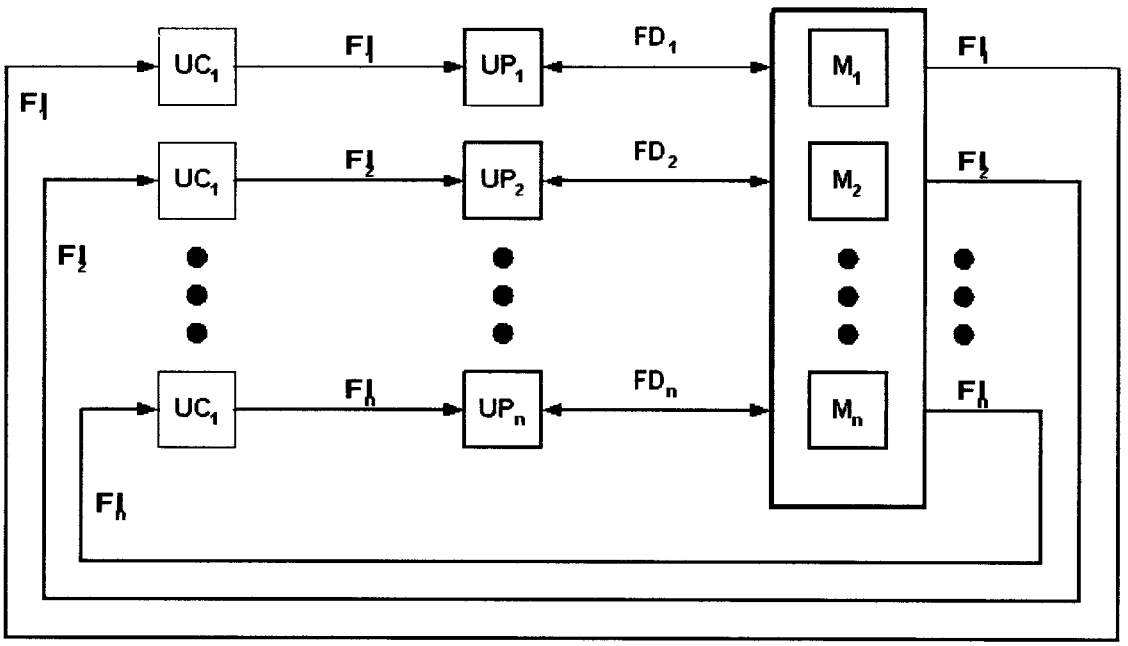


Figura 3.14 - Diagrama da arquitetura MIMD

3.6.2 - TAXONOMIA ESTRUTURAL

A taxonomia estrutural proposta por Hockney & Jesshope [HOCKNEY & JESSHOPE, 1988], foi devido a década de 80, que surgiu uma grande variedade de computadores paralelos(MIMD), abrangendo tanto os computadores seqüenciais quanto os paralelos(Figuras 3.15, 3.16 e 3.17).

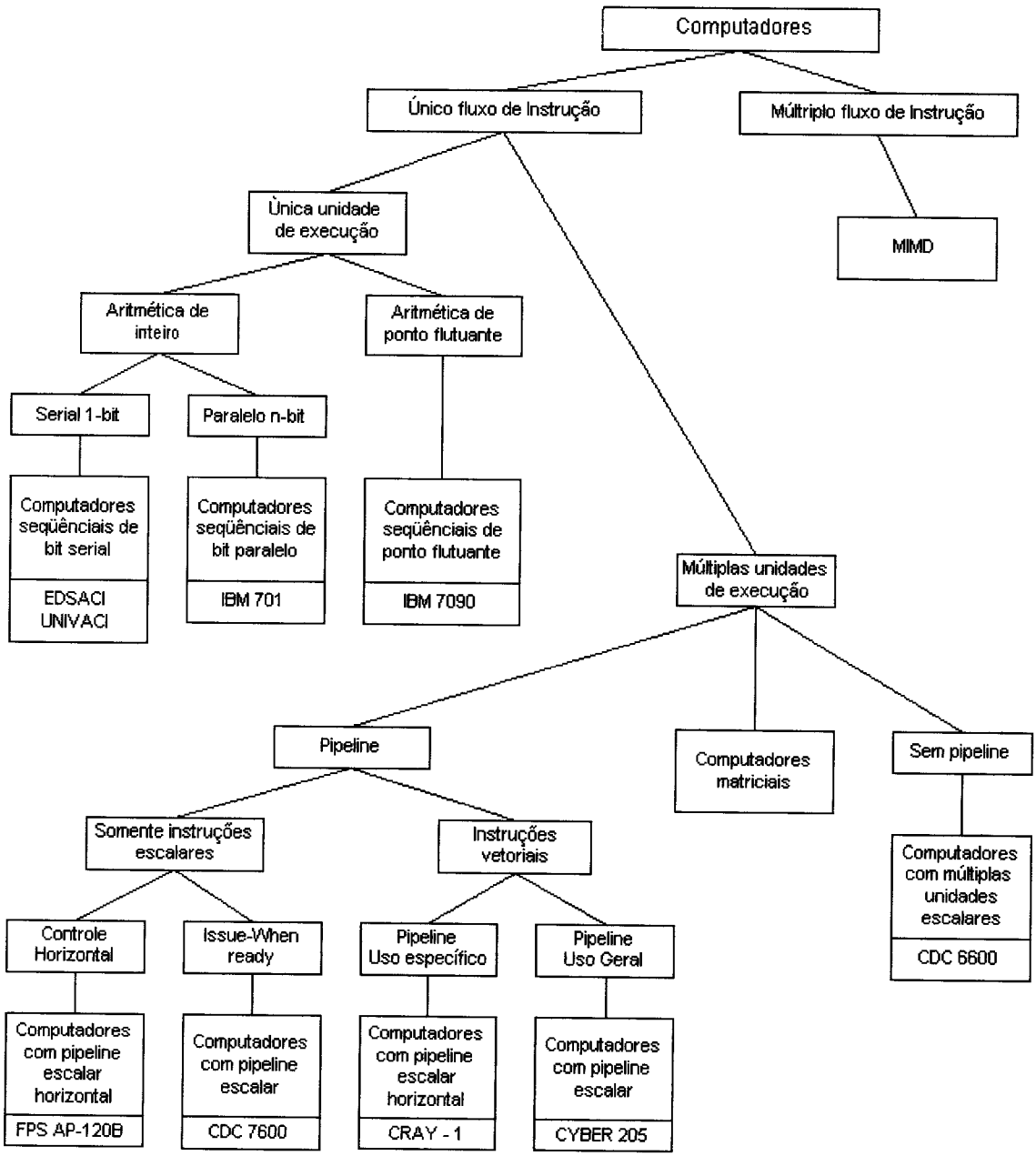


Figura 3.15 – Diagrama da Classificação estrutural.

Como pode ser visto na figura 3.16, os computadores são divididos em dois grandes grupos: máquinas com um único fluxo de instrução e máquinas com diversos fluxos de instrução. Verifica-se que nesta primeira divisão, temos a separação das máquinas MIMD (múltiplo fluxo de instrução) das demais seriais e SIMD (único fluxo de instrução). Nota-se que as máquinas com único fluxo de instrução são então divididas em: única unidade de execução (computadores seqüenciais clássicos) e múltiplas unidades de execução, que é dividida em pipeline, matriciais e sem pipeline.

As máquinas matriciais são divididas em duas classes: máquinas com unidade de execução de ponto flutuante e máquinas com unidades de execução com poucos bits, constituídas por computadores ortogonais e associativos (Figura 3.16).

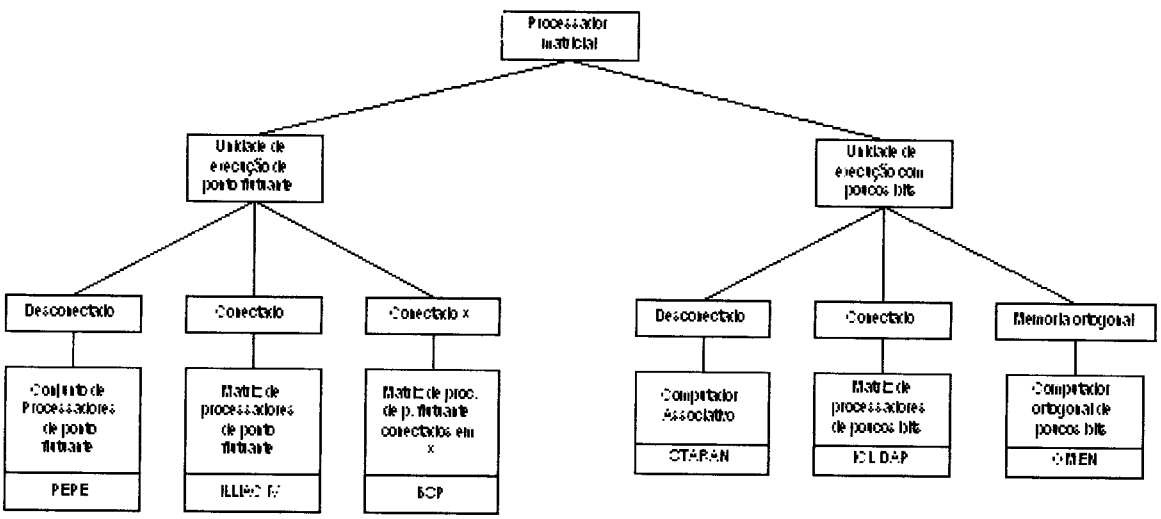


Figura 3.16 – Arquiteturas matriciais.

Na taxonomia estrutural de Hockney as máquinas MIMD são divididas em máquinas chaveadas e máquinas com redes. As máquinas MIMD com memória distribuída e MIMD com memória compartilhada são uma subclasse das máquinas chaveadas, e subdivididas de acordo com o tipo de chaveamento. Verificamos também que as máquinas MIMD com redes são subdivididas em Reconfigurável, Malha (do tipo Anel ou Multidimensional), Cubo (do tipo Hipercubo ou Conectado ciclicamente) e Hierárquica (do tipo Árvore, Pirâmide ou Aglomerado). (Figura 3.18)

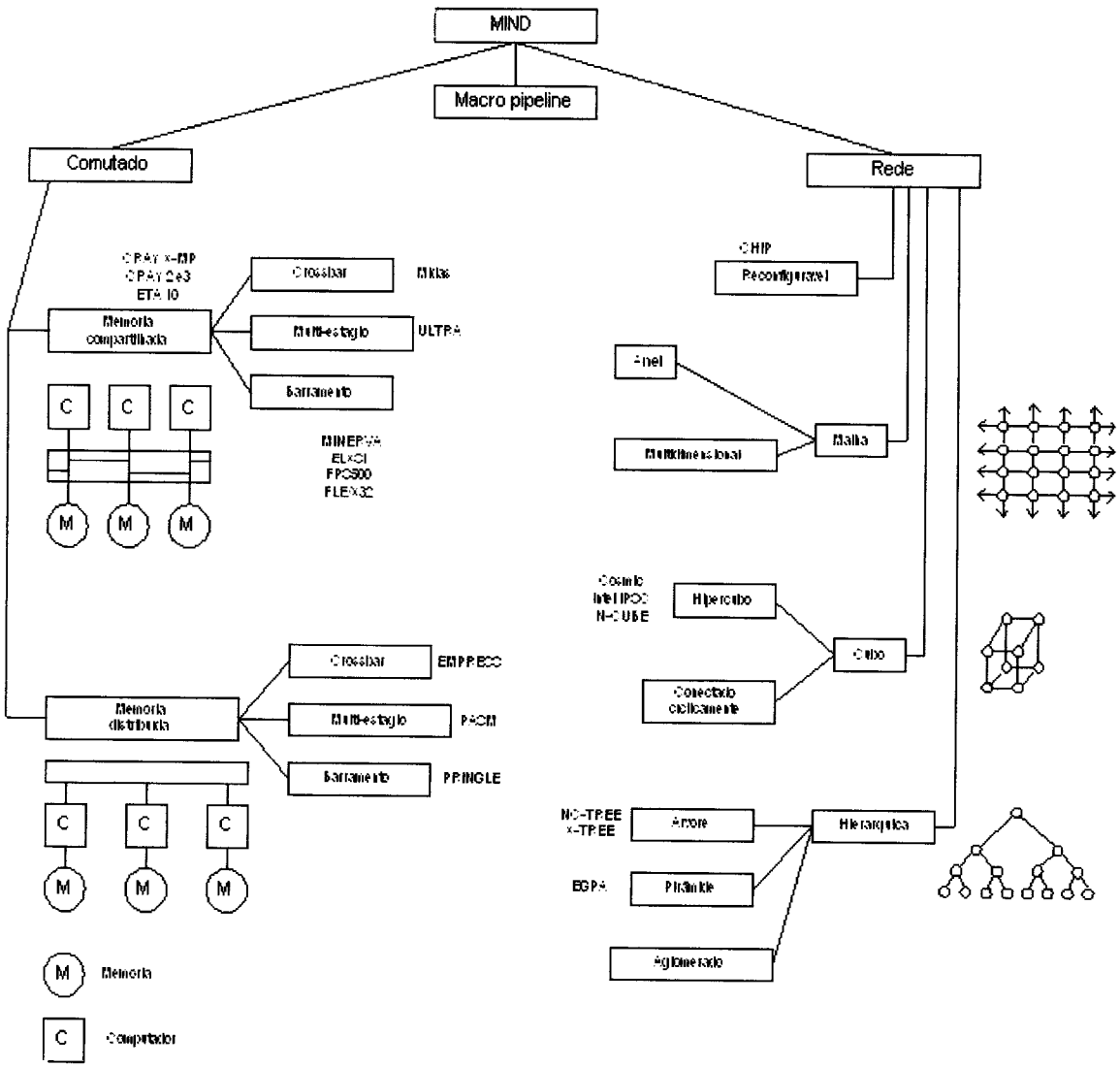


Figura 3.17 - Classificação estrutural para MIMD.

3.6.3 - TAXONOMIA DE DUNCAN

Duncan [DUNCAN, 1990] propôs uma nova classificação para as arquiteturas paralelas, devido às novas descobertas e a proliferação da tecnologia de processamento paralelo, que introduziram novas arquiteturas que não se enquadravam na classificação de arquiteturas paralelas da taxonomia de Flynn.

Na taxonomia de Duncan, foram consideradas as seguintes diretrizes:

- Excluir arquiteturas que incorporam apenas mecanismos paralelos de baixo nível (*pipeline*, múltiplas unidade funcionais e processadores dedicados de entrada e saída), que se tornaram comuns nos computadores modernos e que não fornecem mecanismos para o desenvolvimento de soluções paralelas de alto nível.
- Manter os elementos da difundida classificação de Flynn, baseada em fluxos de instruções e de dados.
- Incluir os processadores vetoriais *pipeline* e outras arquiteturas que eram difíceis de acomodar no esquema de classificação de Flynn.

Sua classificação é representada através da figura 3.19, onde as arquiteturas são basicamente divididas em duas classes: Síncronas e Assíncronas. Na classificação Síncrona, encontramos uma extensão da classe SIMD de Flynn (subdividida em Processadores Matriciais e Memória Associativa), as arquiteturas vetoriais e as arquiteturas sistólicas [DUNCAN, 1990]. Observamos que dentro da classificação Assíncrona, encontramos a arquitetura MIMD sem nenhuma modificação, mantendo as suas subdivisões em Memória Compartilhada e Memória Associativa, e apresenta uma nova classe derivada da arquitetura MIMD para atender os paradigmas introduzidos à arquitetura paralela.

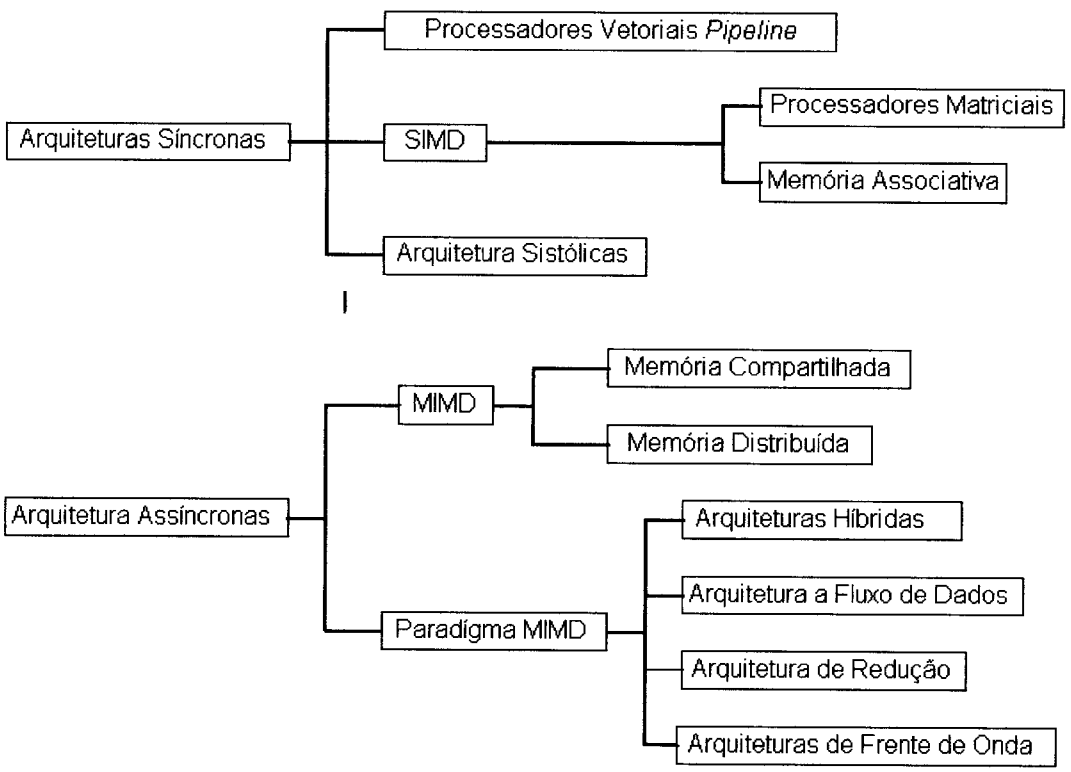


Figura 3.18 – Classificação de Duncan.

3.6.3.1 – ARQUITETURAS SÍNCRONAS

Estas arquiteturas apresentam como característica a coordenação de suas operações concorrentes sincronamente em todos os processadores, através de relógios globais, unidades de controle centralizadas ou controladores de unidades vetoriais [DUNCAN, 1990]. Estas arquiteturas apresentam pouca flexibilidade para expressão de algoritmos paralelos [BLECH, 1994][CORNELL, 1996].

3.6.3.1.1 - PROCESSADORES VETORIAIS *PIPELINE*

Os processadores vetoriais possuem um *hardware* específico para execução de operações vetoriais, implementadas através de instruções vetoriais. Esses processadores são caracterizados por possuírem múltiplas unidades funcionais organizadas que podem operar concorrentemente utilizando *pipelines*, onde são implementadas as instruções vetoriais. Essas arquiteturas foram criadas com a finalidade de prover mecanismos eficientes para o suporte de cálculos pesados com matrizes ou vetores.

A organização básica de um processador vetorial é constituída de: um processador escalar, para a execução do código não vetorizável do programa, uma unidade de processamento vetorial, e um processador de instruções que define quais instruções serão executadas em qual dos processadores [KIRNER, 1991].

Essas arquiteturas efetuam o processamento vetorial paralelo, direcionando os elementos do vetor para o *pipeline* de uma unidade funcional e a saída do *pipeline* de uma unidade funcional para o *pipeline* de outra unidade funcional.

Dentro dessa arquitetura encontramos, por exemplo, o Cray X-MP [ALMASI, 1994] e o Fujitsu VP-200 [DUNCAN, 1990].

3.6.3.1.2 - ARQUITETURAS SIMD

As arquiteturas SIMD apresentam uma unidade central de controle que supervisiona vários processadores, e uma rede de interconexão para comunicação entre processadores ou entre processadores e a memória, que executam a mesma instrução sincronamente em conjuntos de

dados distintos. As arquiteturas dessa classe podem ser organizadas em processadores matriciais e processadores de memória associativa [HWANG, 1984].

- Processadores Matriciais: são projetados para fornecer estruturas para computação sobre matrizes de dados, sendo empregados para fins específicos. Diferencia do modelo de memória associativa, por fornecer acesso à memória via endereço. Um exemplo desse tipo de arquitetura é o Connection Machine [DUNCAN, 1990].
- Memória Associativa: nessa arquitetura o acesso à memória é feito de acordo com o seu conteúdo, permitindo o acesso paralelo à memória, de acordo com certo padrão de dados, empregando uma lógica especial de comparação para acessar simultaneamente vários dados armazenados na memória, e como exemplo dessa arquitetura temos o computador PEPE (Parallel Element Processing Ensemble)[DUNCAN, 1990].

3.6.3.1.3 - ARQUITETURAS SISTÓLICAS

Propostas no início da década de 80, por H.T. Kung da Universidade de Carnegie Mellon, com o objetivo principal de fornecer uma estrutura eficiente para a solução de problemas que necessitem de computação intensiva junto a grande quantidade de operações de entrada/saída. Essas arquiteturas são caracterizadas por possuírem processadores organizados como um *pipeline* multi-dimensional no qual os dados percorrem toda a cadeia de processadores de modo rítmico e sincronizada, da memória para a rede de processadores e retornam novamente à memória. Como exemplo de arquiteturas sistólicas podemos citar os computadores Warp CMU/GE da General Eletric[ALMASI, 1994] e Warp da Universidade Carnigie Mellon [DUNCAN, 1990], nos quais as redes de processadores são programáveis, podendo ser organizados segundo uma distribuição adequada para um determinado algoritmo.

3.6.3.2 - ARQUITETURAS ASSÍNCRONAS

Nessas arquiteturas o controle de *hardware* é descentralizado, onde os processadores trabalham independentemente entre si, sendo formada basicamente pelas arquiteturas MIMD, convencionais ou não [DUNCAN, 1990].

3.6.3.2.1 - ARQUITETURAS MIMD

As arquiteturas MIMD empregam vários processadores que trabalham de modo independente em relação à execução de instruções. Apesar de independentes, os processos que estão sendo executados nos diversos processadores devem cooperar entre si, tornando necessário a comunicação e o sincronismo entre os mesmos.

Como já descrito acima, as arquiteturas MIMD são constituídas por vários processadores que podem executar fluxos de instruções distintos sobre dados locais (Figuras 3.20 e 3.21), suportando aplicações paralelas com alto grau de autonomia entre processos, caracterizando uma granularidade grossa. Nessa arquitetura os processos de software em execução necessitam de sincronização e interação, através da troca de mensagem através de uma rede de interconexão ou acesso a uma memória compartilhada. A sincronização e interação são feitas de modo assíncrono [DUNCAN, 1990], isto é, as operações de sincronização e comunicação entre os processadores, não obedecem a um ritmo ou cadência ditados por um mecanismo de controle de hardware centralizado. Sendo então realizadas conforme a necessidade dos processos e a disponibilidade de elementos de hardware (por exemplo, um canal de comunicação livre entre dois processadores).

O desenvolvimento e utilização de arquitetura MIMD se deve a diversos fatores inter-relacionados. As máquinas MIMD permitem um paralelismo de alto nível (nível de subprogramas e tarefas), que pode ser explorado por algoritmos construídos segundo a técnica "dividir para conquistar" (técnica segundo a qual divide-se, sucessivamente, um problema inicial em problemas menores até que se consiga resolvê-los, e assim compor a resolução do problema inicial). Os computadores MIMD são aplicáveis à computação de propósito geral, ao contrário de computadores SIMD E MISD, que são utilizados em aplicações específicas. Além disso, a relação custo/benefício de computadores MIMD é melhor que a exibida por computadores SIMD por dois motivos: primeiro, o custo de n processadores é menor que o custo de um único processador com poder computacional equivalente ou próximo ao desses n processadores; segundo pode-se expandir, conforme a necessidade e a disponibilidade de recursos, uma arquitetura MIMD adicionando-se a mesma mais processadores [BRONSON,

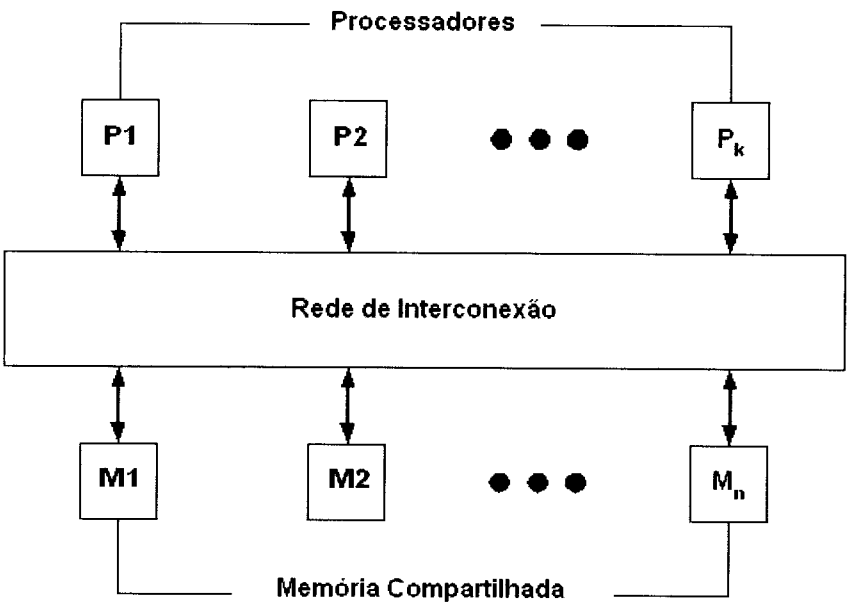
1990]. Isso não é possível em arquiteturas SIMD, a menos que se troque o processador por um mais potente, o que pode ser economicamente inviável [DUNCAN, 1990][HWANG, 1993].

Dependendo da forma em que são organizadas as memórias, a implementação dos métodos de comunicação e sincronismo pode ser com memória compartilhada ou distribuída (Figuras 3.20 e 3.23).

Este tipo de organização oferece, grande flexibilidade para a construção de algoritmos paralelos [BLECH, 1994][CORNELL, 1996].

3.6.2.1.1 - ARQUITETURA MIMD COM MEMÓRIA COMPARTILHADA

Na arquitetura MIMD com memória compartilhada, denominada geralmente por multiprocessador [TANENBAUM, 1992], é composta basicamente por uma rede de interconexão que liga os processadores à memória compartilhada, onde ocorre a comunicação e sincronização entre os processadores (Figura 3.20).



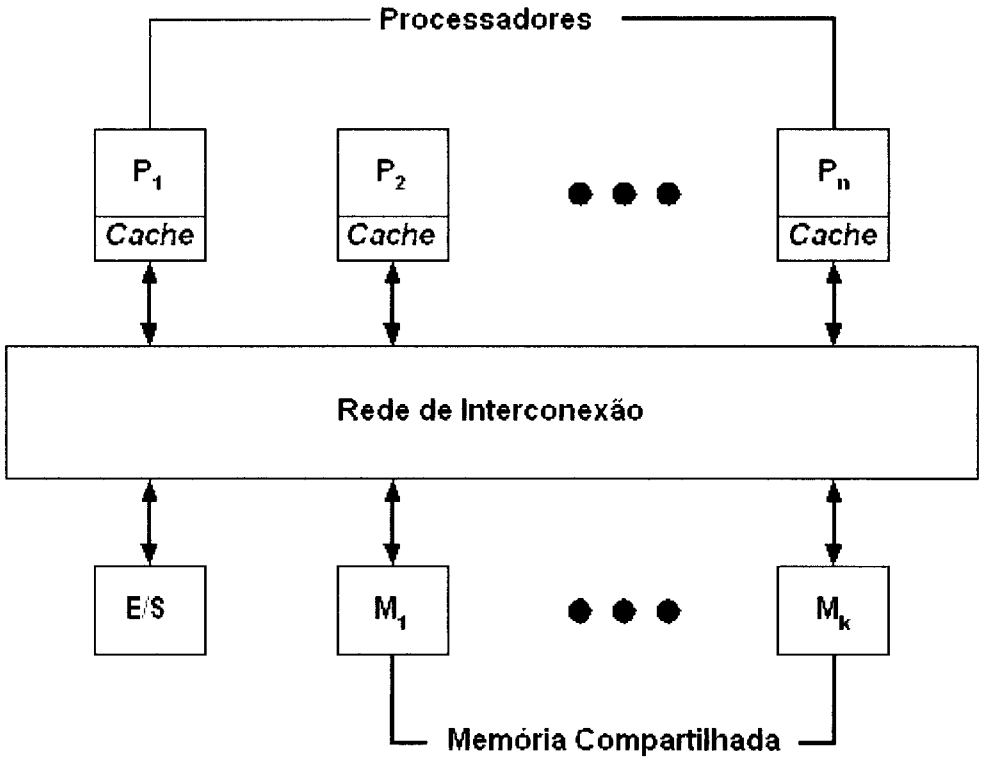
M_i , 1 i n - Módulos da Memória Compartilhada
P_j , 1 j k - Processadores

Figura 3.19 - Arquitetura MIMD com memória compartilhada.

Como exemplo de multiprocessadores temos o Cray Y-MP [ALMASI, 1994] e o Challenge da Silicon Graphics [FOSTER, 1995].

Nos computadores MIMD com memória compartilhada, ou multiprocessador, todos os processadores compartilham uma memória comum, e também periféricos (por exemplo, discos rígidos, etc.) [HWANG, 1993]. A coordenação de eventos paralelos, bem como a sincronização e a comunicação entre os processadores, é feita utilizando variáveis compartilhadas na memória.

A arquitetura básica pode ser também incrementada com o uso de uma memória *cache* em cada processador (Figura 3.21), diminuindo-se o tempo de acesso à memória compartilhada e, consequentemente, o tempo médio de acesso à memória, pois o tempo de acesso a uma memória *cache* local a um processador é muito menor do que o tempo de acesso a uma memória comum e compartilhada. No entanto, a utilização de memória *cache* exige, necessariamente, o uso de mecanismos de hardware e software [FOSTER, 1995] para manter a sua consistência. Sempre que um processador atualizar um dado em sua memória *cache*, o dado associado na memória compartilhada e nas memórias *cache* dos outros processadores deve ser atualizado.



E/S - Dispositivos de Entrada/Saída
M_i , 1 i k - Módulos da Memória Compartilhada
P_i , 1 j n - Processadores

Figura 3.20 - Arquitetura MIMD com memória compartilhada e com memória *cache*.

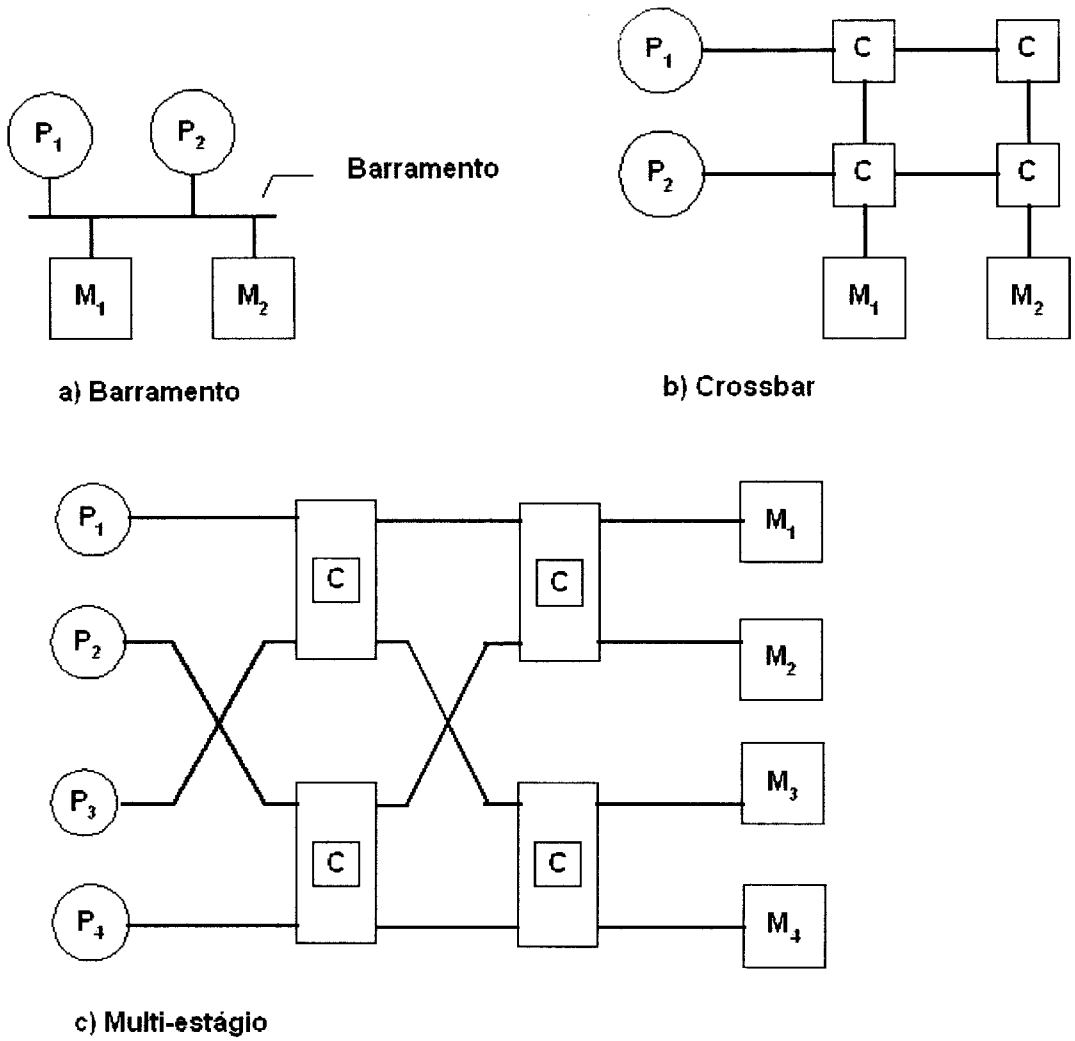
A conexão entre os processadores e a memória compartilhada é feita por uma rede de interconexão, podendo apresentar uma das seguintes topologias:

- Barramento: todos os processadores e a memória compartilhada são conectados a um barramento comum (Figura 3.22 (a)). Para ler uma palavra da memória, por exemplo, o elemento de processamento, após deter o direito de uso do barramento, coloca o endereço da palavra que ele deseja nas linhas de endereço do barramento e informa através das linhas de controle que se trata de uma leitura. A memória responde colocando o valor da palavra nas linhas de dados, que poderá ser lido pelo elemento de processamento que o requisitou. O funcionamento da escrita é similar, embora exija um mecanismo adicional para controlar a coerência da memória *cache*, se ela for utilizada pela arquitetura.

  A principal vantagem do barramento em relação às demais topologias é o custo. A principal limitação é o número de processadores que podem ser utilizados. Com muitos processadores o barramento fica sobrecarregado, tornando-se um gargalo para o sistema.
- Crossbar: nesta topologia utiliza-se uma malha composta por n^2 pontos de intersecção (*crosspoints*), que podem estar abertos ou fechados, para interligar n processadores a n módulos de memória (Figura 3.22(b)). Quando um processador quer ter acesso a um determinado módulo de memória, o ponto de intersecção que os conecta é momentaneamente fechado, permitindo que o acesso ocorra.

  A interconexão por rede *crossbar* permite, em relação ao barramento, mais de um processador comunicando-se com módulos distintos de memória simultaneamente. Entretanto, essa vantagem implica uma desvantagem, na qual o custo da rede *crossbar* é proporcional ao quadrado do número de processadores (n^2 pontos de intersecção), tornando-se proibitivo se o número de processadores for elevado.
- Multi-estágio: nesta topologia os processadores são interligados aos módulos de memória através de múltiplos estágios de chaveamento, compostos por várias chaves de conexão (*switches*) (Figura 3.22(c)). Quando o número N de processadores é uma potência de dois, utiliza-se geralmente $\log_2$ N estágios com N/2 chaves cada um. Para acessar um módulo de memória, especificado por uma certa seqüência de *bits*, um processador submete essa seqüência à rede multi-estágio. Ao atravessar a rede multi-estágio, cada *bit* será responsável por uma decisão de roteamento em cada estágio. O roteamento de dados em cada estágio é feito por uma chave de conexão pertencente ao estágio. Esse roteamento permite a transferência de dados entre os elementos computacionais (chaves de conexão, processadores ou módulos de memória) que estão ligados à chave.

A interligação por multi-estágio constitui um compromisso entre as alternativas de preço/desempenho oferecidas pelas redes *crossbar* e pelo barramento por dois motivos. Primeiro, embora o preço seja superior ao de uma rede por barramento, ele é muito inferior ao de uma rede *crossbar* ($N/2x\log_2N$ têm um preço menor que N^2 chaves de conexão). Do ponto de vista de desempenho, ela é superior ao de barramento por permitir que mais de um processador acesse módulos distintos de memória ao mesmo tempo, embora seja inferior ao de uma rede *crossbar*, pois essa permite um número maior de conexões simultâneas entre os processadores e os módulos de memória.

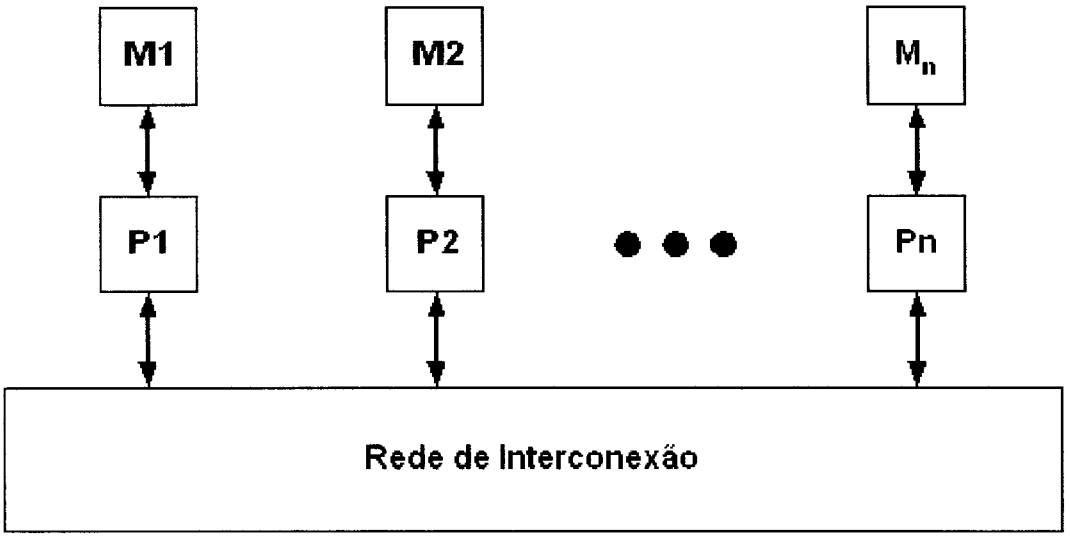


M_i , i l, 1 i 4 - Módulos da Memória Compartilhada
P_j , j l, 1 j 4 - Processadores
C - Elementos de Chaveamento

Figura 3.21 - Redes de interconexão para memória compartilhada.

3.6.2.1.2 – ARQUITETURA MIMD COM MEMÓRIA DISTRIBUÍDA

A arquitetura MIMD com memória distribuída, ou multicomputador [TANENBAUM, 1992], apresenta uma rede de interconexão que interliga os processadores, que possuem memória local (Figura 3.23). Neste caso, por não se ter compartilhamento entre as memórias, os processos se comunicam via troca de mensagens.



M_i , 1 i n - Memória do Processador Pi
P_j , 1 j n - Processadores

Figura 3.22 - Arquitetura MIMD com memória distribuída.

Pode-se citar como exemplo dessa classe os multicomputadores Cray T3D [FOSTER, 1995] da Cray Co. e o SP2 [WOLF, 1995] da IBM.

Dentre as arquiteturas em geral, o modelo MIMD tem se destacado (principalmente o modelo de memória distribuída), devido à sua flexibilidade e por representar uma boa opção para o desenvolvimento de algoritmos paralelos de granularidade média e grossa [BLECH, 1994][CORNELL, 1996][ZALUSK, 1991].

A arquitetura MIMD com memória distribuída, ou multicomputador, múltiplos computadores, denominados freqüentemente de nós de processamento, interconectados por uma rede de passagem de mensagens, denominada rede de interconexão. Cada nó de processamento é um computador autônomo composto por um processador, memória local, e algumas vezes por

IFSC-USP SERVIÇO DE BIBLIOTECA INFORMAÇÃO

discos ou periférico de entrada e saída [HWANG, 1993]. Todas as memórias locais são privativas, significando que somente os processadores locais às mesmas podem acessá-las.

A rede de passagem de mensagens é o meio que os nós utilizam para enviar mensagens a outros nós, possibilitando a comunicação e a sincronização de processos.

As principais topologias para as redes de passagem de mensagem são:

- Barramento: similar à rede de interconexão de mesmo nome para memória compartilhada, porém apresenta uma única diferença onde podemos verificar que somente os nós estão ligados ao barramento (Figura 3.24 (a)). Em função de que a comunicação entre os nós de um multicomputador ser menor que a comunicação entre memórias e processadores em um multiprocessador, a topologia em barramento pode suportar, para uma mesma capacidade de transmissão do barramento, um número maior de elementos de processamento em multicomputadores que em multiprocessadores. Logo, a capacidade de transmissão do barramento pode ser, para o mesmo número de elementos de processamento, muito menor em um multicomputador do que em um multiprocessador.
- Anel: na topologia em anel, cada nó é conectado a dois e somente dois outros nós vizinhos (Figura 3.24 (b)). O caminho que uma mensagem entre dois nós quaisquer deve percorrer é, na média, igual a N/2, onde N é o número de processadores. Pode-se incrementar esse arranjo utilizando-se conexões extras entre nós, que proporcionam uma maior tolerância à falhas e um caminho menor entre dois nós, o que demanda um tempo menor de comunicação entre os nós. A topologia em anel é a mias apropriada para um pequeno número de processadores executando algoritmo que não exijam muita comunicação [DUNCAN, 1990].
- Malha: os nós interconectados por uma rede em malha possuem quatro nós vizinhos, exceto os das extremidades, que têm apenas dois ou três nós vizinhos (Figura 3.24 (c)). A comunicação entre os nós pode ser aumentada utilizando-se ligações entre os nós das extremidades ou barramentos para conectar os nós por linhas ou por colunas. O uso de uma rede em malha é muito apropriado para algoritmos que utilizam matrizes.
- Arvore: na interconexão dos nós segundo uma árvore (Figura 3.24 (d)), todos os nós, menos os nós folhas, possuem pelo menos dois nós vizinhos. O caminho de uma mensagem entre dois nós quaisquer pode ser encurtado utilizando-se conexões auxiliares entre os nós, como, por exemplo, conectando-se todos os nós pertencentes a um mesmo nível da árvore. Esta topologia em árvore é recomendada para algoritmos de busca e ordenação, "dividir para conquistar" e paradigmas de redução e de fluxo de dados [DUNCAN, 1990].

- Hipercubo: nessa topologia os $N = 2^n$ nós do multicomputador estão organizados segundo um cubo de dimensão n, onde cada nó possui $n = \log_2 N$ ligações com nós adjacentes (Figura 3.24(e)). A topologia em hipercubo é de grande utilização em multicomputadores que devem executar aplicações científicas tridimensionais ou de dimensões espaciais superiores.
- Reconfigurável: este tipo de interconexão, apresenta a possibilidade de poder ser modificada através de chaves (*switches*) de programação, de forma a apresentar uma topologia adequada aos padrões de comunicação da aplicação do usuário.

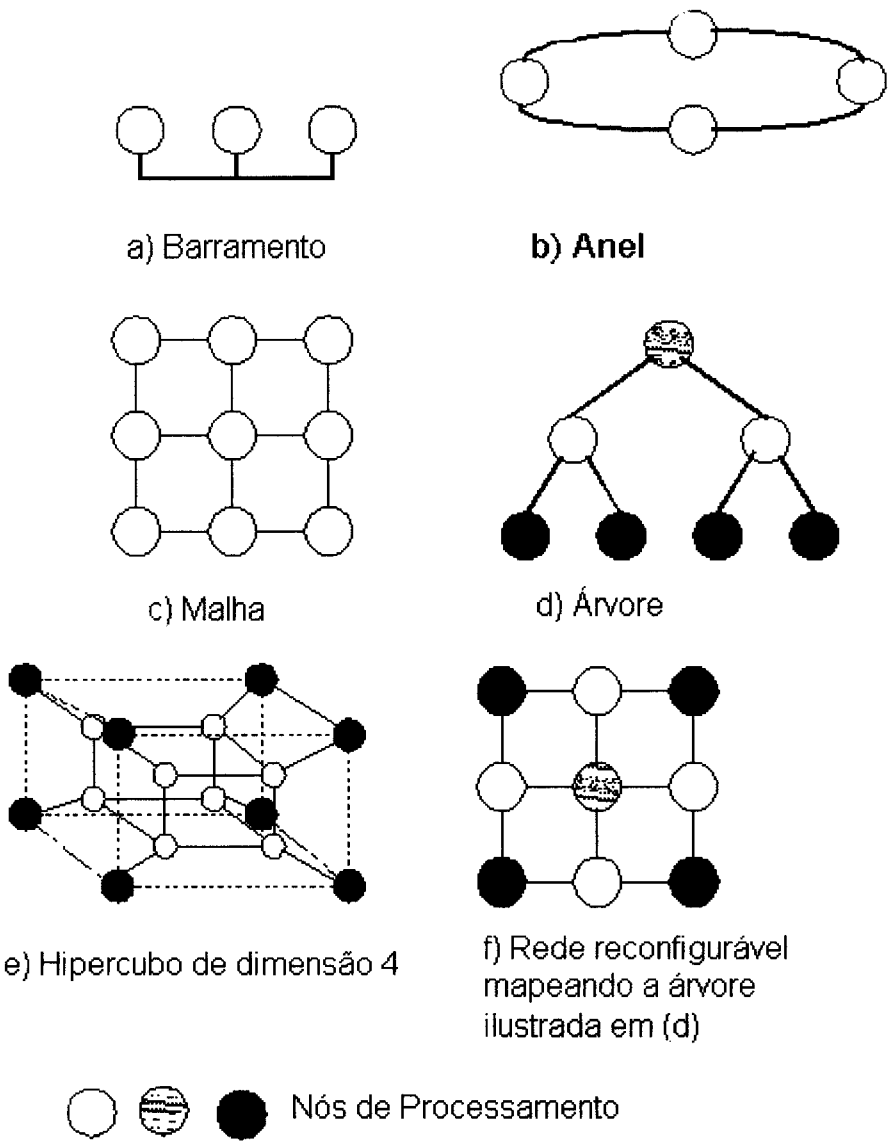


Figura 3.23 - Redes de interconexão para memória distribuída

3.6.3.2.2 - PARADIGMA MIMD

Dentro dessa classe encontramos as arquiteturas assíncronas que apresentam a característica de multiplicidade de fluxo de dados e instruções das arquiteturas MIMD, mas princípios de organização diferentes da mesma. Estas características próprias de cada arquitetura dificultam a sua classificação como puramente MIMD, tornando-as assim, um paradigma dessa arquitetura. Podemos subdividi-las em:

- Arquitetura MIMD/SIMD: conhecida também por MSIMD, onde os grupos de processadores são organizados segundo uma arquitetura SIMD, porem controlados por outros processadores organizados segundo uma arquitetura MIMD. Como exemplo temos o computador Texas Array Re-configurable Computer, ou TRAC [DUNCAN, 1990].
- Arquitetura Fluxo de Dados: criadas para explorar o paralelismo máximo através de um programa, controlando a execução das instruções de acordo com a dependência dos dados ao quais se aplicam estas instruções, gerando computadores baseados em fluxo de dados. Controle este de forma não centralizada na execução de um programa, com os dados fluindo de instrução a instrução, de maneira que se consiga paralelismo em alta escala. Nessa classe temos o computador Manchester Data Flow Computer [DUNCAN, 1990].
- Arquitetura de Redução: essas arquiteturas baseiam-se no conceito de redução, onde partes do código fonte original é reduzida a seus resultados em tempo de execução, conhecida como arquitetura dirigida à demanda [KIRNER, 1991]. As instruções são ativadas para serem executadas quando os seus resultados são necessários como operandos por outra instrução já ativada para execução. Temos como exemplo deste caso os computadores Utah Applicative Multiprocessing System e o Newcastle Reduction Machine [DUNCAN, 1990].
- Arquiteturas Frente de Onda: apresentam uma organização similar à das arquiteturas sistólicas, mas não tem um relógio global que cadencia a execução das instruções. Quando um processador executa uma instrução e está pronto para passar os dados para o seu sucessor, ele informa ao sucessor. Após o sucessor indicar que está apto para recebê-los, o processador envia-os e recebe uma confirmação de recebimento dos dados pelo sucessor. Nessa classe temos como exemplo o computador Wavefront Array da Universidade Johns Hopkins [DUNCAN, 1990].

3.6.4 - A ARQUITETURA DO SPP3

A arquitetura do SPP3 pode ser classificada, segundo a taxonomia de Duncan [DUNCAN, 1990], como assíncrona, MIMD e com memória distribuída. Sua principal característica é que esta máquina foi construída empregando componentes de computadores pessoais, acompanhando a tendência da maioria das arquiteturas MIMD com memória distribuída existentes no mercado [CRAY, 1995][IBM, 1993]. Na figura 3.25 pode ser visto uma visão geral da arquitetura do SPP3.

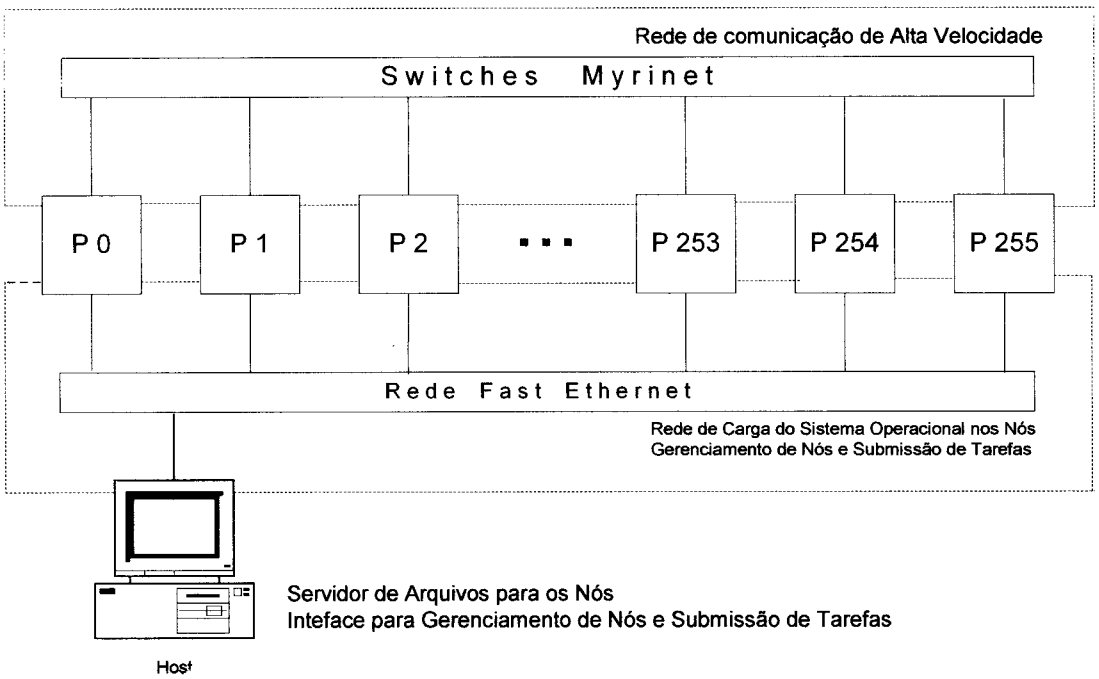


Figura 3.24 - Arquitetura do SPP3

O SPP3 suporta até 256 nós de processamento, ou simplesmente nós, distribuídos em módulos de oito (8) nós de processamento cada. Cada nó é uma placa mãe (*motherboard*) de computador pessoal contendo um ou mais processadores, memória e interfaces de comunicação. Para um computador SPP3 com N nós ($0 < N < 256$), os nós são nomeados como P*i*, onde *i* pode variar de 0 a N – 1. A interligação dos nós é feita por dois tipos de rede de comunicação. Uma rede Ethernet (rede de interconexão em barramento) proporciona o meio de comunicação necessário para cada nó carregar de forma remota o sistema operacional compatível com Unix, Linux [WELSH, 1994]. Essa rede é utilizada também para o controle do sistema e para assumir as tarefas (processos), sendo denominada rede de gerenciamento. Uma rede de comunicação de

alta velocidade formada por um conjunto de *switches* e adaptadores Myrinet [BODEN, 1995] provê a comunicação entre os nós de processamento durante a execução de aplicações paralelas. A submissão de tarefas componentes de aplicações paralelas e o gerenciamento de todo o sistema são realizados através de um computador pessoal, o *host* (computador principal) (Figura 3.25).

3.6.4.1 - ORGANIZAÇÃO E IMPLEMENTAÇÃO DO SPP3

Como já citado, a arquitetura do SPP3 acompanha a tendência da maioria das arquiteturas MIMD com memória distribuída disponíveis no mercado, onde os nós são compostos por placas com até quatro microprocessadores e memória compartilhada, sendo que atualmente contêm apenas um processador. Essas placas estão ligadas por uma rede de comunicação de alto desempenho. A utilização de mais de um processador por nó faz da arquitetura do SPP3 uma arquitetura MIMD híbrida, utilizando ao mesmo tempo os paradigmas de programação de memória compartilhada e memória distribuída. Em sua arquitetura é possível utilizar placas de processadores Pentium Intel, DEC Alpha ou Power PC [TRINDADE, 1995], onde o único requisito para o processador é que seja disponível para ser usado com o sistema operacional Linux. A quantidade de memória depende da placa utilizada.

A arquitetura do SPP3 é escalável podendo atingir até 256 nós, conectado ao *host* através de duas redes de comunicação. A primeira é uma rede Ethernet encarregada de efetuar controle e gerenciamento do sistema, e a segunda uma rede de alta velocidade Myrinet para comunicação de dados.

Os nós processadores são organizados em módulos. Originalmente, cada módulo do SPP3 pode conter até 16 nós. Um único módulo pode conter até 32 processadores, caso a máquina paralela seja constituída por apenas um módulo.

O *host* tem a função de servidor de arquivos e servidor de boot remoto. Opcionalmente, cada nó pode ter seu disco local, o que torna mais ágil o armazenamento e acesso de dados individuais dos processos de cada nó. A área de memória *swap*, nesse último caso, pode ser local em contraposição ao caso de nós sem disco cuja área de *swap* nó reside no *host*.

Quando são usados nós com mais de um processador, o sistema operacional fica encarregado da distribuição de processos. É possível a utilização de programas paralelos escritos de forma a aproveitar a maior velocidade disponível para comunicação entre processos que estão sendo executados na mesma placa (utilizando a memória compartilhada da placa), ao

passo que processos em placas diferentes devem efetuar a comunicação por passagem de mensagens através da rede de comunicação de alto desempenho.

O sistema operacional utilizado é compatível com o UNIX, o Linux. Sua utilização por vários sistemas semelhantes ao SPP3 provocou um aumento na disponibilidade de *software* para processamento paralelo para essa classe de sistemas. Uma indicação nesse sentido é a publicação do documento *Linux Parallel HOWTO* [LINUX, 1998] que relata os principais programas para processamento paralelo utilizando redes de estações de trabalho Linux. Esses programas podem ser utilizado no SPP3 embora nem todos possam fazer uso apropriado dos recurso da rede de comunicação de alto desempenho disponível.

A figura 3.26 ilustra a arquitetura do SPP3. Cada módulo contém 16 nós de processamento, facilitando-se a implementação física do sistema e mantendo as mesmas características da implementação inicial do SPP3. A rede comunicação de alto desempenho é uma rede Myrinet.

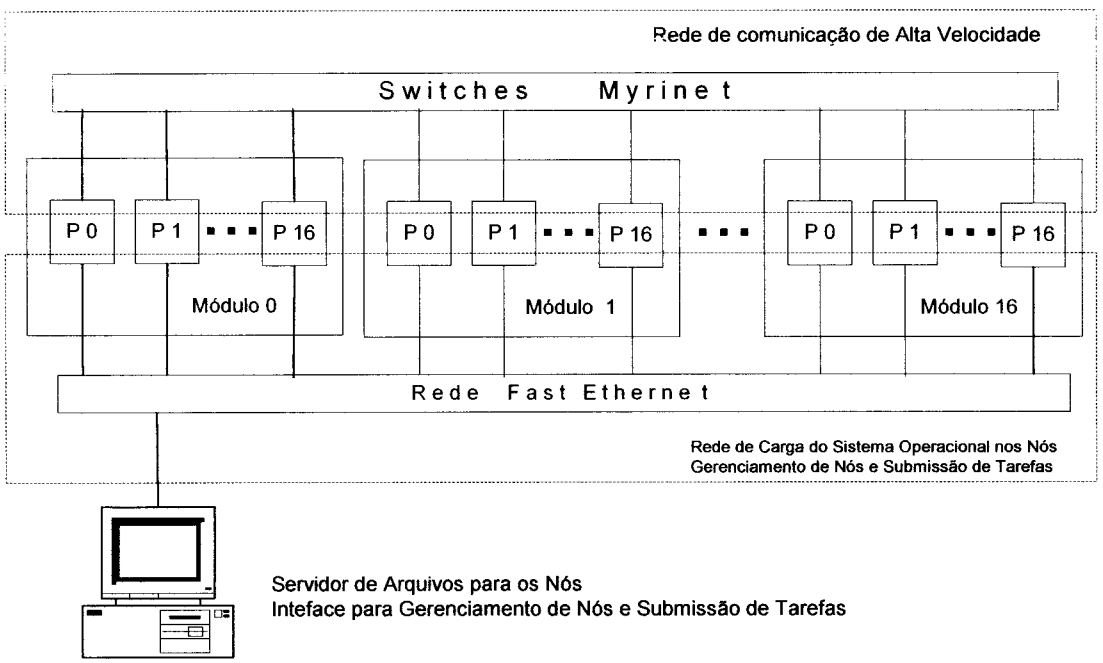


Figura 3.25 - Arquitetura do SPP3

CAPÍTULO 4

4 - PROGRAMAÇÃO PARALELA

A programação paralela requer, além das construções normalmente disponíveis em uma linguagem de programação seqüencial, outras para definir a execução paralela de processos, como iniciar e interromper sua execução e coordenar e especificar a interação entre eles [ALMASI, 1994]. Essas construções podem ser agrupadas segundo seu princípio de funcionamento e adequação a um determinado nível de paralelismo. No item 4.2 são apresentadas as construções básicas para a finalidade acima.

Com relação à cooperação entre os diversos processos componentes de uma tarefa, podemos recorrer a construções que possibilitem a comunicação e a sincronização de suas atividades, conforme pode ser visto no item 4.3.

Segundo Almasi [ALMASI, 1994], para a execução de programas paralelos, deve haver meios de:

- Definir um conjunto de tarefas a serem executadas paralelamente;
- Ativar e encerrar a execução dessas tarefas;
- Coordenar e especificar a interação entre essas tarefas.

A definição da organização das tarefas paralelas é de extrema importância, pois o ganho de desempenho adquirido da paralelização depende fortemente da melhor configuração das tarefas a serem executadas concorrentemente. Alguns dos aspectos que devem ser considerados no desenvolvimento de algoritmos paralelos são discutidos no item 4.1.

Após definir o algoritmo, é necessário um conjunto de ferramentas para que o programador possa representar a concorrência, definindo quais partes do código serão executadas seqüencialmente e quais serão paralelas. As construções para definir, ativar e encerrar a execução de tarefas concorrentes são discutidas no item 4.2.

Temos ainda, processos que cooperam para a resolução de determinado problema devem comunicar-se e sincronizar-se, afim de que haja interação entre eles. Vimos que, para se implementar a comunicação entre processos depende da arquitetura em que se executa a aplicação. No caso de memória compartilhada, utilizam-se variáveis compartilhadas, e no caso de memória distribuída, é utilizada a troca de mensagens. A comunicação e sincronização entre processos são apresentadas no item 4.3.

4.1- ALGORITMO PARALELO

Há, pelo menos, três maneiras de se construir um algoritmo paralelo [QUINN, 1987], que são:

- Detectar e explorar algum paralelismo inerente a um algoritmo seqüencial existente;
- Criar um algoritmo paralelo novo;
- Adaptar outro algoritmo paralelo que resolva problema similar.

Existem três aspectos importantes que se deve considerar quando se projeta um algoritmo paralelo. Em primeiro lugar devemos nos preocupar com o processo de escolha da abordagem a ser seguida, entre as três citadas acima, a qual deve ser feita de maneira cuidadosa, para que se consiga a melhor eficiência, levando em conta o tempo de escrita do algoritmo e o desempenho obtido.

A seguir, deve-se pesar o custo da comunicação entre os processos em relação ao tempo de execução efetiva, visto que operações de comunicação geram uma sobrecarga que pode degradar o desempenho, tornando o algoritmo menos eficiente que o seqüencial. E finalmente, devemos levar em consideração a arquitetura na qual será executado o algoritmo, pois a sua eficiência pode variar de maneira significativa de acordo com o tipo de arquitetura.

4.2 - ATIVAÇÃO E TÉRMINO DE PROCESSOS PARALELOS

Encontramos na literatura diferentes construções para ativação e término de processos paralelos, apresentando características que diferem entre si pela adequação de uso, estruturação e flexibilidade [ALMASI, 1994][SNOW, 1992][QUINN, 1987][KIRNER, 1989][ANDREW, 1983].

A seguir são apresentados modelos representativos dessas construções.

Corotinas: as corotinas se assemelham às subrotinas de processos comuns, apresentando apenas como diferença no modo de transferência de controle, que é não hierárquico. No caso de uma subrotina comum, ativada através de uma chamada *call subrotina*, ao executar um comando *return*, retorna o controle ao módulo de programa que a ativou e termina a sua execução, sendo que toda vez que é ativada, será executada desde o início. No entanto as corotinas transferem controle entre si de maneira livre, através do comando *resume corotina*, e sempre que são ativadas, executam a partir do ponto onde foi executada a última chamada *resume* (Figura 4.1).

As corotinas compartilham um único elemento de processamento (programas concorrentes), estando apenas uma ativa a cada instante, implementando um processo, e na seqüência as demais serão executadas intercaladamente, sendo desta forma adequada para a construção de programas paralelos que compartilhem um único elemento de processamento.

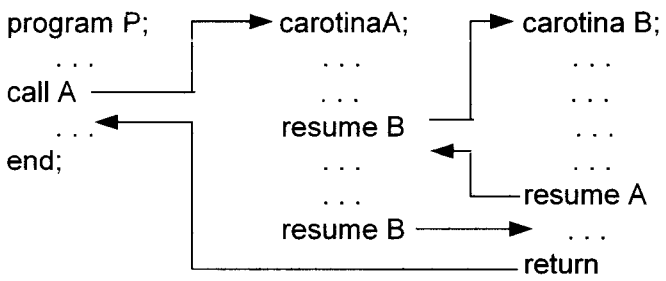


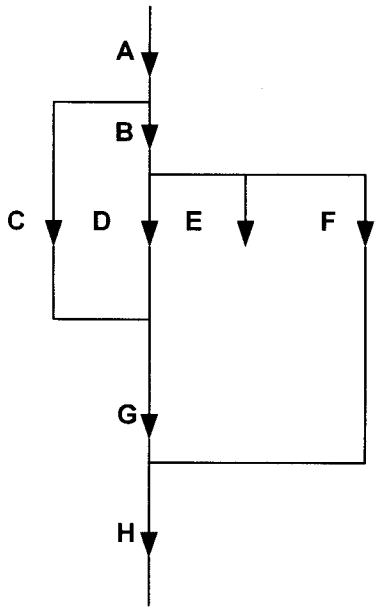
Figura 4.1 - Uso de corotinas

Fork / Join: utiliza-se o comando *fork* para iniciar a execução de um determinado conjunto de instruções (processo filho), paralelamente com outros processos em execução, incluindo o processo que o executou (processo pai). O comando *join* é utilizado para a sincronização dos processos ativados. Um exemplo de sintaxe para os comandos *fork / join* (Figura 4.2) são:

- ***Fork end***: executa o processo a partir do endereço *end*, paralelamente ao processo que executa a chamada *fork*, e demais processos em execução.

- ***Join num, end1, end2***: decrementa a variável *num*, que contém o número de processos que devem sincronizar. Se *num* = 0, desvia o fluxo de execução para o endereço *end1*. Caso contrário, devia para *end2* (geralmente é um comando *quit*, que termina a execução).

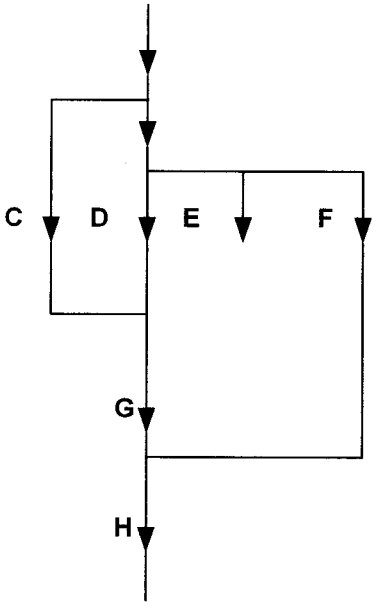
A utilização da construção *fork / join* é uma forma poderosa e flexível de se especificar um processamento paralelo. A utilização dessa construção deve ser feita de modo disciplinado e sistemático, pois a organização do código fonte em que ela aparece é desestruturado e suscetível a erro, pois a sua sintaxe permite saltos para qualquer trecho do código. Para se proporcionar uma maior estruturação, mas com perda de flexibilidade, foram propostos os comandos *cobegin / coend* e *doall*.




```
A
m=2
fork c
B
n=3
fork e
fork f
D
join m.g: quit
g:G
  join n.h: quit
h:H
  quit
c:C
  join m.g: quit
e:E
  join n.h: quit
f:F
  join n.h: quit
```

Figura 4.2 - Exemplo de utilização *fork / join*

Cobegin / Coend: esta construção que também é chamada de *Parbegin / Parend* permite a execução paralela das instruções ou processos delimitados pelos comandos *Cobegin* e *Coend* (Figura 4.3), permitindo uma forma estruturada para ativação de um conjunto de instruções que devem ser executadas paralelamente.




```
A
cobegin
 C
 begin
   B
  cobegin
    D
    E
    F
  coend
  G
 end
coend
H
```

Figura 4.3 - Exemplo de utilização *Cobegin / Coend*

Doall: este comando funciona como um comando *cobegin / coend*, onde as instruções executadas em paralelo são as diversas instâncias de um bloco de comando dentro de um conjunto de repetições (*loop*) (Figura 4.4). Temos também alguns comandos de função semelhante que são: *forall, pardo* e *doacross*.

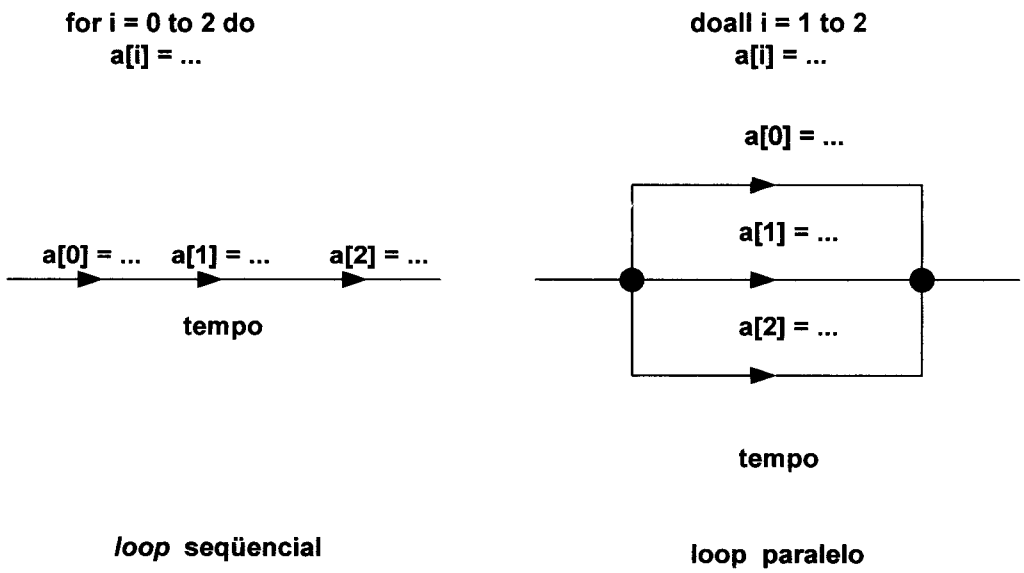


Figura 4.4 - Exemplo da utilização de *doall*

4.3 - COMUNICAÇÃO E SINCRONISMO

A comunicação é necessária para que haja troca de informações na resolução de determinadas aplicações. Ocorrendo comunicação, deve existir operação de sincronização, para fornecer controle de acesso e controle de seqüência.

O controle de seqüência que também é chamado de sincronização condicional ou sincronização de atividades é utilizado para que se determine a ordem na qual os processos (ou partes deles) devem ser executados. A sincronização possibilita a coordenação das atividades dos processos e o controle dos acessos a recursos compartilhados pelos mesmos, para se manter a consistência desses recursos [ALMASI, 1994][SNOW, 1992][KIRNER, 1989][QUINN, 1987][ANDREW, 1983].

A comunicação e a sincronização podem ser feitas basicamente por mecanismos de passagem de mensagens ou por variáveis compartilhadas, havendo uma maior ou menor adequação de um ou outro mecanismo dependendo do arranjo da memória (compartilhada ou distribuída) do computador em que os processos estão sendo executados.

4.3.1 - COMUNICAÇÃO E SINCRONISMO EM MEMÓRIA COMPARTILHADA

Através da utilização de variáveis compartilhadas pelos diversos processos paralelos que cooperam na execução de uma tarefa, se obtém a comunicação e sincronização (controle de acesso e seqüência) entre os processos em computadores com memória compartilhada.

Controle de acesso é geralmente implementado através de exclusão mútua de seções críticas. Essas seções ou regiões críticas são conjuntos de código que devem ser executados de maneira mutuamente exclusiva por um processo sem que outros processos o executem concorrentemente.

Na implementação de exclusão mútua e controle de seqüência utilizando-se de variáveis compartilhadas, podem-se utilizar vários métodos, dentre os quais destacam-se os seguintes:

- ***Busy-waiting***: utilizando-se uma variável compartilhada cujo valor pode ser manipulado (modificado e testado / *test-and-set*) através de uma primitiva indivisível, cria-se um modo simples para a sincronização de processos concorrentes. Para que um processo tenha permissão para entrar nessa região crítica ele deve executar esta primitiva continuamente, gastando-se ciclos de CPU, enquanto se está testando a variável, caracterizando este método de sincronização como um exemplo de *busy-waiting*, ou espera ocupada.
  No *busy-waiting* existem as seguintes desvantagens; gasto supérfluo de tempo de CPU e dificuldade de entendimento, depuração e validação de programas, visto que a primitiva de manipulação é implementada em baixo nível.

- **Semáforos**: é uma variável compartilhada inteira e não negativa utilizada para controlar o acesso a uma região crítica. São utilizadas duas operações atômicas (indivisíveis) **p** e **v**, conhecidas também por ***down*** e ***up***, para controle do semáforo. Um semáforo*s* pode ser implementado utilizando a seguinte semântica:

```
p(s): se (s=0)
      então{
         bloqueia-se o processo até que s > 0
         s = s – 1
      }
      senão  s = s – 1
      v(s): s = s + 1
```

As utilizações de semáforos oferecem um meio de sincronização de nível mais elevado do que *busy-waiting*, evitando o desperdício de CPU.

- **Monitores:** um monitor é um conjunto de procedimentos, variáveis e estruturas de dados agrupados em um módulo de programação. As variáveis e estruturas de dados representam o estado de algum recurso compartilhado enquanto que os procedimentos implementam operações sobre esses recursos. Essas variáveis e estruturas de dados podem ser acessadas somente pelos procedimentos internos a cada monitor, e a execução desses procedimentos é feita de maneira mutuamente exclusiva, e mais estruturada, propiciando menos erros que os semáforos, que são desestruturados.

4.3.2 - COMUNICAÇÃO E SINCRONIZAÇÃO EM MEMÓRIA DISTRIBUÍDA

A comunicação e a sincronização em arquiteturas de memória distribuída devem ser implementadas através de passagem de mensagens entre processos [ALMASI, 1994][SNOW, 1992][KIRNER, 1989][QUINN, 1987][ANDREW, 1983].

Uma operação de comunicação através de passagem de mensagens, de maneira genérica, é realizada pela utilização das primitivas *send / receive* (envia / recebe), cujas sintaxes, podem ser:

- ***Send*** mensagem **to** processo_destino (*envia* mensagem *para* processo_destino).
- ***Receive*** mensagem ***from*** processo_destino (*Receba* mensagem *de* processo_*fonte*).

A forma mais simples de comunicação através de passagem de mensagens é a comunicação ponto-a-ponto. Uma mensagem é transmitida do processo fonte para um processo receptor.

A operação de transferência de mensagens pode ser realizada de modo síncrono ou assíncrono. No modo de operação de transferência síncrona (Figura 4.5), o processo que transmite a mensagem (transmissor) é bloqueado até que receba uma confirmação de recebimento da mensagem pelo processo receptor. E no caso de transferência assíncrona (Figura 4.6), tem-se a preocupação apenas se a mensagem foi enviada, o processo transmissor envia a mensagem (que deve ser armazenada em um *buffer*) e continua a executar outras operações, não se importando com o recebimento dos dados (mensagem).

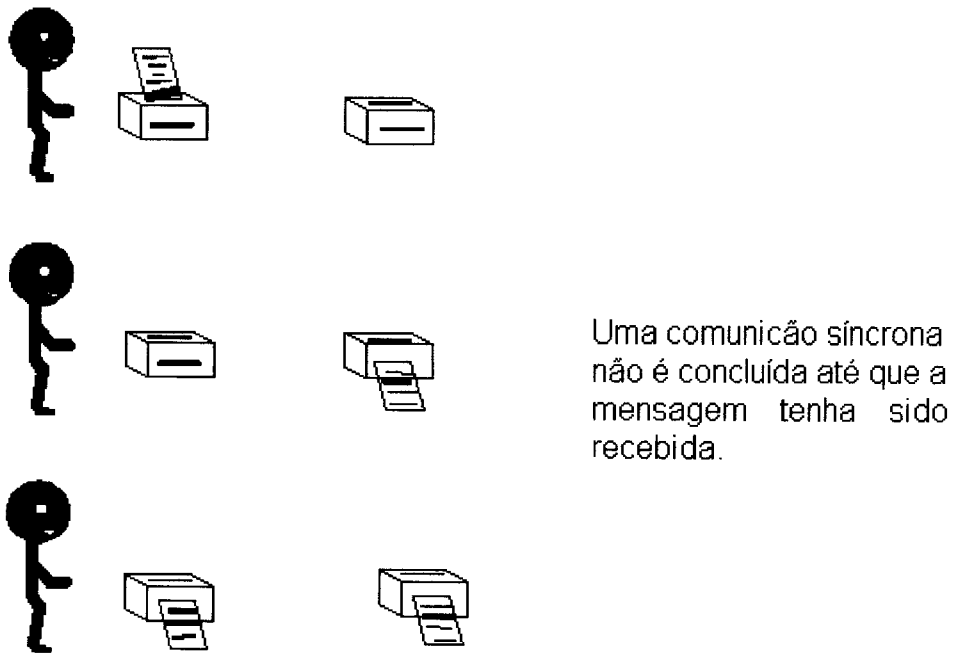


Figura 4.5 - Funcionamento de uma comunicação síncrona

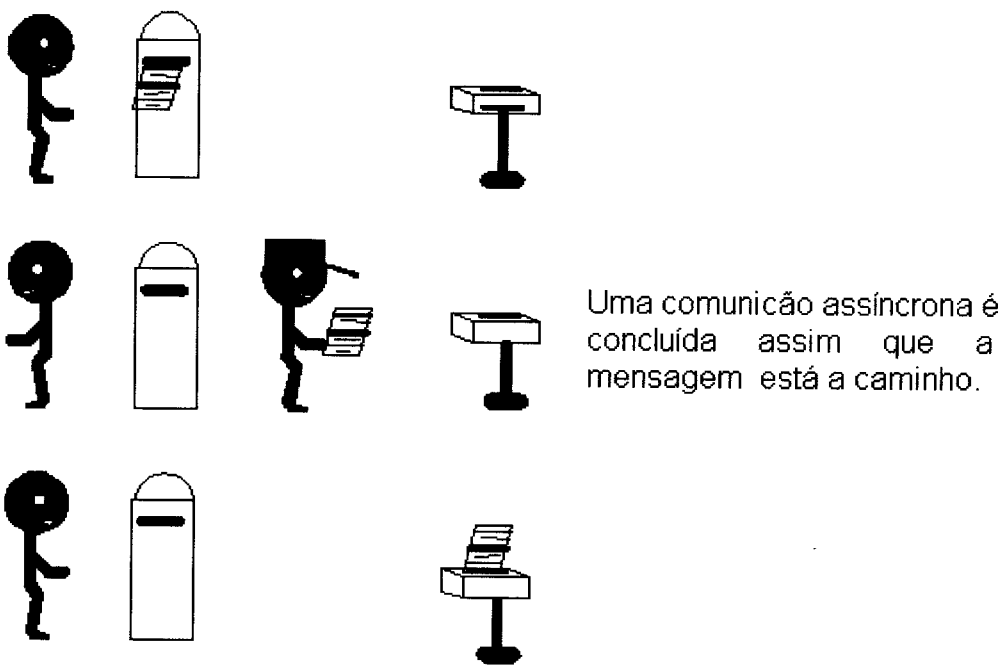


Figura 4.6 - Funcionamento de uma comunicação assíncrona.

O envio de uma carta registrada ou uma mensagem via fax é uma operação síncrona. O remetente recebe a confirmação do recebimento ou se a mensagem foi transmitida. Enquanto que o envio de uma carta simples ou de um cartão postal é uma mensagem assíncrona, pois o remetente só sabe que a correspondência foi colocada dentro da caixa postal, mas não tem informação se esta chegou ao destino.

Temos também que ressaltar como importante a operação bloqueante e a operação não bloqueante. Operações bloqueantes somente retornam da chamada da subrotina associada (*send* e *receive*) quando a operação for completada (Figura 4.7). Um *send* bloqueante pode ser completado pela simples cópia dos dados a serem transmitidos em uma área (*buffer*) de transmissão para a transmissão posterior pelo sistema operacional. Um *receive* bloqueante é completado assim que ele recebe os dados transmitidos por um processo.

Operações não bloqueantes retornam imediatamente e permitem que o processo continue a executar outras operações. Depois de um certo tempo o processo pode verificar se a operação não bloqueante foi completada.

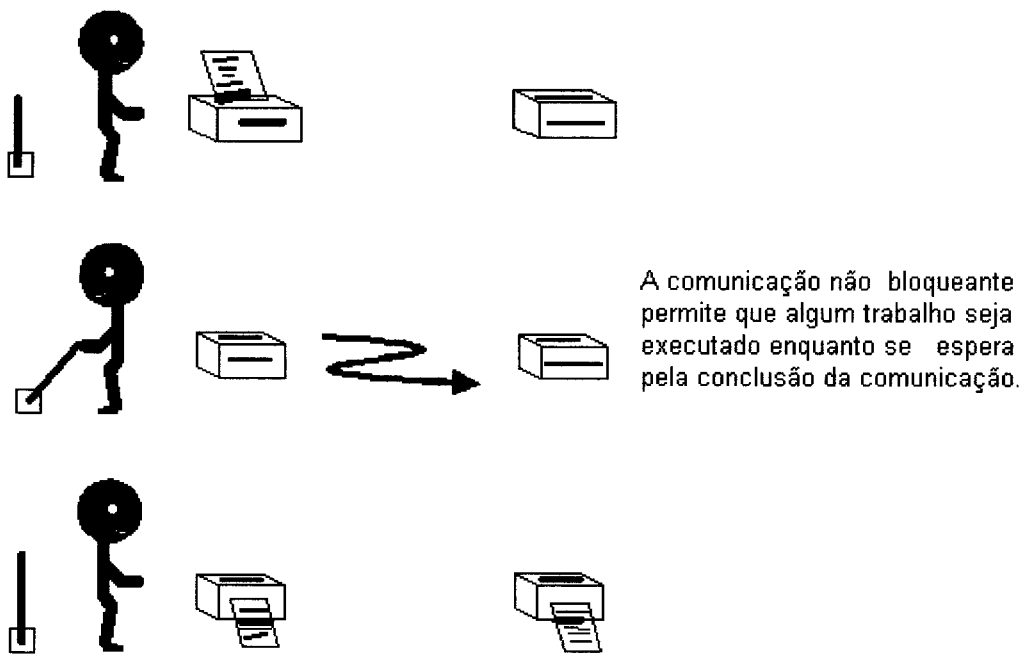


Figura 4.7 - Funcionamento da comunicação não bloqueante

Como exemplo de comunicação não bloqueante temos os aparelhos de fax comum. O fax mantém-se ocupado até que a mensagem tenha sido enviada. Sendo que alguns aparelhos mais modernos de fax possuem uma memória, que permite carregar um documento na memória e, se o aparelho remoto estiver ocupado, realiza outra atividade até que se possa enviar o documento com sucesso, constituindo-se em uma operação não bloqueante.

IESC-USP SERVIÇO DE BIBLIOTECA INFORMAÇÃO

No entanto podemos exemplificar que o recebimento de uma mensagem via fax é também uma operação não bloqueante, quando, deixamos o aparelho de fax em *standby* (estado de espera), para que alguma mensagem possa ser recebida, e observando-se periodicamente o aparelho, pode-se determinar se uma mensagem chegou.

Com as primitivas básicas *send / receive* podemos gerar mecanismos mais poderosos de comunicação [ALMASI, 1994], destacando-se as seguintes:

- **Comunicação Ponto-a-Ponto**: caracterizada pelo uso de uma operação *send / receive* síncrona, com processos sincronizados (Figura 4.8), com uma comunicação unidirecional.

- ***Rendezvouz***: com a utilização de dois conjuntos de operações *send / receive* bloqueantes, organizadas de forma a fornecer uma comunicação bidirecional, permite que um processo comande a execução de um trecho de programa em outro processo (Figura 4.9).

- **RPC** (***Remote Procedure Call*** – Chamada Remota de Procedimento): utiliza uma sintaxe semelhante à chamada de procedimentos locais, para chamar procedimentos remotos a um processo, sendo que o processo requisitador do serviço é bloqueado até que se obtenha os resultados do procedimento desejado (Figura 4.10).

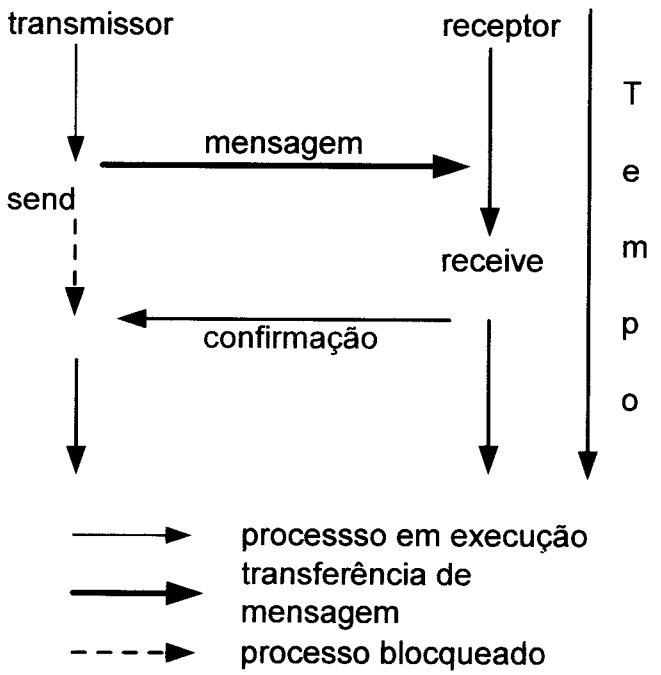


Figura 4.8 - Mecanismo de comunicação Ponto-a-Ponto

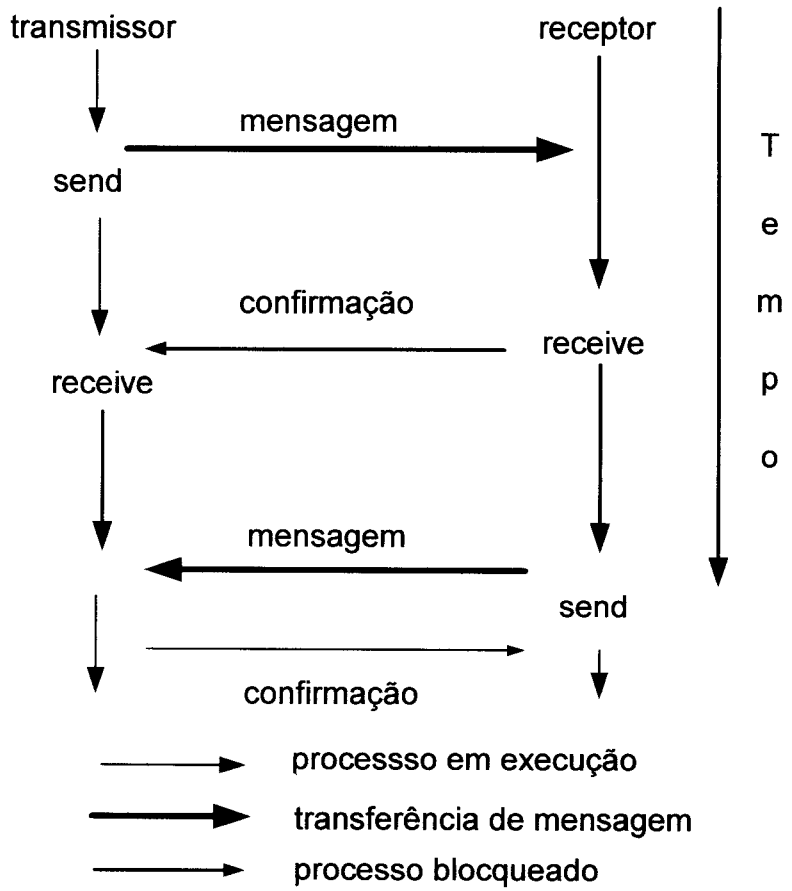


Figura 4.9 - Mecanismo de comunicação *rendezvouz.*

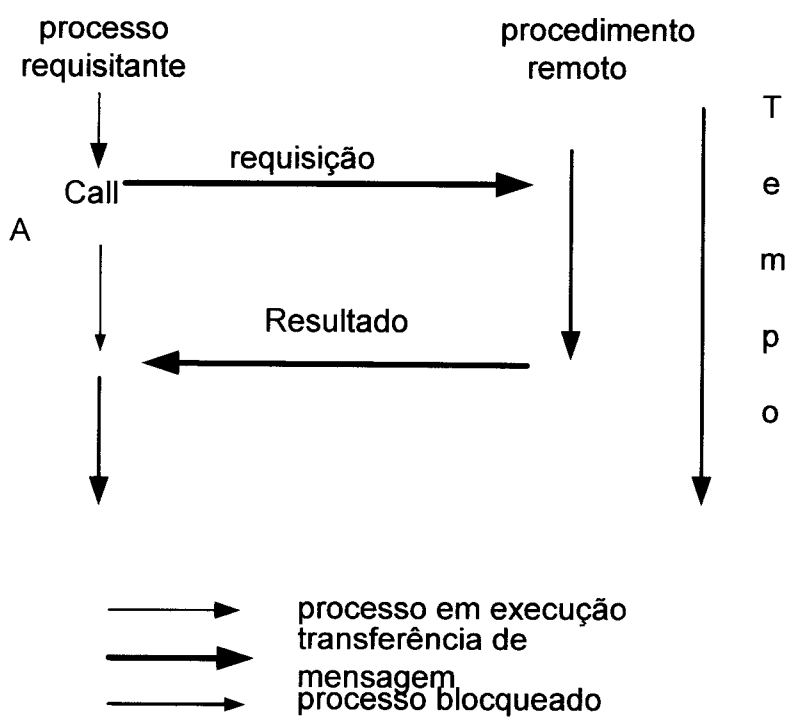


Figura 4.10 - Mecanismo de comunicação RPC

4.4 - FERRAMENTAS PARA DESENVOLVIMENTO DE APLICAÇÕES PARALELAS.

O desenvolvimento de aplicações paralelas é uma tarefa complexa. Devido à complexidade da programação paralela aliada à busca por maior desempenho na execução das aplicações, exige-se o uso de diferentes tipos de ferramentas. Essas ferramentas são destinadas às seguintes atividades: programação, auxílio à paralelização , depuração e análise de desempenho.

4.4.1 - FERRAMENTAS DE PROGRAMAÇÃO

A implementação de um programa paralelo requer, como já visto, mecanismos que permitam a definição, ativação, interrupção e coordenação da execução de tarefas paralelas, além do particionamento e distribuição de dados. Esses mecanismos diferem quanto a granulosidade do paralelismo explorado e à arquitetura utilizada. As ferramentas que oferecem esses mecanismos apresentam diferentes relações entre o esforço do programador e o desempenho obtido pelo programa implementado, podendo ser classificadas em: compiladores e pré-processadores paralelizadores, extensões paralelas para linguagens imperativas e linguagens paralelas [ALMASI, 1994].

Os compiladores e pré-processadores paralelizadores (Paráfrase [ALMASI, 1994] para código Fortran, por exemplo) produzem um programa paralelo a partir de um programa serial. Na sua utilização esse tipo de ferramenta requer um esforço mínimo do programador, pois a conversão do código fonte para sua versão paralela é automática. Entretanto, tais compiladores e pré-processadores possuem duas limitações: o desempenho na execução do código obtido com o seu uso é modesto e eles nem sempre estão disponíveis, principalmente para arquiteturas de memória distribuída.

As extensões paralelas para as linguagens imperativas são bibliotecas que complementam as linguagens convencionais com os mecanismos para programação paralela. O uso dessas extensões, se comparado ao dos compiladores, fornece um desempenho maior, exigindo um esforço adicional do programador. Esforço este que se concentra basicamente no aprendizado das rotinas componentes da extensão paralela da linguagem serial. Pode-se citar como exemplo dessas extensões o Pascal Concorrente, para ambientes de memória compartilhada, o MPI (*Message Passing Interface)* [MPI, 1995] e o PVM (*Parallel Virtual Machine*) [GEIST, 1994]. O MPI e o PVM são ambientes para programação por passagem de mensagens.

As linguagens paralelas podem ser classificadas em dois paradigmas de programação. No paradigma imperativo as linguagens destinam-se a um determinado modelo computacional e possuem comandos relacionados com os mecanismos de programação paralela (por exemplo, linguagem Occam para arquiteturas que utilizam processadores Transputer). No paradigma não imperativo as linguagens não são destinadas a um modelo computacional específico e caracterizam-se por expressar *o que* fazer e não *como* fazer (por exemplo, linguagem Haskell). As linguagens paralelas exigem um grande esforço do programador, se comparado ao exigido pelos outros dois tipos de ferramentas descritos anteriormente, pois ele deverá aprender uma linguagem ou um paradigma de programação completamente nova. A respeito de tal esforço, o uso de linguagens paralelas quase sempre produz programas com código bem estruturado e com desempenho melhor que o obtido com a utilização de compiladores e extensões paralelas.

4.4.2 - FERRAMENTAS DE AUXÍLIO À PARALELIZAÇÃO

A paralelização automática de programas seqüenciais é um processo complexo, que envolve, entre outras tarefas, intensa análise do código original (por exemplo, investigação de dependência de dados e de controle), mas nem sempre os resultados da paralelização automática são satisfatórios. Às vezes é preferível não paralelizar um trecho de código porque o tempo de carga em um processador remoto pode ser maior do que o tempo gasto na execução seqüencial.

A abordagem mais apropriada à paralelização de programas envolve uma combinação de intuição humana, análise automática e transformação de código. A intuição humana, auxiliada pelas ferramentas de análise automática de código, fornece a heurística necessária para uma boa paralelização do código [KENNEDY, 1992].

Alguns exemplos de ferramentas de auxílio à paralelização são as ferramentas *p2d* [PEREZ, 1996], IDA [MERLIN, 1994] e ParaScope [CALLAHAN, 1988], voltadas para o código fonte escritos em Fortran.

4.4.3 - FERRAMENTAS DE DEPURAÇÃO

As ferramentas de depuração são utilizadas para a determinação e correção de erros nos programas em desenvolvimento. A depuração de programas paralelos apresenta algumas dificuldades, dentre as quais cita-se a não repetição de resultados (programas incorretos podem apresentar resultados não determinísticos). Essas ferramentas podem ser agrupadas, segundo MacDowell [MCDOWELL, 1989], em duas categorias:

- Ferramentas interativas: são aquelas que permitem que a execução do programa seja feita passo a passo, isto é, interrompendo-se a seqüência de execução, para que as variáveis possam ser inspecionadas ou até mesmo modificadas.
- Ferramenta *post-mortem*: utilizadas após o término da execução do programa, geralmente após a ocorrência de um erro. A utilização dessas permite observar apenas os efeitos finais do erro, tal como o valor de uma determinada variável.

4.4.4 - FERRAMENTAS DE ANÁLISE DE DESEMPENHO

As ferramentas de análise de desempenho são utilizadas para fornecerem informações sobre a execução de programas paralelos. Essas informações podem ser analisadas e utilizadas para a modificação do código de um programa paralelo com o objetivo de aumentar a velocidade de sua execução. Algumas dessas ferramentas utilizadas para análise de desempenho são: XPVM [GEIST, 1994, KOHL, 1997], o XMPI [XMP, 1997] e o VT (*Visualization Toll*) da IBM [PEA, 1995].

CAPÍTULO 5

5. VISÃO COMPUTACIONAL

5.1 - VISÃO COMPUTACIONAL E ROBÓTICA

Nos últimos anos, houve um grande avanço nas pesquisas e aplicações práticas na área de visão computacional. Área como reconhecimento de faces, reconhecimento de objetos, robótica, automação industrial, sensoriamento remoto, medicina, etc., utilizam os conceitos de visão computacional.

Analisando sobre o ponto de vista de automação industrial, podemos dividir a área de visão computacional em duas áreas: Processamento de Imagens e Reconhecimento de Padrões.

- Processamento de Imagens: área responsável principalmente pelo tratamento de operações nas imagens com a finalidade de melhorar sua qualidade, ou enfatizar os aspectos de importância particular.
- Reconhecimento de Padrões: essa área está relacionada com a identificação ou interpretação das imagens. Nessa fase é feita a extração das informações da imagem.

Os Sistemas de Visão Artificial é definido como um sistema computadorizado capaz de adquirir, processar e interpretar imagens correspondentes a cenas reais [OGÊ, 1999]. O Sistema de Visão Artificial, pode ser dividido em: aquisição, pré-processamento, segmentação, extração de características e reconhecimento e interpretação.

- Aquisição da Imagem: neste primeiro processo temos um elemento sensor (por exemplo câmera CCD) e um digitalizador. O elemento sensor converterá a informação óptica em sinal elétrico e o digitalizador transformará a imagem analógica em imagem digital.
- Pré-processamento: a função desta fase é melhorar a qualidade da imagem para as etapas subseqüentes.
- Segmentação: sua função básica é de dividir uma imagem em suas unidades significativas, ou seja, nos objetos de interesse que a compõem.
- Extração de Características: nesta etapa extraem-se as características das imagens resultantes da segmentação através de descritores.

- Reconhecimento e Interpretação: denominamos reconhecimento o processo de atribuição de um rótulo a um objeto baseado em suas características, e a interpretação consiste em atribuir significado a um conjunto de objetos reconhecidos.

Uma das áreas principais da utilização da visão artificial é a robótica, com aplicações que incluem a manipulação de pequenos objetos em operações repetitivas, montagem de produtos formados por diversas peças, além de tarefas de inspeção visual. Pode também ser utilizada em sistemas capazes de analisar o meio ambiente de operação, levando em conta aspectos característicos do meio, para tomada de decisão, utilizando realimentação sensorial sofisticada, baseada principalmente em dados obtidos do sistema de visão acoplados ao robô. Outra aplicação encontrada é em sistemas de rastreamento de alvo que utilizam a informação visual para implementar tarefas como soldagem e pintura [SONG & CHANG, 1996][NAJJARI & STEINER, 1995].

Podemos considerar um robô, como sendo constituído de três elementos principais, conforme pode ser visto na figura 5.1 abaixo.

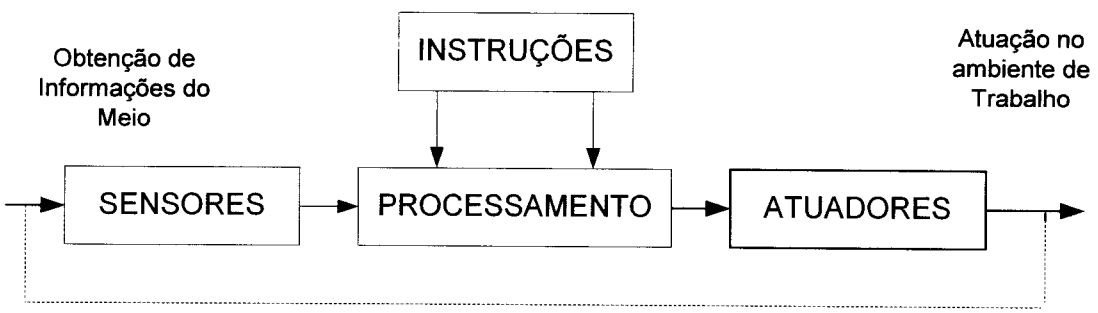


Figura 5.1 - Estrutura básica de um robô

O elemento de processamento é o bloco principal, controlando precisamente quais ações o sistema executará. Nos sistemas mais simples ele permite o movimento retilíneo em uma seqüência programada de instruções definidas por um conjunto específico de instruções, sendo o ambiente de operação constante, ou seja, todos os componentes são individualmente localizados e todas as relações espaciais são completamente determinadas.

Em um ambiente mais sofisticado, ou seja, não controlado, a unidade de sensoriamento desempenha um papel extremamente importante na operação global do sistema, já que diversas

variáveis do meio devem ser monitoradas e controladas, de acordo com os requisitos definidos pelo processo em execução.

Um robô para que possa ter as habilidades de interação com o meio ambiente de trabalho de forma precisa e inteligente, ele deve operar com um poderoso sistema de realimentação de sinais. A visão se enquadra dentro desse importante componente do sistema, pois ela proporciona uma variedade de informações que podem habilitar um mecanismo a trabalhar com incertezas inerentes à tarefa, reagir a variações do meio, e de forma delicada, recuperar-se de falhas [NAYAR et al., 1996].

A visão computacional e suas aplicações são de grande importância na obtenção de meios tecnológicos para a implementação de robôs inteligentes, processando e analisando imagens e fornecendo sua descrição. Tal descrição é normalmente na forma simbólica e depende da aplicação em questão.

Os sistemas de visão computacional estão relacionados a três campos específicos [KARATHANASSI et al., 1996]: processamento de imagens digitais, reconhecimento de padrões e análise de cena.

- Processamento de Imagens digitais: neste campo temos diversos passos de pré-processamento, em cada aplicação específica. Na fase de aquisição da imagem, a imagem resultante pode apresentar diversas imperfeições, tais como presença de pixels ruidosos, contraste ou brilho inadequados e outros diversos erros devidos a fatores internos e externos ao processo que degradam a qualidade da mesma, afetando consequentemente os dados obtidos através da análise subseqüente. Através do processamento digital, novas imagens são obtidas, utilizando diversas técnicas as quais minimizam tais erros e melhoram a qualidade das imagens obtidas. As operações realizadas nesta etapa são ditas de baixo nível porque trabalham diretamente com os valores de intensidade dos pixels.
- Reconhecimento de Padrões: neste se trabalha com a classificação de objetos em categorias definidas. Cada categoria é única, baseada em características específicas. Cada objeto de imagem é classificado em uma determinada categoria depois de comparar a suas características individuais.
- Análise de Cena: a análise de cena trata da transformação de uma representação simbólica da imagem para outro que seja menos complicada e mais fácil de ser interpretada. Geralmente, representação de conhecimento e inferência são utilizados para este propósito.

Os avanços na área de visão artificial estão diretamente ligados ao desenvolvimento de hardware e muitos destes desenvolvimentos são impulsionados pela necessidade de processamento mais rápido de imagens.

A seguir temos algumas das aplicações de visão artificial em robótica.

- Detecção de presença ou tipo e objeto;
- Determinação de localização e orientação de objeto antes do acionamento da garra;
- Realimentação, durante a manipulação com a garra;
- Realimentação para controle de trajetória em processos de soldagens e outros processo contínuos;
- Realimentação em processo de ajustes, durante operações de montagem;
- Leitura de códigos de identificação;
- Contagem de objetos;
- Inspeção de peças e objetos em processos de controle de qualidade.

A configuração básica de um sistema de visão é mostrada na figura 5.2 abaixo.

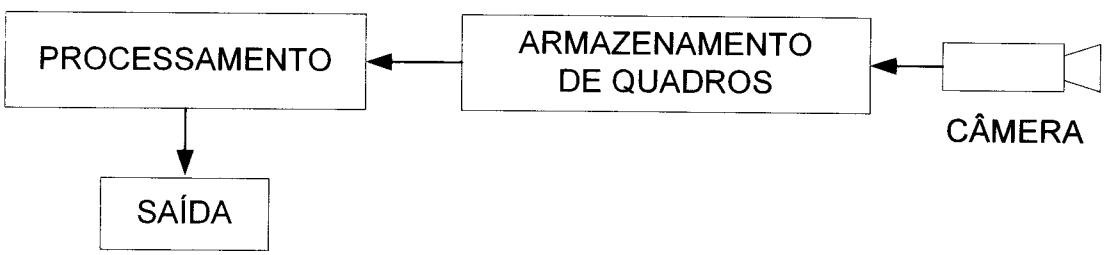


Figura 5.2 - Sistema básico de visão.

A câmera é utilizada para capturar uma imagem que será armazenada e acessada pelo sistema de processamento e análise utilizado.

A informação obtida da câmera utilizada gera uma imagem em tons de cinza na qual o nível de intensidade de cada pixel da imagem é representado por um valor binário. Normalmente se trabalha com 6 ou 8 bits para codificar a escala de cinza. Quando se utiliza cor, utiliza-se um esquema de filtragem apropriado, aumentando-se assim o tempo de processamento. A resolução é determinada pelo número de pixels utilizado. Para diversas

aplicações industriais, imagens de 256x256 ou 512x512 pixels, trabalhando com escala de tons de cinza de 8 bits, necessitando de 64K ou 256K bytes de armazenamento são suficientes. Para outras aplicações como na área médica e de satélites, resoluções bem maiores são necessárias, trabalhando com imagens da ordem de 10^5 x 10^5 pixels.

Após a imagem ser capturada pela câmera e armazenada, ela deve ser processada para extração de informações nela contida, envolvendo um estágio de pré-processamento, para correção de distorções, e em um estágio de extração de características, no qual características determinantes na informação da imagem são extraídas e utilizadas para o propósito de classificação.

Em robótica podemos citar os seguintes tipos de visão computadorizada:

- **Bidimensional com objetos isolados e imagem binária:** utilizadas com o objetivo de determinar a posição e orientação de um objeto, de tal maneira que um robô possa manipulá-lo. Os objetos normalmente são do mesmo tipo, sendo desnecessária a tarefa de reconhecimento.
- **Bidimensional, com objetos isolados e imagens em tons de cinza:** onde a informação de tons de cinza é utilizada para delimitação de fronteira e, consequentemente, para diferenciação de objetos.
- **Bidimensional, com objetos sobrepostos:** neste caso as comparações são feitas, levando em conta dados como ângulos e distâncias, formando modelos que são utilizados para determinar quais os tipos de objeto presentes na cena.
- **Inspeção Bidimensional:** onde a presença de defeitos e falhas é checada, numa imagem do objeto sob análise, utilizando métodos de comparação e definição de regiões de interesse.
- **Rastreamento Bidimensional:** utilizado por um manipulador ou robô, para seguir ou rastrear uma trilha definida, utilizando-a como base para a operação que está realizando (por exemplo, soldagem).
- **Extração de informação Tridimensional de um objeto**: neste caso o sistema deve ser capaz de medir algumas dimensões específicas de uma classe determinada de objetos. Neste caso normalmente é utilizada a técnica de visão estéreo para que a informação de profundidade possa ser obtida.
- **Análise Tridimensional de cena:** onde as informações obtidas dos itens anteriores são utilizadas para determinar situações específicas presentes numa determinada cena, utilizando diversas técnicas e procedimentos.

Na área de visão artificial encontramos diversos trabalhos sendo desenvolvidos, levando em consideração os requisitos de integração de informações sensoriais no esquema de acionamento e posicionamento utilizados.

5.2 - PROCESSAMENTO DIGITAL DE IMAGENS

Na manipulação de imagem, é fundamental gerar um método apropriado de representação dos dados obtidos desta imagem. Esse método deve conter as informações que define as características importantes da imagem e deve proporcionar uma base para o processamento eficiente e conveniente através do computador.

Uma imagem monocromática pode ser descrita matematicamente por uma função bidimensional $f(x, y)$ da intensidade luminosa, sendo seu valor, em qualquer ponto de coordenadas espaciais (x, y), proporcional ao brilho (ou nível de cinza) da imagem naquele ponto.

Para o propósito de visão computacional, será utilizado o conceito de imagem digital, o qual pode ser obtido de duas maneiras [FAIRHUST, 1988]:

- **Digitalização Espacial:** é aquela em que envolve a representação da função contínua original como um arranjo de exemplos para pontos discretos dentro da estrutura bidimensional de referência. Esta imagem, digitalizada especialmente, consiste de N x N exemplos igualmente distribuídos, onde cada ponto discreto do arranjo é identificado como um *pixel*.
- **Digitalização em Amplitude:** neste tipo um nível de intensidade de imagem é indicado como um nível de cinza, e a faixa total disponível de tais níveis é referenciada como escala de tons de cinza. O número de níveis de cinza é escolhido, por conveniência computacional, como potência de 2, e então qualquer arranjo N x N tem um nível de amplitude, g, onde $0 \leq g \leq 2^n - 1$ e n é um inteiro.

Um caso particular de digitalização de amplitude ocorre quando a escala de cinza consiste de somente dois níveis possíveis, ou seja, para $n = 1$, tal que a imagem resultante consiste de um arranjo de pontos, cada um dos quais completamente branco ou completamente preto. Tal imagem é conhecida como *imagem binarizada.* A geração de uma imagem binarizada a partir de uma imagem geral em escala de tons de cinza é bastante simples, basta apenas definir um valor apropriado de *thresholding* (limiarização) para particionar a imagem em pixels com somente um de dois valores possíveis.

Assim, se $f(x,y)$ de N níveis de cinza produz à saída uma imagem $g(x,y)$, chamada de imagem limiarizada, cujo número de níveis de cinza é menor que N. Normalmente, $g(x,y)$ apresenta dois níveis de cinza, sendo:

$$g(x,y) = 1 \quad \text{se } f(x,y) \geq T$$
$$g(x,y) = 0 \quad \text{se } f(x,y) < T \qquad (5.1)$$

onde os pixels rotulados com **1** correspondem aos objetos e os pixels representados por **0** correspondem ao fundo da imagem e **T** é um valor de tom de cinza predefinido, ao qual denominamos limiar. Agora temos que *g(x,y)*, corresponde a imagem binarizada de *f(x,y)*. Assim, como as imagens binarizadas trabalham com um único bit de informação, elas representam vantagens no processo de manipulação de imagens.

O processamento digital de imagens pode ser entendido como sendo a análise e manipulação de imagens pelo computador, e tem como finalidade:

- Extrair informações da imagem;
- Transformar a imagem (por exemplo, aumentar o contraste, realçar as bordas) de tal modo que a informação seja facilmente reconhecida.

5.2.1 - Operações Básicas

As operações básicas com imagem são aquelas realizadas para melhoria da qualidade da imagem, as quais alteram os valores dos níveis de cinza dos pontos da imagem, de tal forma que a imagem resultante destas operações venham a apresentar características desejáveis que a imagem original não contém. Podemos resumi-las em dois grupos de operações que são:

- **Realce da imagem:** que consiste em enfatizar alguma característica interessante da imagem original.
- **Restauração:** na restauração tem-se a preocupação em corrigir alguma distorção sofrida pela imagem.

Podemos enquadrar nestes dois grupos as operações de Transformação de Escala de Cinza, Modificação de Histograma e Filtragem, Modelos de Degradação, Técnica de Reconstrução, entre outras.

5.2.2 - Operações Estruturais

Neste tipo de operação encontram-se as transformações geométricas, nas quais se modifica a geometria da imagem, mantendo-se o máximo possível os valores dos níveis de cinza. Dentre elas temos:

- Normalização de posição.
- Ampliação e redução.
- Rotação.
- Detecção de borda.
- Segmentação.
- Codificação.

Estas operações normalmente são realizadas, na etapa de pré-processamento, gerando uma imagem com melhor qualidade em relação àquela originalmente obtida pela câmera.

As operações de Filtragem e Detecção de bordas são as mais importantes dentro do estudo de visão artificial. A Detecção de borda segmenta objetos presentes em uma determinada cena para definição de regiões (grupos de pontos conectados) as quais descrevem os objetos de uma cena, e a Filtragem é utilizada principalmente para a eliminação de ruídos da imagem.

5.3 - RECONHECIMENTO DE PADRÕES

Reconhecimento de padrões, nos sistemas de visão computacional, tem como função a extração de características de uma determinada imagem ou a classificação de seu conteúdo, podendo ser dividido em: Classificação de Padrões e Reconhecimento Sintático, descritos a seguir.

- **Classificação de Padrões:** na qual o objetivo é relacionar um objeto desconhecido a uma determinada classe. Utiliza-se, normalmente na classificação de padrões, abordagens estatísticas para a implementação do classificador, o qual é fundamentado em uma rigorosa base probabilística, assumindo que a estrutura e distribuição dos dados no espaço do problema são conhecidos antecipadamente. Logo a construção de classificadores se baseia na conversão de probabilidades já conhecidas das classes dos padrões de treinamento em medidas de probabilidade condicionada de saída de um dado padrão pertencer a uma dada classe de treinamento.
- **Reconhecimento Sintático:** no qual a descrição de um objeto é feita através de relações entre suas partes. O reconhecimento sintático fundamenta-se no fato de que o inter-relacionamento ou interconexões de características forneça importantes informações estruturais que possam ser utilizadas na descrição de padrões e em sua classificação [VASCONCELOS, 1995].

Os sistemas de reconhecimento de padrões podem ser divididos normalmente em três etapas:

- **Treinamento:** fase em que o sistema aprende a discriminar as classes através da apresentação de exemplos representativos das classes de padrões.
- **Teste:** onde um conjunto de padrões não vistos na etapa de treinamento é apresentado, de tal maneira que a capacidade de generalização (derivação correta de classes) do sistema é avaliada.
- **Execução:** é a fase prática do reconhecimento de padrões.

Temos na figura 5.3 a representação típica de uma estrutura de um sistema de reconhecimento de padrões [VASCONCELOS, 1995].

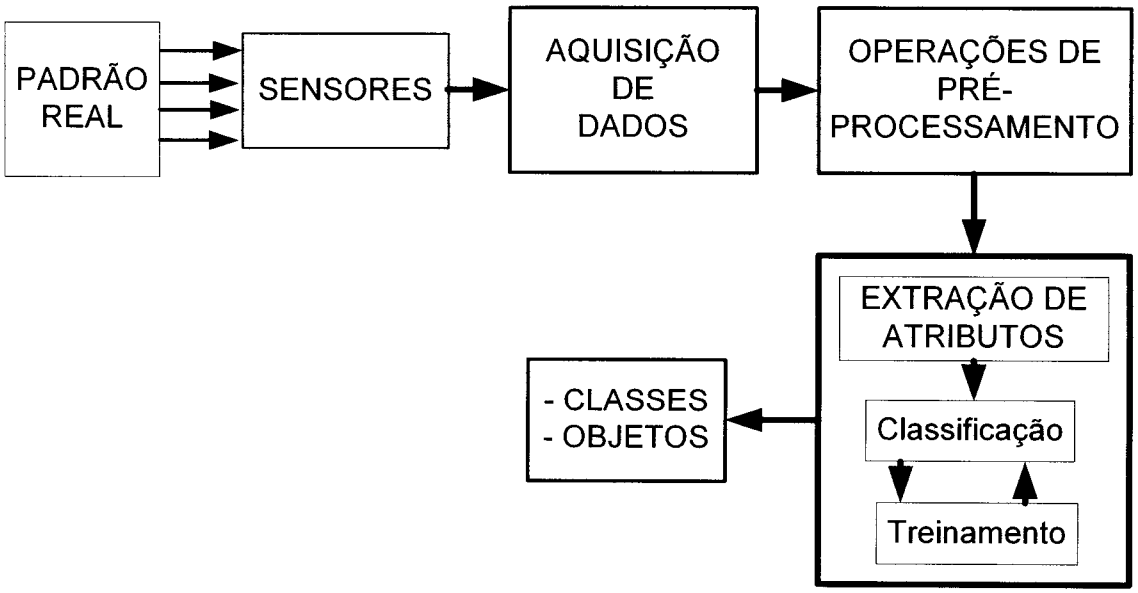


Figura 5.3 - Sistema de Reconhecimento de Padrões

O sistema de reconhecimento de uma imagem é então dividido em Aquisição de dados, Pré-processamento, Extração de atributos e Classificação.

- **Aquisição de Dados:** os objetos a serem classificados devem ser capturados por sensores de imagens, que produzirão uma representação das imagens com algumas simplificações em relação às imagens reais, sendo discretas, bidimensionais, limitadas em extensão e em número de cores ou níveis de cinza possíveis. Estas representações são descritas em forma numérica, tornando-se possível o armazenamento e o processamento das imagens através do computador. As imagens tipicamente trabalhadas apresentam o tamanho de 128x128, 512x512 pixels, sendo 256 os níveis de cinza. Esta fase é extremamente importante, sendo o que o seu desempenho na obtenção da cena estará intimamente ligado à natureza e qualidade dos dados de entrada [DUDA et al, 1979]. Outro fator que se deve preocupar na obtenção da imagem é a iluminação da cena, que deve ser bem iluminada, pois a luz ambiente geralmente não é aceitável por produzir imagens com baixo contraste, sombras e outros ruídos.

IESC-USP SERVIÇO DE BIBLIOTECA INFORMAÇÃO

- **Pré-processamento:** onde a imagem é filtrada para minimizar os ruídos e distorções que possa resultar do processo de aquisição dos dados na entrada. Nesta fase, pode-se também realizar operações de segmentação e normalização sobre os dados de entrada. A finalidade da segmentação é para separar os padrões que estejam de alguma forma ligados a outros, enquanto que a normalização é feita para modificar os dados de entrada reduzindo-os à escala normal das classes dos padrões.
- **Extração de atributos:** nesta etapa, são obtidas as características relevantes que possam ser usadas na caracterização de padrões. Tais características podem ser numéricas (áreas, volume, etc), simbólicas (cor), ou uma combinação das duas. A extração de características é uma forma de compressão de dados, já que reduz a dimensionalidade da informação, eliminando a redundância e aumentando a discriminação entre as classes.

  Para a escolha do conjunto de características deve-se levar em conta algumas propriedades dos mesmos [PARKER, 1994]:

  - Velocidade de processamento.
  - Grande discriminação de classe.
  - Pouca variância em cada classe.
  - Completude da descrição.

  A utilização de características invariantes a transformações é um método bastante utilizado no reconhecimento de padrões, sendo que a escolha destes na maioria das vezes influencia as taxas de reconhecimento alcançadas. Outros fatores como o tempo de cálculo dos atributos na fase de pré-processamento e a redução de dados, devem ser levados em consideração [YUCEER & OFLAZER, 1993].

  Nessa fase de seleção de características deve-se escolher as que melhor definam um modelo de entrada [BEALE AND JACKSON, 1990]. Estas podem ser naturais quando são extraídas da aparência visual da imagem, tendo como exemplo a luminância de uma região de pixels e a escala de cinza de regiões de textura, ou artificiais quando são resultantes da manipulação da imagem, como exemplo tem-se a amplitude de histogramas e o espectro de freqüência espacial [PRATT, 1991].

- **Classificação:** nesta etapa se faz um mapeamento entre as características e as classes dos padrões, ou seja, através dos conjuntos de características, o classificador define uma determinada classe baseada numa função de decisão. As n características podem ser visualizadas como coordenadas de um espaço n-dimensional e os objetos a serem identificados podem ser representados por pontos

neste espaço. As diversas classes de objetos particionam o espaço de características, formando subconjuntos durante a fase de treinamento do sistema. Tais subconjuntos são regiões de decisão. Assim, um objeto será atribuído a uma determinada classe se seu pontos de características representativas estiverem em apenas uma região de decisão, e em caso contrário, o objeto não é classificado (Figura 5.4).

Logo neste processo de classificação utiliza-se de um vetor de características extraído dos dados de entrada. Normalmente, várias informações são necessárias para que se possa distinguir adequadamente o conjunto de dados de entradas que pertencem a diferentes classes [BEALE AND JACKSON, 1990]. O vetor de características deve reunir todas as medidas utilizadas num classificador e o número de elementos do vetor *n* cria um espaço de características n-dimensional.

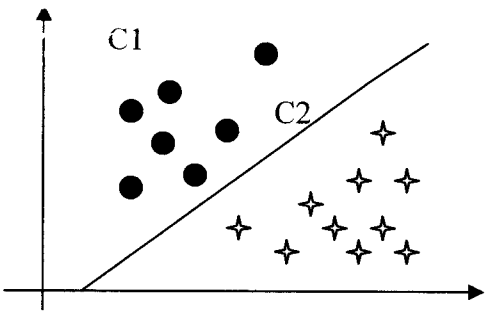


Figura 5.4 - Escolha das características de entrada.

Em reconhecimento de padrões, tem-se utilizado Redes Neurais em diversas aplicações.

5.3.1 - CARACTERÍSTICAS DE UMA IMAGEM

5.3.1.1 - Características radiométricas

A característica básica de amplitude de uma imagem é o valor de seu "pixel" em análise. Esta medida pode ser feita em um ponto específico ou sobre uma vizinhança (média), o que traz como vantagem a redução de ruído. Seja uma imagem F(i,j), sendo (i,j) a coordenada do ponto central de uma janela WxW. A imagem da média (M(i,j)) é dada por:

$$M(i,j) = \frac{1}{W^2} \sum_{m=-W/2}^{W/2} \sum_{n=-W/2}^{W/2} F(i+m, j+n) \tag{5.2}$$

A **mediana** dos "pixels" em uma vizinhança W x W é definida como sendo a amplitude do "pixel" da janela W x W no qual metade dos "pixels" são menores ou iguais a esta amplitude, e metade são iguais ou maiores [PRATT, 1991].

Dentro deste estudo temos como característica importante que podemos extrair de uma imagem o seu **desvio padrão (S),** conhecido também conhecido como dispersão de imagem. O desvio padrão pode ser calculado por:

$$S(i,j) = \frac{1}{W^2}[\sum_{m=-W/2}^{W/2} \sum_{n=-W/2}^{W/2} [F(i+m, j+n) - M(i,j)]^2]^{1/2} \tag{5.3}$$

5.3.1.2 – Características de textura

A textura é considerada uma das mais importantes características usadas para identificar objetos ou regiões de interesse em uma imagem [HARALICK et al, 1973]. Esta é descrita pelas primitivas que compõe a imagem e pela dependência espacial (ou interação) entre as primitivas. Uma primitiva tonal pode ser descrita em termos da média dos tons de cinza, ou dos valores de tons máximo e mínimo de uma região [HARALICK, 1979].

O tom e textura não são conceitos independentes, coexistindo sempre em uma mesma imagem, embora algumas vezes uma propriedade possa dominar a outra. Neste caso, há uma tendência de se falar neste caso somente em tom ou somente em textura [HARALICK, 1979]. No caso em que uma área de imagem apresenta uma pequena variação de primitiva tonal, a propriedade dominante desta área é o tom; caso contrário, se houver uma grande variação de primitiva tonal, a propriedade dominante será a textura.

Existem três métodos em processamento de imagem para descrever textura de regiões: método estatístico, método estrutural e método espectral [GONZALEZ & WINTZ, 1987].

No modelo estatístico a textura é descrita por regras estatísticas que governam a distribuição e a relação dos níveis de cinza [BALLARD & BROWN, 1982]. No modelo estrutural são consideradas as primitivas que formam um padrão repetitivo e descreve tais padrões são descritos em termos das regras que o geram [BALLARD & BROWN, 1982]. Nas técnicas espectrais usa-se o espectro de Fourier, para detectar o período de uma imagem pela identificação de alta energia (ou picos estreitos no espectro de freqüência) [GONZALEZ & WINTZ, 1987].

- **Métodos estatísticos:** Em texturas não tem normalmente uma regularidade geométrica, apresentando variações que são difíceis de serem descritas pela sua forma, mas pode ser descrita por modelos estatísticos [BALLARD & BROWN, 1982]. Esse

método trabalha melhor com as texturas naturais, uma vez que tais texturas possuem variações nas formas geométricas que compõem a imagem.

Além das características de amplitude (média e mediana) para medir textura, Haralick [HARALICK, 1979] cita: textura de bordas, elementos estruturais e matrizes de dependência espacial dos níveis de cinza (SGLDM).

. **Textura de bordas:** considera a textura sendo a quantidade de bordas existente em uma área. Uma borda local é uma área pequena na imagem onde os níveis de cinza mudam rapidamente [BALLARD & BROWN, 1982]. Para detectar essas bordas são usados operadores de bordas conhecidos, tais como: Sobel, Prewitt, Roberts, Laplaciano entre outros.

. **Elemento estrutural:** é uma região de células que constituem uma forma específica, tais como: linhas ou quadrados, e gera uma nova imagem binária pela erosão do elemento estrutural com a imagem original [HARALICK, 1979]. A característica textural pode ser obtida pela contagem dos "pixels" de valor igual a 1. O elemento estrutural está ligado à matemática morfológica pelo uso da função de erosão. A morfologia matemática pode ser aplicada em várias áreas de processamento e análise de imagens, com objetivos tão distintos como realce, filtragem, segmentação, detecção de bordas, esqueletização, afinamento, análise de formas e codificação de imagens, entre outras [OGÈ, 1999].

. **Matriz de dependência espacial do nível de cinza (SGLDM):** considera-se como o método estatístico mais utilizado. Este consiste da construção de matrizes, contando o número de ocorrência de pares de "pixels" com um dado nível de cinza em um dado deslocamento [CHEN et al, 1995]. Em outras palavras, o nível de cinza co-ocorrente pode ser especificado em uma matriz de freqüências relativas P_{ij} com a qual duas células vizinhas separadas pela distância d e com uma orientação "θ" ocorrem na imagem, uma com nível de cinza i e outra com nível de cinza j [HARALICK, 1979]. Haralick [HARALICK et al, 1973] sugere 28 (vinte e oito) características de textura que podem ser extraídas da matriz de dependência espacial dos níveis de cinza. Entre essas características encontram-se:

. Segundo momento angular: que indica o quão uniforme é a imagem:

$$f_u = \sum_i \sum_j \{p(i,j)\}^2 \tag{5.4}$$

. Contraste, indica a quantidade de variações locais presente na imagem.

$$f_c = \sum_{n=0}^{Nq-1} n^2 \{ \sum_i^{Nq} \sum_{|i-j|=n}^{Nq} p(i,j) \}$$

(5.5)

. Entropia: mede a não uniformidade da imagem:

(5.6)

$$f_e = -\sum_i \sum_j p(i,j).\log(p(i,j))$$

. Momento da diferença inversa: medida de homogeneidade da imagem:

$$f_h = \sum_i \sum_j \frac{1}{1+(i-j)^2}.p(i,j)$$

(5.7)

Os dados estatísticos podem ser computados indiretamente através do histograma da imagem. A distribuição de probabilidade de 1ª. ordem da amplitude do histograma pode ser definido como [PRATT, 1991]:

$$P(b) = P\{I(i,j) = b\}$$

(5.8)

onde P(b) é a quantidade de "pixels" no nível b ($0 \leq b \leq L-1$).

O histograma estimado de 1ª. ordem é dado por [PRATT, 1991]:

$$P(b) = \frac{N(b)}{M}$$

(5.9)

onde M é o número total de "pixels" numa janela e N(b) é o quantidade de "pixels" de nível b na mesma janela.

O histograma estimado de 2ª. ordem, também denominado de matriz de co-ocorrência, é definido por:

$$P(a,b) = P\{F(i,j) = a, I(m,n) = b\}$$

(5.10)

onde a e b representa valores da amplitude dos "pixels" e o histograma de 2ª ordem estimado será:

$$P(a,b) = \frac{N(a,b)}{M}$$

(5.11)

onde M representa o número total de "pixels" numa janela e N(a,b) denota o número de ocorrência na qual F(i,j) = a e F(m,n) = b.

Através do histograma várias características podem ser extraídas, entre elas:

Histograma de 1ª ordem

média:

$$S_M = \bar{b} = \sum_{b=0}^{L-1} b.P(b)$$

(5.12)

desvio padrão:

$$S_D = [\sum_{b=0}^{L-1} (b-\bar{b}).P(b)]^{1/2}$$

(5.13)

curtose

$$S_K = \frac{1}{\tau_b^{\,3}} \sum_{b=0}^{L-1} (b-\bar{b})^4.P(b) - 3$$

energia:

$$S_N = \sum_{b=0}^{L-1} [P(b)]^2$$

(5.14)

entropia:

$$S_E = -\sum_{b=0}^{L-1} P(b).\log_2[P(b)]$$

(5.15)

Histograma de 2ª ordem

auto correlação:

$$S_A = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} a.b.P(a,b)$$

(5.16)

covariância;

$$S_C = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} (a-\bar{a}).(b-\bar{b}).P(a,b)$$

(5.17)

onde:

$$\bar{a} = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} a.P(a,b) \qquad \bar{b} = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} b.P(a,b)$$

(5.18)

energia:

$$S_G = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} [P(a,b]^2$$

(5.19)

entropia:

$$S_T = -\sum_{a=0}^{L-1}\sum_{b=0}^{L-1} P(a,b).\log_2[P(a,b)]$$

(5.20)

- **Métodos Estruturais:** A seleção de primitivas dos métodos estruturais pode ser agrupada em dois tipos, os que enfatizam as bordas e os que enfatizam a região [FU, 1974].

. **Bordas e esqueletos:** O conjunto de primitivas mais comumente usadas para descrever bordas e esqueletos é o código cadeia [FU, 1974]. O código cadeia consiste de

segmentos de linhas que devem pertencer a uma grade fixa com um conjunto fixo de orientação possíveis [BALLARD & BROWN, 1982]. A grade de orientação a ser usada pode ser 4 ou 8 conectada (Figura 5.5). Cada segmento de linha da imagem é assinalado com dígito referente à orientação (Figura 5.6).

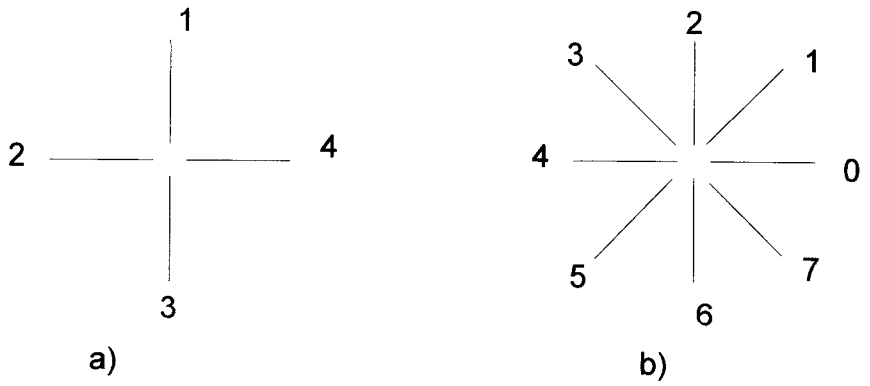


Figura 5.5- Código cadeia. (a) orientação para 4-conectado.

(b) orientação para 8-conectado

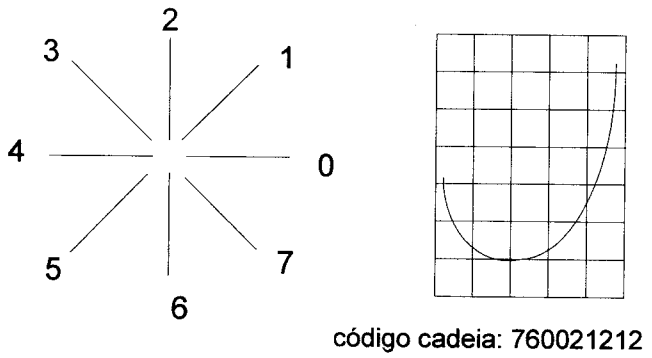


Figura 5.6 - Exemplo da aplicação do código cadeia.

Esse código é independente de posição e sua derivada é útil porque o torna invariante à rotação de borda [BALLARD & BROWN, 1982]. Através do código pode-se calcular a área de uma figura fechada a partir do seguinte algoritmo:

Computa o primeiro ponto (x,y) da figura:

1) área = 0

   pos_y = y

2) para cada elemento do código cadeia:

se 0 -> área = área - pos_y

se 1 -> pos_y = pos_y + 1

se 2 -> área = área + pos_y

se 3 -> pos_y = pos_y - 1

. **Região:** enfatizando a região as primitivas básicas são semi-planos no campo de observação. Neste caso, vê-se que qualquer figura pode ser aproximada por um polígono arbitrário, o qual pode ser expresso como a união de um número finito de polígonos convexos [FU, 1974]. Cada polígono convexo pode ser representado como a intersecção de um número finito de meio planos. A definição de uma ordem adequada do polígono convexo compõe o polígono arbitrário. Logo, é possível determinar o conjunto mínimo de polígonos que através da união representa o polígono arbitrário (Figura 5.7).

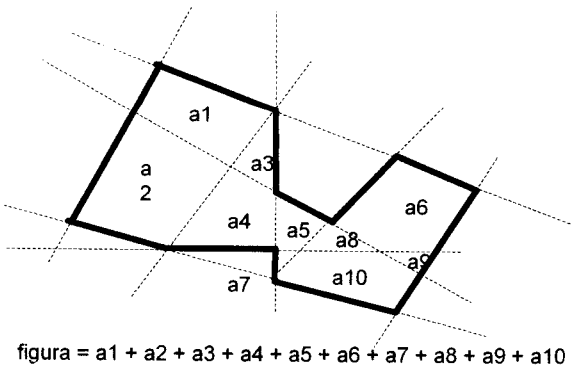


Figura 5.7 – Figura aproximada por polígonos.

- **Métodos Espectrais:** O espectro de potência de Fourier bi-dimensional de uma imagem de textura freqüentemente revela a periodicidade e a direção da textura [CHEN et al, 1995]. Texturas grossas tendem a gerar componentes de baixas freqüências enquanto as texturas finas geram componentes de altas freqüências [PRATT, 1991]. A transformada de Fourier é definida como [SCHALKOFF, 1989]:

$$F(u,v) = \int\int \exp[-j.2\pi(ux - vy)].f(x,y).dx.dy \tag{5.21}$$

onde F(u,v) é forma geral. O espectro de potência é dado por:

$$|F(u,v)|^2 = F(u,v).F^*(u,v) \tag{5.22}$$

onde F^* denota o complexo conjugado.

A transformada de Fourier geralmente trabalha bem com texturas que possuem uma periodicidade dominante e seu desempenho cai quando a periodicidade enfraquece [CHEN et al, 1995].

5.3.2 - Classificação

A classificação de padrões pode ser divididas em duas categorias: técnicas numéricas e não numéricas. As técnicas numéricas incluem medidas determinísticas e estatísticas, as quais podem ser consideradas como uma medida feita no espaço geométrico do modelo, e as técnicas não numéricas são aquelas que possuem processamento de símbolos e são tratadas com métodos dos conjuntos fuzzy [BEALE & JACKSON, 1990]. Os métodos de classificação do vizinho mais próximo, o classificador linear e o classificador Bayesiano são os mais utilizados na prática, sendo que os dois primeiros são métodos determinísticos e o último estatístico.

- **Classificação do vizinho mais próximo:** este método se baseia no conceito da distância que um modelo desconhecido possui em relação às classes definidas em um espaço de modelos (Figura 5.8).

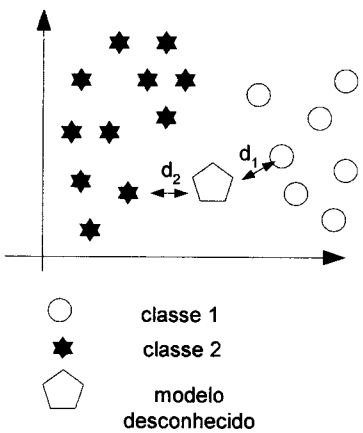


Figura 5.8 – Classificação pelo método do vizinho mais próximo. d_1 representa a distância do modelo desconhecido à classe 1 e d_2 a distância até à classe 2

Essa distância pode ser calculada de várias formas, entre as quais:

Distância de Hamming: $$H = \sum_n \left(|x_i - y_i| \right) \tag{5.23}$$

Distância Euclidiana: $$d(x,y) = [\sum_n (x_i - y_i)]^{1/2} \tag{5.24}$$

Distância City-Block: $$D_{cb} = \sum_n |x_i - y_i| \tag{5.25}$$

Distância Quadrado: $D_{sq} = \max|x_i - y_i|$ (5.26)

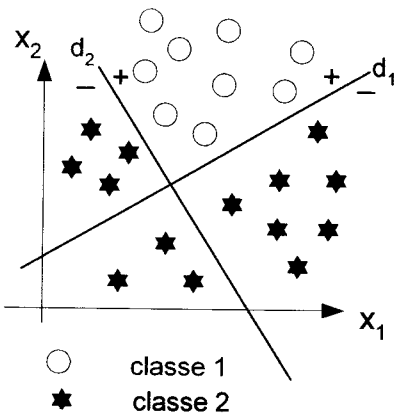
- **Classificador Linear:** este classificador divide o espaço de modelo através de funções discriminantes. Quando um novo modelo é apresentado ao classificador a função discriminante se encarrega em classificá-lo em uma ou outra classe. O limite de decisão entre classes no espaço de amostra define a função discriminante f(x) da seguinte forma [BEALE & JACKSON, 1990];

$$f_x = \sum_n W_i \cdot X_i \tag{5.27}$$

onde: X_i = i-ésima componente do vetor de entrada

W_i = i-ésima componente do vetor peso

A figura 5.9 mostra um exemplo de classificador linear:



a)

	Função discriminante	
	d1	d2
Classe 1	+	+
Classe 2	+	–
	–	+
	–	–

b)

Figura 5.9 - Exemplo de classificador linear. (a) duas funções discriminantes para separar as duas classes. (b) resultado da classificação através das funções discriminantes.

No exemplo da figura 5.9(a) existem duas classes e para separá-las foram necessárias duas funções discriminantes. A figura 5.9(b) mostra o resultado da classificação com base na saída da função discriminante.

- **Classificador Bayesiano:** A teoria de decisão de Bayes é uma aproximação estatística fundamental para o problema de reconhecimento de padrões. Esta aproximação

é baseada na suposição de que o problema de decisão seja colocado em termos probabilísticos e que todos os valores de probabilidades relevantes sejam conhecidos [DUDA & HART, 1973].

O problema de classificação é formulado a partir de w_i (i = 1, 2....n) classes, com a probabilidade de ocorrência (probabilidade a priori) dada por $P(w_i)$, onde $0 \leq P(w_i) \leq 1$ e $\Sigma P(w_i) = 1$.

A probabilidade condicional de um modelo pertencer à classe w_i, dado o fato do modelo possuir os valores do vetor de características **x** (probabilidade a posteriori), será:

$$P(w_i / \mathbf{x}) = \frac{p(\mathbf{x} / w_i).P(w_i)}{\sum_{i}^{n} p(\mathbf{x} / w_j).P(w_j)} = \frac{p(\mathbf{x} / w_i).P(w_i)}{p(\mathbf{x})} \tag{5.28}$$

onde $p(\mathbf{x}/w_i)$ é a função densidade probabilidade condicional do modelo ter os valores do vetor de características **x**, dado que o mesmo pertença à classe w_i.

Algumas vezes é mais prudente um sistema de automação que não decida sobre todas as classes [RAUBER, 1997]. Isso significa que deve ser introduzida a opção de rejeição através de uma nova classe (w_0).

– **Regra de Bayes para o mínimo erro:** Define-se o erro de um dado vetor de característica **x**, se for decidido que o mesmo pertence à classe w_j, embora a probabilidade posterior de outra classe w_i ser maior do que a probabilidade da classe w_j. A probabilidade do erro é dada por [DUDA & HART, 1973]:

$$P(erro / \mathbf{x}) = P(w_i / \mathbf{x}) \quad \text{se for decidido que } \mathbf{x} \text{ pertence à classe } w_j \tag{5.29}$$

Minimizar a probabilidade de erro para cada vetor de característica **x** é equivalente a escolher a classe que possui a maior probabilidade a posteriori de **x** [RAUBER, 1997].

Portanto, a decisão de bayes para o mínimo erro sem rejeição é dada por:

$$d(\mathbf{x}) = w_i \quad se \quad P(w_i / \mathbf{x}) > P(w_j / \mathbf{x}) \quad \text{p/ todo } j \neq i \tag{5.30}$$

Para permitir a rejeição deve ser introduzido um "threshold" de rejeição. Por simplicidade é definido como uma constante para todas as classes λ_r. Neste caso, λ_r é o mínimo das probabilidades a posteriori que é necessário para realizar a decisão [RAUBER, 1997]. Então, a regra de Bayes para o mínimo erro com a opção de rejeição torna-se:

$$d(\mathbf{x}) = w_i \quad se \quad P(w_i / \mathbf{x}) > P(w_j / \mathbf{x}) \qquad \text{p/ todo } j \neq i \text{ E } P(w_i/\mathbf{x}) > \lambda_r$$

$$d(\mathbf{x}) = w_0 \qquad \text{caso contrário} \tag{5.31}$$

– **Regra de Bayes para a máxima verossimilhança:** substituindo a equação 9.2 na equação 9.1 é fácil verificar que a regra de decisão não é afetada pela divisão do fator de normalização (p(**x**)). Portanto a regra de decisão de Bayes pode ser reescrita como:

$$d(\mathbf{x}) = w_i \quad se \quad p(\mathbf{x} / w_i).P(w_i) > p(\mathbf{x} / w_j).P(w_j) \quad \text{p/ todo } j \neq i$$

ou:

$$d(\mathbf{x}) = w_i \quad se \quad l(\mathbf{x}) = \frac{p(\mathbf{x} / w_i)}{p(\mathbf{x} / w_j)} = \frac{P(w_j)}{P(w_i)} \quad \text{p/ todo } j \neq i \tag{5.32}$$

onde $l(\mathbf{x})$ é chamado de razão de verossimilhança.

Quando as probabilidades a priori são iguais ($P(w_i)=P(w_j)$), ou quando não se tem informação útil sobre as probabilidades a priori, tem-se um caso particular em que a decisão é realizada pelo maior valor das funções densidades condicionais $p(\mathbf{x}/w_i)$. Essa decisão é chamada de decisão de Bayes para a máxima verossimilhança:

$$d(\mathbf{x}) = w_i \quad se \quad p(\mathbf{x} / w_i) > p(\mathbf{x} / w_j) \qquad \text{p/ todo } j \neq i \tag{5.33}$$

A representação do classificador Bayesiano é ilustrada na figura 5.10.

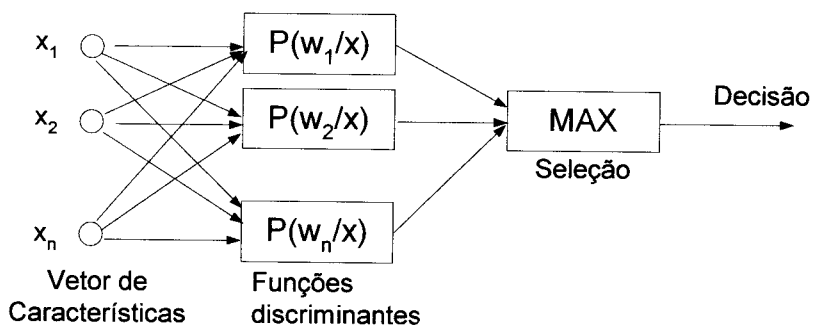


Figura 5.10 – Classificador Bayesiano.

5.3.3 - MÉTODOS PARA RECONHECIMENTO DE PADRÕES

Existem diversos métodos utilizados para a realização da tarefa de reconhecimento de Padrões. Os mais utilizados são: métodos estatísticos, métodos utilizando Redes Neurais, método K-Vizinhos Mais Próximos (KNN), método utilizando lógica Fuzzy e métodos utilizando algoritmos genéticos.

- **Métodos estatísticos:** no método de Reconhecimento Estatístico de Padrões, utilizam teoremas como o de Bayes para o cálculo da probabilidade das classes de objetos, assumindo conhecimento prévio da distribuição das variáveis utilizadas na classificação [MATERS, 1995]. Como exemplo deste método, tomemos um problema prático envolvendo um certo número de classes, conhecendo-se as funções de densidade de probabilidade associada a cada classe, utiliza-se a regra de probabilidade condicional de Bayes para estimar a classe de cada padrão.

  Dado conjunto de características ($A = a_1, a_2, \ldots, a_n$) como sendo um conjunto das medidas da instância cuja classe se deseja determinar, a probabilidade condicional $P(C_i/A)$ fornece uma visão do quanto às medidas estão relacionadas aos valores padrão para a classe determinada. A decisão por uma determinada classe será feita através de comparações destas probabilidades para cada classe, sendo o maior valor tomado como resultante.

  A probabilidade condicional é dada por:

$$P(C_i/A) = \frac{P(A/C_i).P(C_i)}{\sum_j P(A/C_j).P(C_j)} \tag{5.34}$$

- **K-Vizinhos mais próximos (KNN):** utiliza o conceito de relação de distância entre um modelo desconhecido e a dos padrões previamente aprendidos (conhecidos). Diversas abordagens podem se utilizadas para obter a distância entre atributos do objeto e dos padrões armazenados, entre elas podemos citar as distâncias Hamming, Euclidiana, City-Block e quadrada.

- **Redes Neurais:** as redes neurais vem sendo utilizadas no reconhecimento de padrões através de vários tipos de arquiteturas. Essas redes são utilizadas como base para o desenvolvimento adaptativo dos coeficientes das funções de decisão através de apresentações sucessivas de padrões de treinamento.

- **Redes Neuro-*Fuzzy* (*Logica Fuzzy*):** possui um conceito muito atrativo pois fornece uma estrutura flexível para representar conceitos vagos e pouco definidos,

representando e numericamente manipulando regras lingüísticas de maneira natural. Neste conceito, o reconhecimento de padrões é realizado a partir do conhecimento das fontes de informação (atributos, informação contextual, etc.) e um medida *fuzzy* é gerada para cada classe de padrão, usando uma medida subjetivamente gerada ou estimada dos dados de treinamento ou de valores de todos os subconjuntos das fontes de informação. Esta geração de medidas é a fase de treinamento para a abordagem *fuzzy*. Dado um padrão a ser classificado, uma função evidência $h_i(x_j)$ é avaliada para cada fonte de informação x_j e cada classe w_i. As funções h_i são então integradas em relação a suas correspondentes medidas *fuzzy* de classe, resultando em um valor confidência para cada classe. Estes valores confidência são usados para uma tomada de decisão final de classificação, ou seja, designar o padrão à classe com a maior confidência.

As redes neuro-*fuzzy* podem ser divididas em dois grupos [LIN &CUNNINGHAM III, 1995], onde no primeiro grupo o raciocínio *fuzzy* usa pesos *fuzzy* nas conexões dos nós e no segundo os dados são *fuzificados* na primeira ou segunda camada, mas os pesos da rede não são *fuzzy*, sendo este o grupo mais utilizado na prática.

- **Algoritmos Genéticos:** este conceito busca, como em redes neurais, simular um processo biológico natural, utilizando-se de dois conceitos básicos: representação de soluções possíveis em strings de dígitos binários (conceito de codificação) e a utilização de um algoritmo para avaliar a solução (conceito de avaliação). Em uma analogia biológica, os strings são cromossomos e cada dígito binário é um gene, sendo também utilizados com outros conceitos do tipo crossover e mutação. A descrição e utilização de tais conceitos, em conjunto com redes neurais, é apresentado por MARTINS [MARTINS, 1994], para reconhecimento de padrões.

5.4 - REDES NEURAIS

5.4.1 – Introdução

O interesse em redes neurais data do início dos anos 40, publicado por McCullloch & Pitts em 1943 [GONZALES &WOODS,1992], despertando interesse até os dias de hoje. Procura-se entender como que o cérebro constituído por uma rede de neurônios massivamente paralela, estes disparando em velocidades na ordem de milisegundos (comparadas com velocidades de nanosegundos para componentes eletrônicos disponíveis comercialmente), possam apresentar uma performance com velocidades incríveis de realização de tarefas como aquisição, armazenamento,

representação, recuperação e análise de dados sensoriais muito complexos. Isto leva-nos a verificar que o cérebro humano é superior ao computador em várias tarefas, como por exemplo, visão e reconhecimento de voz.

Os computadores são melhores em tarefas seqüenciais como aritmética (soma, multiplicação,...) e busca em banco de dados. Isso se deve ao fato de os computadores são máquinas seriais de alta velocidade, enquanto que o cérebro possui uma velocidade de processamento baixa, mas sendo altamente paralelo.

O cérebro humano apresenta várias características que são desejáveis a um sistema artificial [HERTZ et al, 1993]:

. É robusto e tolerante à falha. Células nervosas morrem diariamente sem afetar seu desempenho significativamente.
. É flexível. Ajusta-se facilmente a um novo ambiente pela "aprendizagem".
. Trabalha com informações nebulosas, probabilísticas, ruidosas ou inconsistentes.
. É altamente paralelo.
. É pequeno, compacto e dissipa pouca energia.

Em função dessas características apresentadas pelo cérebro humano, motivaram-se pesquisas em sistemas, conhecidos como redes neurais.

5.4.2 - Neurônio

5.4.2.1 - Neurônio humano: o cérebro humano possui algo da ordem de 100 bilhões de neurônios, organizados em uma rede complexa em que cada indivíduo pode ser conectado a diversos milhares de outros neurônios [GONZALEZ & WOODS, 1992]. Na figura 5.11 pode ser visto a imagem de um neurônio.

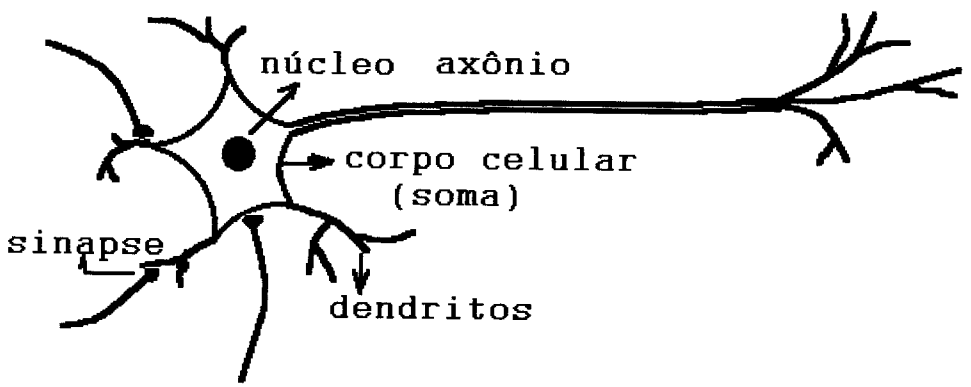


Figura 5.11 - Neurônio.

IFSC-USP SERVIÇO DE BIBLIOTECA

Os neurônios são delimitados por uma membrana celular e são constituídos basicamente de um corpo celular ou soma, um filamento alongado denominado axônio e dendritos.

A transmissão dos sinais entre células neurais é um processo eletroquímico. Os neurônios em geral apresentam apenas um único axônio, que possui a função básica de transmissão do sinal resultante do processamento de informações recebidas pelos dendritos, através de pulsos elétricos. O efeito de tais sinais é aumentar (caso o sinal for excitatório) ou diminuir (caso o sinal for de inibição) o potencial elétrico no corpo celular. Se esse potencial atingir um limiar, um pulso (potencial de ação) é enviado pelo axônio às outras células através da arborização do axônio. Os dendritos são prolongamentos que constituem a parte receptiva dos neurônios, onde o seu forma de distribuição ramificada tem a finalidade de se ter uma ampla área para a recepção dos sinais. A transmissão do sinal entre os axônios e dendritos ocorre em estruturas de contato denominada sinapse. Embora as sinapses freqüentemente ocorram entre axônios e dendritos (axodentríticas), elas podem ocorrer também entre axônios e a soma (axossomáticas), e em casos mais raros entre: axônio-axônio (axoaxônicas), dendrito-dentrito (dendrodentríticas), dendrito-soma (dendrossomáticas), soma-soma (somatossomáticas), soma-dendrito (somatodendríticas) e soma-axônio (somatoaxônicas) [MACHADO, 1993].

Como o cérebro possui bilhões de neurônios conectados, cada célula contribui com um pedaço muito pequeno da informação. Portanto, a morte de algumas células não afeta o desempenho do cérebro significativamente. Além disso, o cérebro possui um auto-aprendizado dinâmico, isso faz com que a morte de algumas células sejam compensada por outras.

5.4.2.2 - Neurônio artificial: atualmente existem vários modelos matemáticos de neurônios artificiais, os estudos foram iniciados pelo neurofisiologista McCulloch e pelo matemático Walter Pitts no ano de 1943 [McCULLOCH & PITTS, 1943]. Eles realizaram um trabalho que fazia uma analogia entre células vivas e o processo eletrônico, simulando o comportamento do neurônio natural, onde o neurônio possuía apenas uma saída, que era uma função de entrada (*threshold*) da soma do valor de suas diversas entradas (Figura 5.12), utilizando uma função de ativação *threshold* unipolar, estado de ativação binário e função de saída.

Figura 5.12 - Modelo de Neurônio artificial de McCulloc & Pitts

A estrutura básica de um neurônio artificial pode ser definida em termos de seu estado de ativação, função de ativação, função de saída e suas conexões. A diferença básica entre os diversos modelos de neurônios, concentra-se no tipo de entrada (binária ou contínua), tipo de saída, função de ativação e representação dos pesos das conexões.

Em um neurônio, o seu estado de ativação pode assumir valores binários (1 para ativo e 0 para inativo), bipolares (-1 e 1) ou valores reais. Encontramos uma grande variedade de funções de ativação ou de transferência e apresentamos a seguir as funções mais utilizadas:

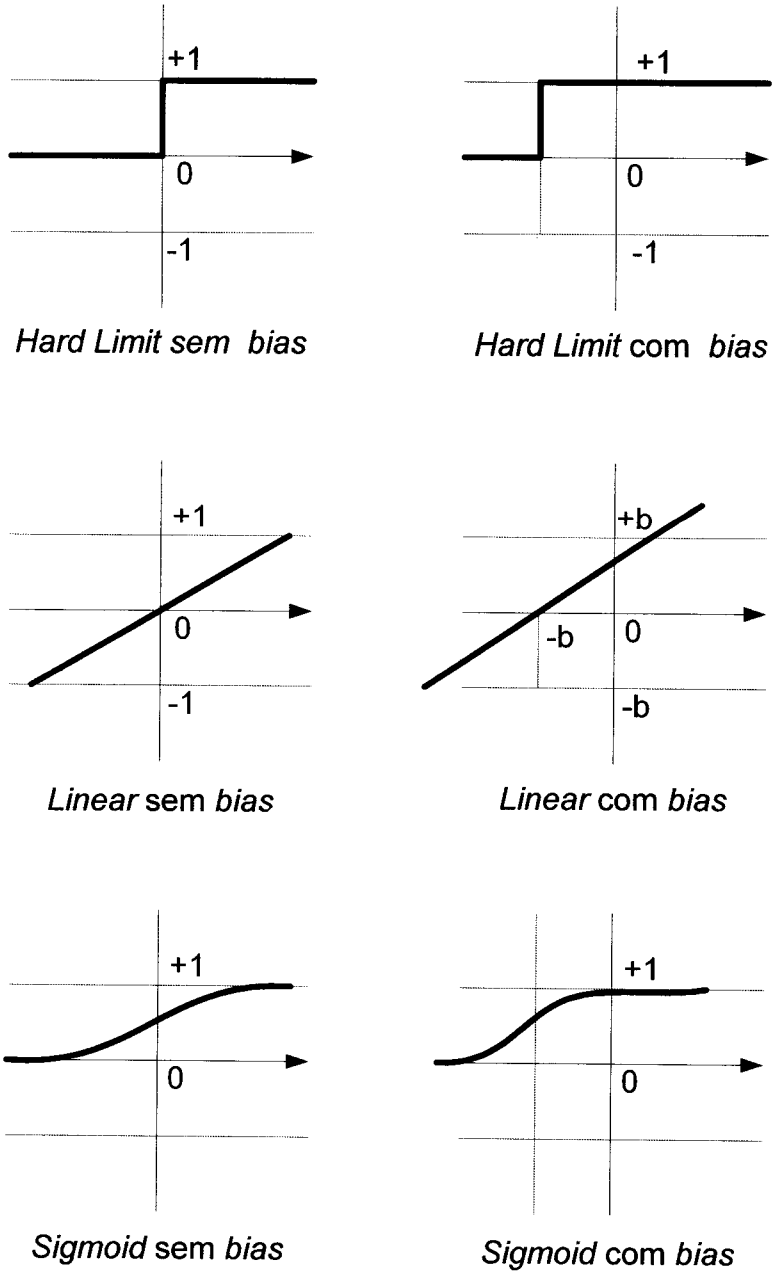


Figura 5.13 - Funções de ativação ou de transferência.

Um modelo de neurônio artificial tem a função básica de somar as entradas, que chegam através dos dendritos ou pelo corpo celular, e ativar uma saída caso essas entradas ultrapassem um certo limiar ("threshold").

A função de ativação é calculada a partir de um somatório envolvendo os estímulos de entrada de um neurônio e a intensidade das sinapses associadas a cada uma destas entradas, gerando um estado de ativação. Os estímulos de entrada de um neurônio são representados através de um vetor de entrada $X = (x_1, x_2, ..., x_p)$, e a intensidade da sinapse através de um vetor de pesos

$W = (w_1, w_2, \ldots, w_p)$. Os estímulos de entradas do neurônio são ponderados pelo vetor de pesos através da equação 5.35.

A saída do neurônio, que pode ser ativada ou não dependendo da soma das entradas atingir ou não um limiar.

As entradas do sinal são multiplicadas pelos pesos. A saída é uma função *Hard Limit* da soma das entradas, onde w_i são os pesos e Y é um *threshold* (limiar).

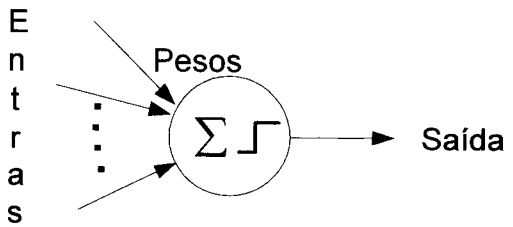


Figura 5.14 – Modelo básico do neurônio.

$$y = f_h[\sum_{i=1}^{n} w_i x_i - \theta] \tag{5.35}$$

onde: y - saída

wi - peso da i-ésima entrada

xi - i-ésima entrada

θ - threshold. É o limiar de comparação

fh – função *Hard Limit*: = 1 se x > 0

= 0 caso contrário.

A figura 5.15 mostra a função *Hard Limit*.

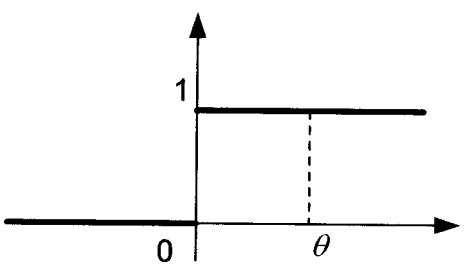


Figura 5.15 – Função *Hard Limit*

Fazendo $\theta = -w_0x_0$, a equação 9.1 pode ser rescrita da seguinte forma:

$$y = f_h[\sum_{i=0}^{n} w_i x_i] \tag{5.36}$$

O modelo de neurônio apresentado acima foi proposto por McCulloch e Pitts em 1943, e em 1962, Frank Rosenblatt deu o nome de "perceptron" a este modelo de neurônio.

- Aprendizagem : Neste modelo mais simples de perceptron no resultado temos dois níveis de saída. Se for obtida uma saída incorreta, procura-se reduzir as chances dessa saída ocorrer novamente. Se a saída for correta, nada se deve fazer.

O seguinte algoritmo para aprendizagem do perceptron:

1. Inicializar pesos e threshold aleatoriamente, fazendo $w_0=-\theta$ e $x_0 = 1$.
2. Apresentar as entradas e a saída desejada.
3. Calcular a saída por:

$$y = f_h[\sum_{i=0}^{n} w_i x_i]$$

4. Adaptar pesos:

se correto	wi(t+1) = wi(t), vá para o passo 6
se saída 0 e deveria ser 1	wi(t+1) = wi(t) + ηxi(t)
se saída 1 e deveria ser 0	wi(t+1) = wi(t) - ηxi(t)

5. Voltar ao passo 3.
6. Fim.

O η do passo 4 é uma taxa de adaptação que diminui a mudança dos pesos para que a rede não ultrapasse a solução desejada. O η é um valor inteiro positivo entre 0 e 1.

Com o *perceptron* podemos resolver vários problemas, porém este modelo é limitado quando o problema é linearmente inseparável, como no caso do OU-exclusivo. Na operação do OU-exclusivo não há maneira de separar os resultados "1" dos resultados "0" através de uma linha no espaço bi-dimensional, conforme pode ser visto na figura 5.16 abaixo.

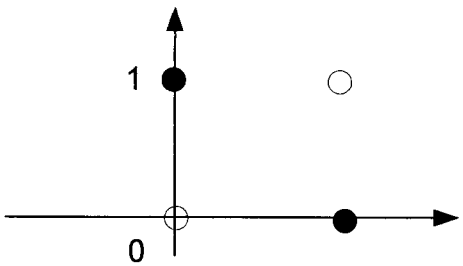


Figura 5.16 - Problema do ou-exclusivo.

5.4.3 - Classificação das redes neurais

Por causa de diferenças entre algumas ou às vezes todas as entidades envolvidas na construção de uma rede neural, diferentes estruturas de redes neurais têm sido desenvolvidas por pesquisadores, existindo vários modelos de estruturas de redes em uso. Porém, esses modelos podem ser classificados por alguns elementos básicos próprios das redes. Esses elementos são: operação das unidades de processamento ("*perceptron*"), topologia das redes, relação entre a entrada e saída da rede e o algoritmo de aprendizagem.

5.4.3.1 - Operação das unidades de processamento

Podemos classificar o neurônio quanto à natureza de seus sinais e sua função de transferência. A natureza dos sinais de entrada nos "perceptron" podem ser discretas {0,1} ou {-1, 0, 1} ou podem ser contínuas, assumindo valores nos intervalos [0, 1] ou [-1, +1] [GALLANT, 1994].

O tipo do modelo é dependente do tipo de não linearidade (função de ativação) incorporada na camada escondida [BROWN & HARRIS, 1994]. Diferentes funções de ativação podem ser implementadas: função sigmoidal (em redes de "backpropagation"), funções trigonométricas (em redes de "links" funcionais-FLN), funções gaussianas (em redes de bases radiais-RBF) e funções polinomiais (em rede de memória associativa-AMN).

5.4.3.2 - Topologia das redes

Quanto à topologia as redes são classificadas pela forma na qual as unidades básicas estão arranjadas. As estruturas podem ser classificadas nos tipos direto e recorrente [JANG & SUN, 1995].

. **Estrutura direta:** a estrutura direta é um conjunto de unidades de processamento dispostos em camadas que se interconectam da saída de uma unidade à entrada de outra na camada subsequente. Não há ligações cíclicas e nem retroalimentação (Figura 5.17(a)).

. **Estrutura recorrente:** esta estrutura apresenta retroalimentação da saída de um neurônio para outra unidade em camadas predecessoras ou possuem nós cíclicos (Figura 5.17(b)). Como exemplo desta estrutura podemos citar a rede de Hopfield.

Segundo alguns autores, existem duas outras estruturas consideradas como casos particulares da estrutura direta. Uma é a **estrutura em treliça,** que é composta de uma camada de elementos de entrada que tem o papel de passar informações para a camada de processamento da rede (Figura 5.17(c)). A outra estrutura é a **pirâmide,** na qual as unidades vão diminuindo da primeira à última camada, que possui apenas um neurônio (fig.5.17(d)).

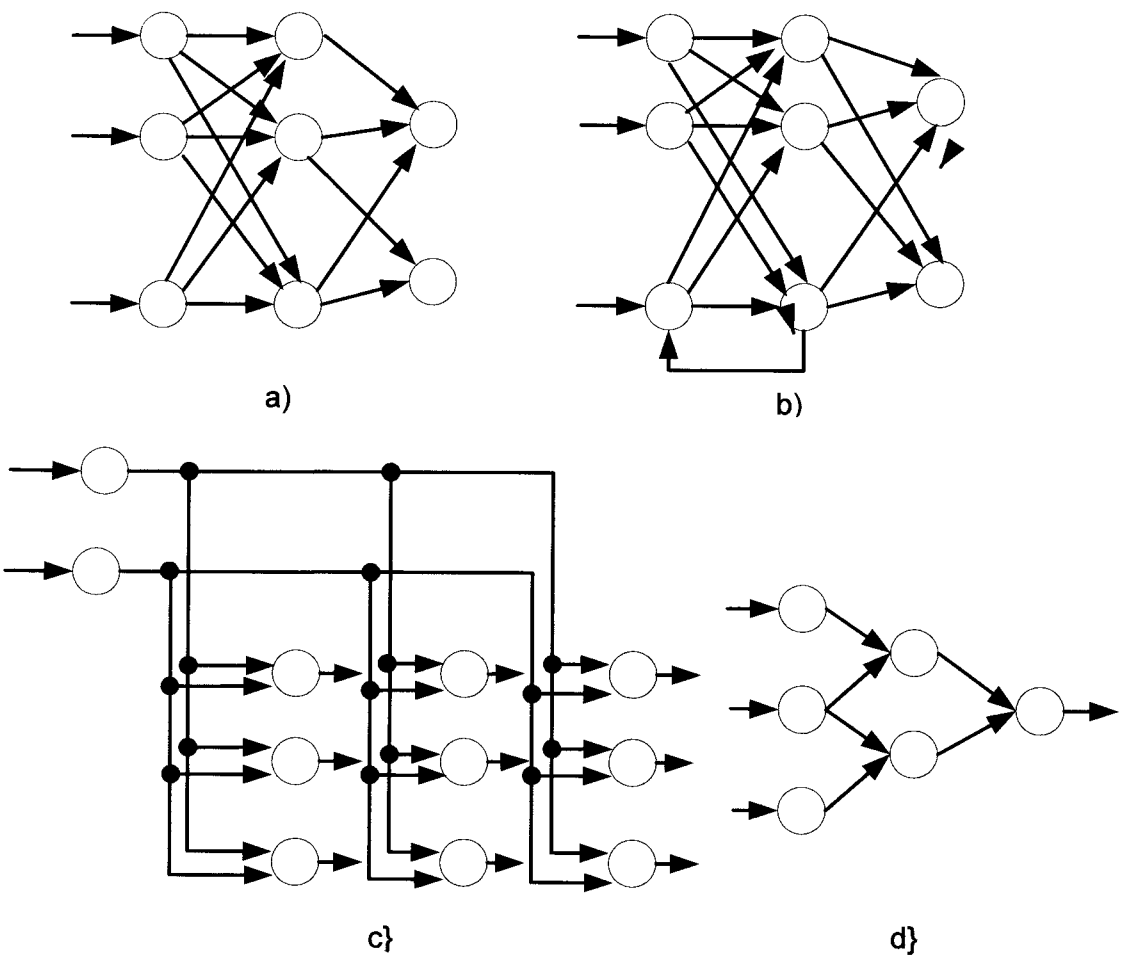


Figura 5.17 - Estruturas de redes neurais. (a) direta, (b) recorrente, (c) treliça e (d) pirâmide.

5.4.3.3 - Relação entre entrada e saída de dados

Em relação entre a entrada e saída dos dados de uma rede neural, estas podem ser classificadas como: redes autoassociativas e redes hetero-associativas.

. **Redes autoassociativas**: reconstroem a versão original de sinais de entrada corrompidos ou ruidosos [BROWN & HARRIS, 1994]. Como exemplo temos a rede Hopfield.

. **Redes hetero-associativas**: estas redes mapeam as entradas em uma representação alternativa, constituindo-se a maioria das classes de redes que tratam do problema de classificação e tarefas de aproximação funcional [BROWN & HARRIS, 1994].

5.4.3.4 -Algoritmo de aprendizagem

Uma das características mais importantes de uma rede neural é a sua habilidade de aprender a partir de seu ambiente e melhorar o seu desempenho. Esta habilidade ocorre através de um processo iterativo de ajustes aplicados aos pesos sinápticos e *thresholds*, até que a informação do ambiente seja aprendida.

Os ajustes interativos dos pesos e *thresholds* são realizados através de um algoritmo de aprendizagem seguindo algum paradigma de aprendizado. A forma interativa pela qual a rede neural se relaciona com o ambiente constitui os três paradigmas de aprendizado conhecidos, que são:

. **Aprendizado supervisionado:** os pesos são ajustados baseando-se no erro entre a resposta desejada e a resposta obtida, como orientador ou supervisor do ajuste de peso. Temos como exemplo as redes MLP (*perceptron* múltiplas camadas) com algoritmo *backpropagation,* que utilizam esta forma de aprendizado.

. **Aprendizado não supervisionado**: neste caso, o processo de aprendizagem da rede desenvolve suas próprias representações de acordo com algum critério de similaridade definido. Este treinamento só é aplicado nos casos em que não se têm informações das classes. Lin [LIN & LU, 1995] comenta que a desvantagem deste aprendizado não supervisionado é sua ineficiência. A rede da teoria de ressonância adaptativa (ART) e Hopfield são exemplos que fazem uso deste aprendizado.

. **Aprendizado por reforço**: este aprendizado é utilizado quando as informações de saída não estão disponíveis, isto é, a resposta é avaliada em termos de certo ou errado. A rede neural recebe um sinal de "*feedback*" de seu ambiente, sendo este sinal apenas avaliativo e não instrutivo. É um método de tentativa e erro. A rede não sabe, por exemplo, porque uma saída é errada e então tenta diferentes respostas até atingir a resposta certa [HERTZ et al, 1993]. O aprendizado por reforço é considerado, por alguns autores, como um tipo próprio

de aprendizado [LIN & LU, 1995]; outros o consideram um caso especial de aprendizado supervisionado [HERTZ et al, 1993].

5.4.3.5 -Taxonomia das redes neurais

Na tabela 5.1 abaixo, termos uma classificação das redes neurais mais utilizadas:

Estrutura	Aprendizagem	Redes
Direta	Supervisionado	Adaline/Madaline (fç.limiar) MLP com "*Backpropagation*" (fç.sigmoidal) FLN - (fç. Trigonométrica). RBF - (fç. Gaussiana) AMN - (fç. Polinomial) LVQ "*counterpropagation*"
	Não Supervisionado	matriz de aprendizagem *Kohonen* memória associativa
Recorren-te	Supervisionado	*Cauchy "machine"* *Boltzamn "machine"* "*Brain-state-in-a-box*"
	Não Supervisionado	*Hopfield* ART BAM "*Additive Grossberg*" (AG)

onde:

MLP – *perceptron* múltiplas camadas.

FLN - redes de "links" funcionais.

RBF - redes de função de base radial.

AMN - rede de memória associativa.

LQV - Aprendizagem do vetor de quantização.

Tabela 5.1 - Taxonomia de redes neurais

As redes de "*counterpropagation*" são consideradas como híbridas uma vez que combinam o treinamento supervisionado com o não supervisionado. A idéia geral é haver uma camada que aprende de uma forma não supervisionada, seguida de uma (ou mais) camadas que são treinadas pelo algoritmo de "*backpropagation*" [HERTZ et al, 1993].

5.4.3.6 - Escolha do modelo de rede neural

A escolha de uma rede de neural deve ser feita por aquela que apresente as características mais adequadas e apropriadas para a aplicação a ser realizada. Devemos também levar em consideração os seguintes pontos:

. **Tempo de classificação**: considerado como uma das características importantes na escolha do modelo. Verifica-se, por exemplo, que as redes diretas são muito mais rápidas que as redes recorrentes, pois estas dependem de uma realimentação dos nós posteriores gastando mais tempo para atingir o seu estado estável.

. **Treinamento:** se existirem modelos de pares de entrada-saída para treinamento, o mesmo deve ser supervisionado, uma vez que o treinamento não supervisionado pode ser ineficiente [LIN & LU, 1995].

Podemos levar em consideração outras características, não menos importantes para a escolha de um modelo, como por exemplo: o tempo de treinamento da rede, a capacidade de armazenamento, requerimento de memória, facilidade de implementação em "hardware", facilidade de programação em computadores seriais e paralelos e facilidade de se adaptar a mudanças do ambiente.

5.4.3.7 – *Perceptron* Múltiplas Camadas

O estudo do *perceptron* serviu de base para o desenvolvimento das redes multi-camadas, cuja estrutura baseia-se no conceito inicial de *perceptron*, possibilitando assim, resolver problemas mais complexos.

A rede *perceptron* múltiplas camadas apresenta geralmente a camada de entrada, a camada de saída e a camada escondida ou várias camadas escondidas entre as camadas de entrada e saída. A figura 5.18 mostra o modelo de um *perceptron* múltiplas camadas.

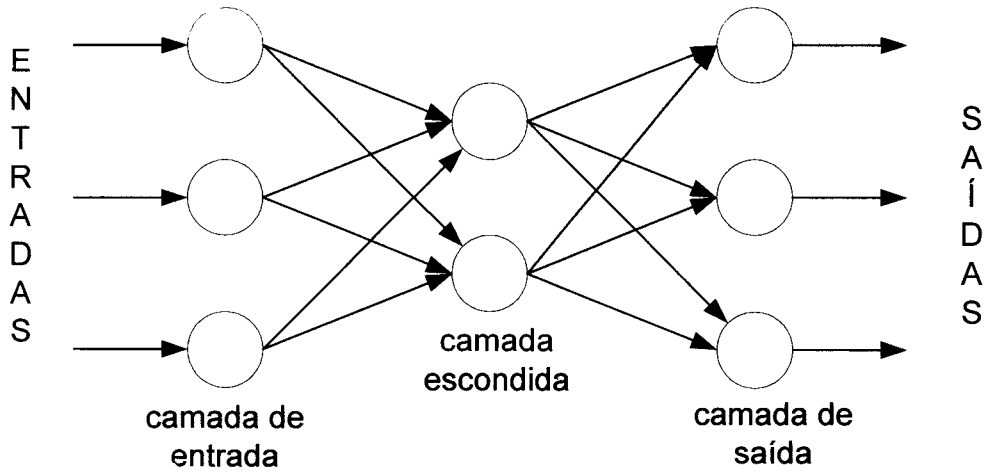
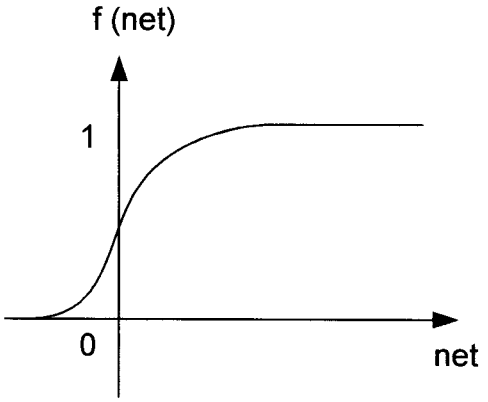


Figura 5.18 - *Perceptron* múltiplas camadas

A função *hard limit*, usada no *perceptron* simples, é substituída por uma função sigmoidal nas unidades de *perceptron* da rede. Isso se faz necessário porque a função *hard limit* remove informações da saída de uma unidade para a entrada da outra, tornando mais difícil o ajuste dos pesos da rede. A figura 5.19 mostra a função sigmoidal. Essa função possui infinitos valores entre 0 e 1, portanto, qualquer variação nos pesos modifica a saída da função, o que não ocorre com uma função *hard limit*.

A função sigmoidal é uma das mais eficientes [MASTERS, 1994]. É uma função real, limitada, diferenciável e possui sempre derivada positiva. Como na fase de aprendizagem, as redes necessitam das derivadas de suas funções de ativação; assim fica claro o porquê da escolha das redes de "*backpropagation*" pela maioria dos pesquisadores [MASTERS, 1994]. A forma da função de ativação provoca profundo efeito na sua velocidade de aprendizagem, mas não na capacidade da rede [MASTERS, 1994].



$$f(net) = \frac{1}{1 + e^{(-k.net)}}$$

Figura 5.19 – Função sigmoidal

5.4.3.8 - Aprendizado

O aprendizado tem como objetivo achar o ponto no qual a função erro ou função energia sejam mínimas (Figura 5.20).

Pode ocorrer que em um aprendizado a rede neural encontre o ponto C da figura 9.20 como a solução estável do problema, ao invés do ponto B. Isto ocorre se for considerado grande o valor do termo de ganho η, que leva a rede a dar largos passos, e consequentemente, passando o ponto C como solução ótima.

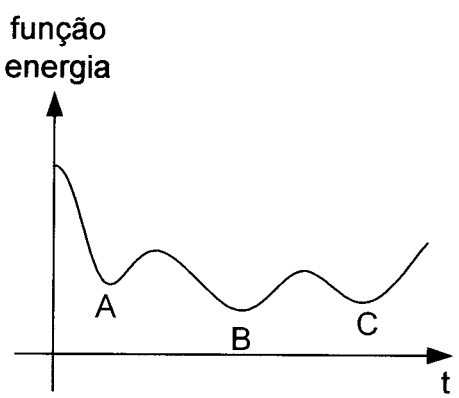


Figura 5.20 – Função de energia de uma rede neural

Quando isto ocorrer devemos reduzir o termo de ganho η e aumentar a quantidade de unidades na camada escondida, levando à função de energia ter menos pontos de mínimos [BEALE AND JACKSON, 1990], ou adicionar o termo de momento, que provocará uma convergência mais rápida da rede.

$$w_{pi}(t+1) = w_{ji}(t) + \eta\,\delta_{pj}\,O_{pi} + \alpha(w_{ji}(t) - w_{ji}(t-1)) \qquad (5.38)$$

onde α - fator de momento (de 0 a 1).

Uma vez encontrado a melhor inclinação, a convergência se torna mais lenta.

5.4.4 - RECONHECIMENTO DE PADRÕES USANDO REDES NEURAIS

Diversos modelos de redes neurais têm sido utilizados para o reconhecimento de padrões. Porém, o objetivo principal é reconhecer um objeto, levando em consideração o conceito de "Invariância Geométrica", que leva ao reconhecimento de padrões invariantes à transformação de escala, rotação e translação do padrão de entrada apresentado à rede. Na utilização deste conceito, encontramos diversas abordagens, as quais podem ser classificadas em quatro categorias principais, que são: Normalização de Padrões, Invariância pela Estrutura da Rede, Invariância pelo Treinamento, e Utilização de Atributos Invariantes.

5.4.4.1 – NORMALIZAÇÃO DE PADRÕES

Neste conceito, os padrões são apresentados de uma forma padrão ao classificador. Logo, o padrão de entrada deve ser transformado, de forma que sua posição, orientação e tamanho sejam representados sempre de mesma forma ao classificador.

Esquematicamente, podemos visualizar a implementação desta abordagem (Figura 5.21), verificando que ela ocorre através de um estágio de pré-processamento.

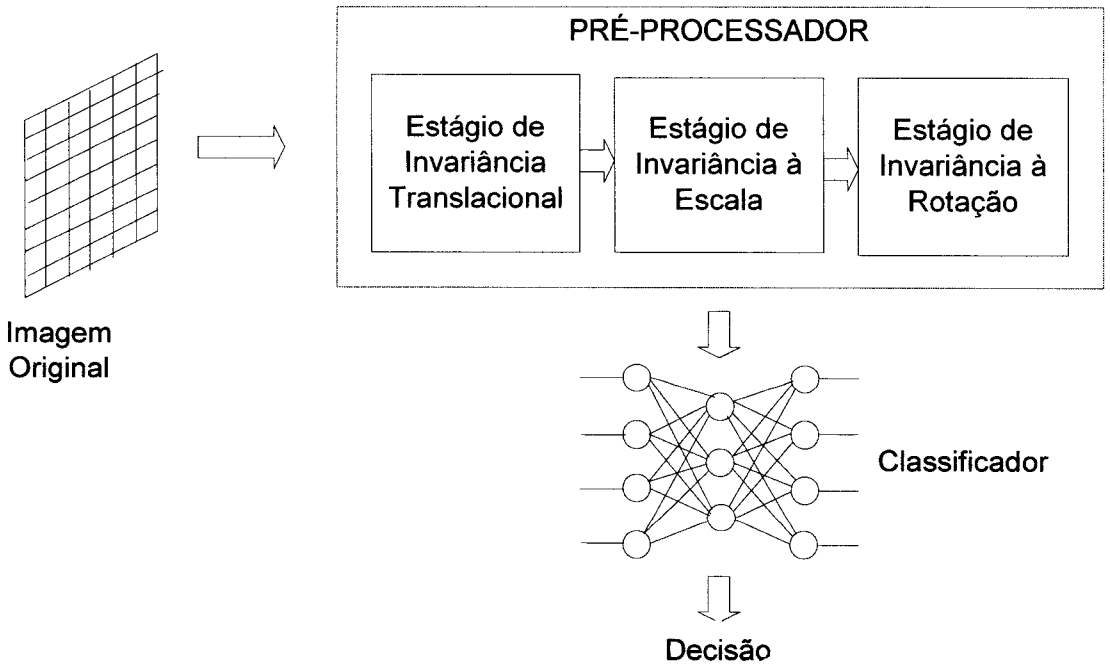


Figura 5.21 - RNA utilizando Normalização de Padrões para Reconhecimento de padrões.

Uma das técnicas de obtenção de invariância através deste conceito é com a utilização da transformada de Fourier, que normaliza o padrão em relação à posição.

5.4.4.2 - INVARIÂNCIA PELO TREINAMENTO

Na utilização desta técnica é apresentado em seqüência à rede neural, diversas vistas transformadas do padrão a ser reconhecido (Figura 5.22). O objetivo de se incluir vistas transformadas no conjunto de treinamento é para que as camadas intermediárias ou internas venham a extrair as características invariantes necessárias à generalização das mesmas pela rede.

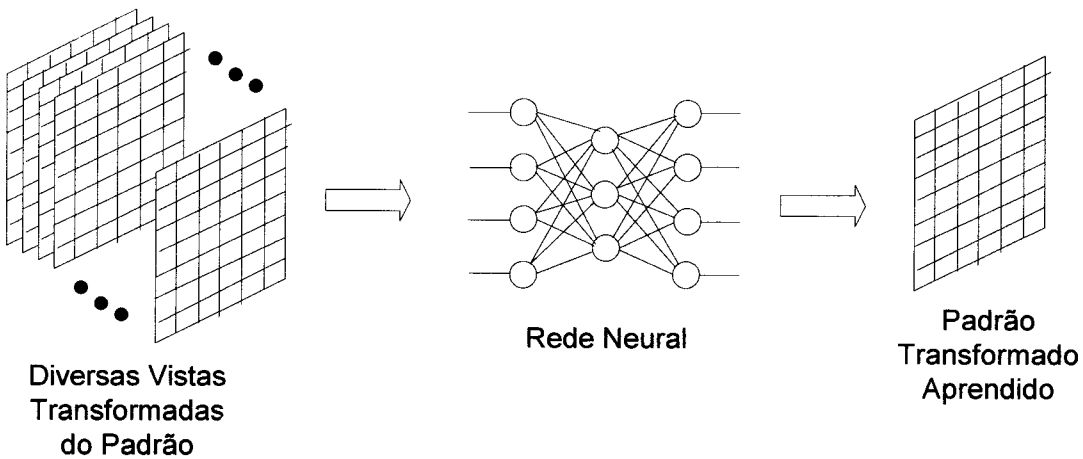


Figura 5.22 - RNA com utilização de Invariância pelo treinamento para o reconhecimento de padrões.

Para a realização deste processo é necessário um grande conjunto de treinamento, uma boa capacidade de generalização do modelo de rede neural utilizado, mas, o número de classes não deve ser muito grande. Logo, verificamos que o tempo de treinamento será proporcional ao tamanho do conjunto de treinamento, que deve envolver diversas características de cada padrão a ser reconhecido [TRUXEL et al, 1988][RUMELHART et al, 1986][ZHANG et al, 1991]. Encontramos implementações desta abordagem utilizando redes baseadas em *backpropagation*.

5.4.4.3 - UTILIZAÇÃO DE ATRIBUTOS INVARIANTES

No uso desta abordagem, após a extração de atributos invariantes, uma rede neural classificadora faz a determinação de classes dos objetos apresentados ao sistema (Figura 5.23).

Os atributos são propriedades mensuráveis dos objetos presentes na imagem e a representação de padrões através de atributos invariantes às transformações geométricas é um método de implementação do reconhecimento de padrões. Estes atributos podem ser globais (área, perímetro, momentos), locais (segmentos de reta, vértices) ou relacionais (regiões de objeto, distância), onde a escolha destes atributos pode vir a determinar o desempenho do sistema de reconhecimento de padrões [CHIN & DYER, 1986].

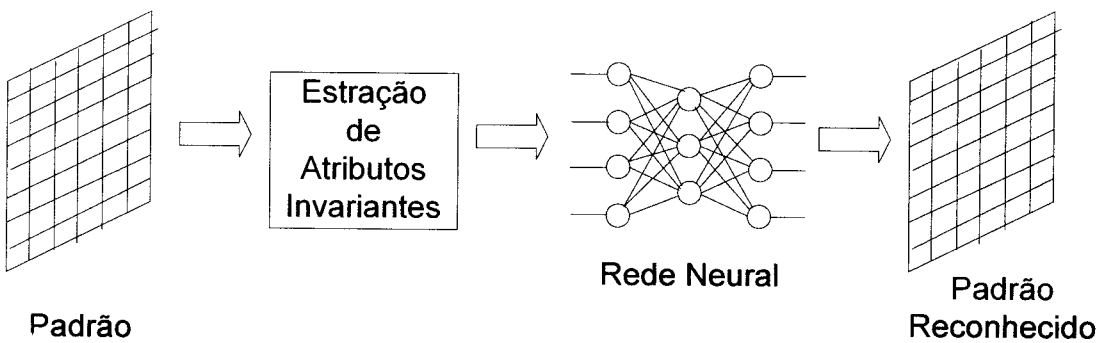


Figura 5.23 - RNA utilizando a abordagem e Atributos Invariantes para reconhecimento de padrões.

A escolha dos tipos de atributos a serem utilizados depende dos tipos de objetos ou padrões a serem reconhecidos. Idealmente um atributo invariante deveria apresentar um valor constante, independente do grau de transformação sofrido pela imagem do objeto. No entanto, na prática, diversas fontes de incerteza são acrescentadas ao modelo ideal, resultando, em uma variedade de valores medidos.

Encontramos diversos modelos de implementação de redes neurais utilizando-se de atributos invariantes para reconhecimentos de padrões, entre as quais podemos citar como as mais utilizadas as arquiteturas *Perceptron Multilayer* e *Radial Basis Function* [CHTIOUI et al, 1996][VASCONCELOS, 1995][BORGES et al, 1994][MITZIAS & MERTZIOS, 1994][VARELLA et al, 1994][WANG 7 COHEN, 1994][YAN & WU, 1994][ANAND et al, 1993][HWANG & TSENG, 1993][LEJEUNE & SHENG, 1993][GUPTA et al, 1989]. Citamos também o uso de redes neurais do tipo *counterprogation* [STEMMER, 1994], GSN[CARVALHO, 1995], Redes Neurais *Constraint Satisfaction* [TSAO et al, 1993], Redes neurais recorrentes – *Recurrent Neural Network* [WAARD, 1994], etc.

5.4.4.4 - INVARIÂNCIA PELA ESTRUTURA DA REDE

O reconhecimento de objetos, independente de sua posição no campo de entrada, ou sua rotação no plano, vem a atender o objetivo de invariância com relação à sua posição, escala e rotação. Várias técnicas vêm sendo utilizadas para obtenção deste objetivo, utilizando a abordagem de inserção - no modelo da rede - de características que forçam o modelo de rede neural a classificar vistas transformadas do objeto como sendo da mesma classe. Encontramos esta implementação em Redes Neurais de Alta Ordem (*Higher-Order-Neural networks* – HONN) [KOWALCZYK & FERRA, 1994][SPIRKOVSKA & REID, 1990][REID et al, 1989][GILES et al, 1988], redes *Neocognitron* [FUKUSHIMA, 1989; 1985 e 1983], entre outras.

Nestas aplicações, as relações conhecidas são exploradas e as invariâncias desejadas são construídas diretamente na arquitetura da rede. Desta forma a arquitetura da rede é construída de acordo com os conhecimentos de domínio, obtendo-se uma rede que é pré-treinada, dispensando a fase de treinamento referente a invariância de transformação. Assim para cada novo conjunto de objetos treinados, a rede necessita somente aprender a distinguir entre os objetos treinados, não necessitando generalizar o conceito embutido nas invariâncias. Desta forma, o tempo de treinamento é reduzido significativamente e estas redes necessitam ser treinadas em somente uma vista de cada objeto. A acuracidade de 100% no reconhecimento é garantida para imagens livres de ruído caracterizadas pelas transformações embutidas [SPIRKOVSKA & REIS, 1993].

CAPÍTULO 6

6 SISTEMA DE VISÃO COMPUTACIONAL RÁPIDA UTILIZANDO REDE NEURAL IMPLEMENTADA EM PROCESSAMENTO PARALELO

6.1 - INTRODUÇÃO

O sistema de Visão Inteligente de Alto Desempenho foi desenvolvido, para atender as características de invariância geométrica ao tratar com as imagens de peças transladadas, escalonadas e rotacionadas em relação à posição original, integrando hardware e software, obtendo Processamento em Tempo Real, utilizou processamento paralelo a nível de hardware e software.

Para atender os objetivos propostos, o sistema foi implementado, utilizando:

- Um Multicomputador Paralelo (SPP3).
- Uma câmera CCD.
- Um sistema de iluminação.
- Uma placa de captura de imagem.

6.2 - IMPLEMENTAÇÃO DO SISTEMA DE VISÃO

O sistema desenvolvido para Visão Artificial Inteligente de Alto Desempenho é constituído pelos seguintes módulos:

- Aquisição da Imagem.
- Pré-processamento.
- Extração de Características.
- Reconhecimento.
- Aplicação.

IFSC-USP SERVIÇO DE BIBLIOTECA INFORMAÇÃO

6.2.1 - AQUISIÇÃO DA IMAGEM

6.2.1.1 - ILUMINAÇÃO

Em um sistema de visão de máquina, a iluminação adequada é a que vai nos permitir obter uma imagem com qualidade de um objeto ou cena. A imagem, como será descrita nos tópicos seguintes, é formada por pixels. A intensidade de cada pixel é a produzida por duas componentes, a iluminação e a refletância. A iluminação é a soma das fontes de luz presentes no objeto, e a refletância é a intensidade de luz refletida do objeto.

Então, uma imagem depende, incontestavelmente, da ação da luz ou do sistema de iluminação. Para que se possa ver uma cena ou objeto, é necessário que raios luminosos provenientes de uma fonte de luz incidam sobre ele e sejam refletidos à câmera (ou sensores), tornando a cena ou o objeto uma fonte de luz.

A escolha de um método de iluminação de uma cena ou objeto será determinada pela natureza dos objetos escolhidos, e pelo efeito desejado. Uma composição brilhante e repleta de detalhes delicados, por exemplo, necessita de uma iluminação frontal suave, capaz de evitar a formação de reflexos indesejáveis; por outro lado, objetos opacos com formas e texturas interessantes exigiriam uma luz mais dirigida, a fim de destacar estas qualidades.

Para se obter uma boa técnica de iluminação, é necessária a realização de vários testes, obedecendo aos seguintes critérios:

Primeiro, deve-se deslocar as luzes de modo a descobrir como suas diferentes posições alteram a aparência do objeto ou cena. Não se deve acender, de início, todas as luzes, e estas não devem ser escolhidas e colocadas ao acaso. Inicialmente, deve-se ligar a luz principal. Depois disso, coloca-se as luzes de efeito, necessárias para elevar a tonalidade de algum detalhe importante e, por último, a luz de fundo.

Para se melhorar a iluminação, pode-se utilizar um ou mais refletores, os quais podem ser constituídos por qualquer superfície branca, desde uma folha de papel até uma tela para projeção de slides. Um refletor prático é feito de uma placa de fibra, tipo Eucatex ou Duratex, pintada de branco, branco-mate ou prata. O efeito de tal refletor pode ser bastante surpreendente se colocado bem perto do objeto e do lado oposto à luz principal, o qual pode aliviar consideravelmente as sombras e até eliminar a necessidade de uma luz auxiliar. Logo, ao invés de direcionarmos uma fonte de luz diretamente para a cena ou objeto, podemos refleti-la numa superfície, para obtermos uma luz mais difusa e suave, utilizando-se de um refletor tipo guarda chuva, um refletor plano ou uma caixa de luz com tela suavizadora de material semitransparente, para suavizar a luz e distribuí-la de forma homogênea sobre toda a área.

Para direcionar a luz utilizamos projetores e espelhos. Os projetores possuem uma grande lâmpada opalescente para espalhar e difundir a luz. Quanto mais largo o refletor

(floodlight), menos intensa a luz e mais suaves as sombras. Também pode-se trabalhar em refletores menores produzindo uma luz mais dura (spotlight). Também pode ser utilizados refletores que refletem a sua própria luz, os quais proporcionam sombras mais suaves. Utilizando-se de lâmpadas halogénas, de tungsténio é possível obter uma iluminação eficiente com um equilíbrio de cor constante, quando utilizada em projetores de luz difusa para dar um efeito contrastado e direcional.

Um pequeno espelho pode, às vezes, substituir o dispendioso spotlight (luz direcional, luz dura), capturando uma parte da luz principal e redirigindo-a de acordo com as necessidades.

Em um sistema de iluminação pode-se constatar que:

- Uma luz dura e frontal é capaz de dar uma idéia das formas planas dos objetos.
- A luz fortemente dirigida, suavizada por uma tela, evidencia a forma espacial e textura.
- A colocação de um refletor do outro lado dos objetos atenua as sombras densas, fortalecendo a forma espacial.
- A utilização de um espelho no lugar de um refletor obtém-se uma iluminação mais dirigida como se fosse um spotlight em miniatura.
- Quando os objetos são iluminados diretamente por trás, com uma tela, tanto a forma espacial quanto a textura são evidenciadas.
- Combinando-se o uso da luz refletida com a direta, obtemos uma evidência da forma espacial e, assim, introduzimos a profundidade.

Um caso especial é a iluminação de superfícies refletoras em excesso, como a prata, o latão, o cromo, o alumínio e aço polido, etc.

As superfícies brilhantes dos objetos metálicos muito polidos acentuam as altas luzes e adquirem reflexos. Para evitar de forma eficaz este problema, devemos utilizar uma cúpula de papel ou tecido, translúcida e, se possível, sem costuras, colocada de forma a circundar por completo o objeto. A cúpula apresenta duas finalidades básicas: atua como superfície refletora para produzir uma iluminação suave, destinada a evitar um excesso de altas luzes, e impede também que o objeto apresente reflexos de outros, situados nas proximidades.

Quando o objeto se compõe principalmente de superfícies planas, às vezes é dispensado o uso da cúpula e é colocado apenas refletores em ângulos em relação às superfícies para rebater a luz contra elas.

Quanto ao posicionamento das fontes de luz, temos os seguintes tipos de iluminação:

- Iluminação lateral: uma fonte luminosa a noventa graus sobre um objeto

proporcionará um contraste entre o lado iluminado e o não iluminado.

- Iluminação vertical: colocando uma luz diretamente sobre o objeto, teremos pouca sensação de profundidade, pois a iluminação vertical destrói o volume.
- Iluminação oblíqua: com fonte colocada a um ângulo de aproximadamente quarenta e cinco graus será obtida uma idéia de volume do objeto, e quanto suave revela a sua textura.
- Iluminação de fundo: é utilizada para realçar o contorno do objeto.
- Iluminação de efeito: é quando se coloca uma fonte de luz mais afastada que a fonte de luz principal para iluminar as áreas de penumbra.

Neste trabalho, a imagem é capturada através da utilização de uma câmera CCD, ligada a uma placa de captura de vídeo com capacidade de digitalizar imagens a um ritmo de trinta quadros por segundo. Para minimizar a perda de definição nas bordas do objeto, foi utilizado um sistema de iluminação adequado à aplicação, para proporcionar um alto contraste e minimizar as sombras e ruídos, o qual deve ser bem projetado para cada caso, para se obter melhores resultados. Foi necessário efetuar um grande número de imagens capturadas para se formar classes de objetos a serem reconhecidos, pois quanto maior é o número de dados de treinamento para uma rede neural, melhor é o seu desempenho.

6.2.1.1.1 - ILUMINAÇÃO EM SISTEMAS PARA VISÃO DE MÁQUINA

Analisando as considerações descritas nos parágrafos do item anterior, e transportando para aplicação em sistemas de visão de máquina, temos que:

. Em sistemas de visão de máquina devemos utilizar técnicas especiais de iluminação para se obter uma imagem com bom contraste, a fim de ser processada.

. Para se ter uma visão de máquina eficiente, é essencial que se tenha uma iluminação adequada.

. Um bom sistema de iluminação proporcionará uma melhor qualidade dos dados de entrada, possibilitando, assim, obter um melhor desempenho dos programas a serem utilizados para análise da imagem. Utilizando-se de uma técnica adequada para iluminação esta proporcionará um alto contraste bem como minimizará as reflexões

especulares e as sombras do objeto visualizado destacando as características relevantes do mesmo.

Em visão de máquina, os dispositivos ou recursos de iluminação podem ser reunidos nas seguintes categorias:

1. Dispositivo de difusão de superfície. Como exemplos desses dispositivos temos as lâmpadas fluorescentes típicas e mesas de luz.
2. Projetores condensadores. Utilizados para transformar uma fonte de luz expansiva em uma fonte de luz condensada.
3. Projetores difusos ou pontuais de luz. Esses são usados para iluminar determinadas áreas da superfície.
4. Colimadores. Os colimadores servem para proporcionar um feixe de luz paralelo de luz sobre o objeto.
5. Dispositivos de imagem. São exemplos desses dispositivos os projetores de slides e amplificadores óticos, os quais formam uma imagem do alvo no plano do objeto.

Os dispositivos acima e os efeitos causados pelos tipos ou posições das fontes de luz, com a finalidade de dirigir o feixe de luz para iluminação, podem ser reunidos a fim de formar algumas técnicas utilizadas para a captura de uma imagem em sistemas de visão de máquina, conforme pode ser vista na tabela 10.1 a seguir:

Técnica	Função/uso
1. Fonte de luz frontal	
a. Iluminação frontal	Área inundada de luz de modo que a superfície defina a característica da imagem
b. Iluminação especular (campo escuro)	Usada para reconhecimento dos defeitos da superfície. Utiliza-se fundo escuro
c. Iluminação especular (campo iluminado)	Usada para reconhecimento dos defeitos da superfície, câmera em linha com os raios refletidos. Utiliza-se fundo Iluminado.
d. Imagem frontal	Aplicações em luz estruturada, luz com imagens superpostas sobre a superfície do objeto, onde o feixe de luz está deslocado em função da espessura.
2. Fonte de luz traseira	
a. Iluminação traseira (campo iluminado)	Usa um difusor de superfície para delinear características; usado na inspeção de peças e medições básicas.

b. Iluminação traseira (condensador)	Produz imagens de altos contrastes; útil para aplicação com alta ampliação.
c. Iluminação traseira (colimador)	Produz uma fonte com raios de luz paralelos de modo que as características do objeto não se encontram no mesmo plano.
d. Iluminação traseira Deslocada	Útil para produzir relevo das características quando a característica está em meio transparente.
3. Outros dispositivos	
a. Divisor de feixes	Transmite luz ao longo do mesmo eixo ótico que o sensor; a vantagem é que pode iluminar objetos de difícil observação.
b. Espelho dividido	Semelhante ao divisor de feixes, porém mais eficiente e com menores requisitos de intensidade de luz.
c. Redirecionadores não seletivos	A fonte de luz é redirecionada para proporcionar iluminação adequada.
d. Retrorefletor	Um dispositivo que redireciona os raios incidentes de volta para o sensor; o ângulo de incidência pode ser variado; proporciona um alto contraste para o objeto entre a fonte e o refletor.
e. Dupla densidade	Uma técnica usada para aumentar a intensidade de iluminação no sensor; usada com meios transparentes e retrorefletor.

Tabela 6.1 - Técnicas de Iluminação.

Da tabela acima podemos destacar duas técnicas de iluminação usadas em visão de máquina que são: iluminação frontal e iluminação traseira. Na iluminação frontal é utilizada uma fonte de luz no mesmo lado da cena e câmera. Nesta técnica, a luz refletida é utilizada para criar a imagem na câmera. Dentro desta técnica pode-se utilizar a iluminação direcional para inspeção da superfície de objetos, que pode explorar a colocação da fonte de luz e do sensor de imagem ou câmera, sob certos ângulos de incidência e reflexão em relação à superfície do objeto. Na iluminação traseira ou de fundo o objeto está colocado entre a fonte de luz e a câmera, sendo esta utilizada quando se pretende obter contornos, furos ou medidas relevantes, porem esta técnica não fornece características de volume do objeto.

Podemos citar também a iluminação estruturada, a qual consiste em projetar pontos, linhas ou uma grade sobre o objeto, que através de análise, podem-se extrair características tridimensionais do objeto.

A seguir temos na figura 6.1 o sistema de aquisição e iluminação para aquisição das imagens utilizadas nesta tese.

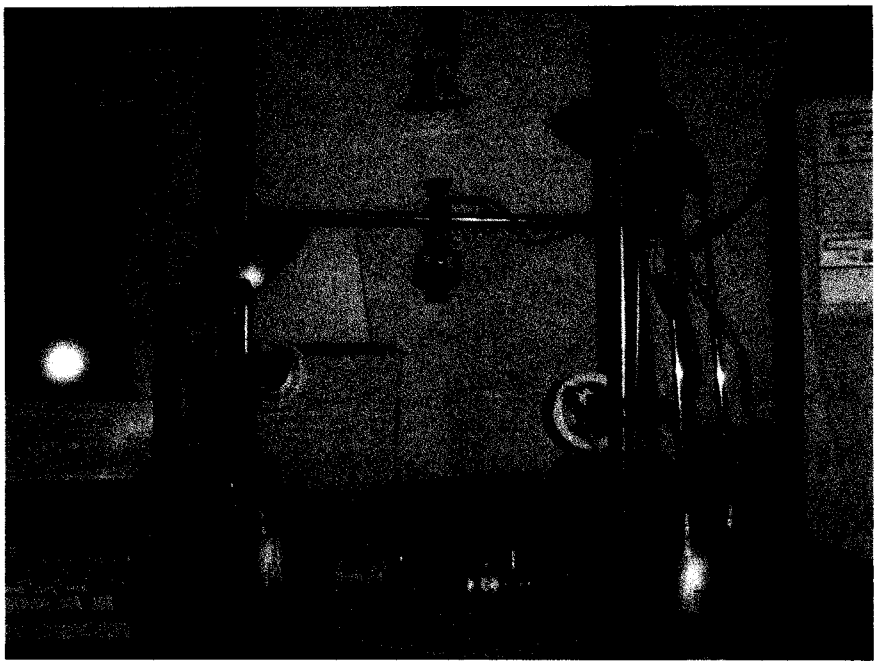


Fig. 6.1 - Sistema de aquisição/iluminação de imagens utilizado nesta tese

No desenvolvimento do projeto da presente tese foi explorada ao máximo as técnicas de iluminação, porém, a técnica de iluminação refletida (difusão e direcional) foi a mais utilizada, por possibilitar a extração de características da superfície e tonalidade, sendo que esta última característica possibilita o estudo da graduação em níveis de cinza do objeto. Também me preocupei em analisar a forma e volume, pois a forma descreve o contorno bidimensional de um objeto, permitindo que este seja reconhecível mesmo quando colocado em contra luz sem apresentar pormenores de superfície; o volume ou modelação descreve não apenas as características superficiais de um objeto, mas também as suas qualidades tridimensionais.

6.2.1.2 - FOCALIZAÇÃO DA IMAGEM

6.2.1.2.1 - LENTE

A seguir temos as informações básicas, a serem utilizadas para se obter com nitidez uma imagem ou cena de um objeto, a ser adquirido. Isso ocorre através do uso correto de uma lente ou objetiva, as quais foram seguidas e obedecidas para o bom desenvolvimento desta tese.

6.2.1.2.2 - DISTÂNCIA FOCAL (F) E ABERTURA DA LENTE (f)

Para controlar a intensidade de luz que passa pela lente, utilizamos uma parte mecânica chamada íris. Para se trabalhar com intensidade fraca de luz exige-se um diâmetro maior; menor será o diâmetro quando a intensidade de luz for maior. O diâmetro de abertura do diafragma íris é controlado por um anel que fica em torno da armação da lente e que é chamado de anel do diafragma ou anel de abertura da lente. O controle de abertura da lente é adquirido para ser utilizado de forma manual, automático ou fixo.

Os números que normalmente (ou quase sempre) aparecem gravados no anel de abertura da lente são conhecidos universalmente como números **f**. Este "**f**" é minúsculo, o qual refere-se à abertura da lente, enquanto que para a distância focal utiliza-se a letra "**F**" maiúsculo.

Os números **f** normalmente são encontrados na seguinte seqüência:

1.8 2.8 4 5.6 8 11 16 , aparecendo às vezes até 22 ou

1.4 1.5 1.7 1.9 ou até 2

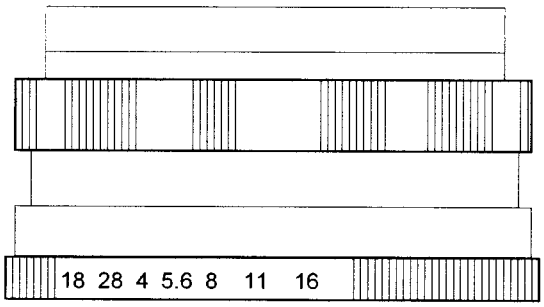


Figura 6.2 - Marcações típicas de abertura da lente no anel de abertura da lente

Conforme pode ser visto na figura abaixo, verificamos que quanto maior o número **f**, menor será a abertura associada a ele. Isto ocorre porque o número **f** está associado à relação entre a distância focal da lente e o diâmetro de abertura da lente (abertura do diafragma da íris).

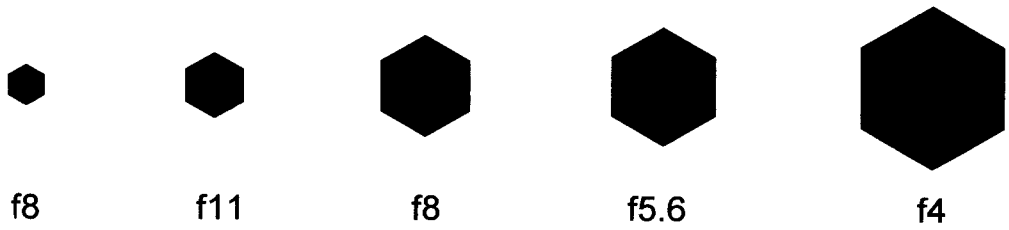


Figura 6.3 - O tamanho da abertura do diafragma íris para diferentes regulagens da lente.

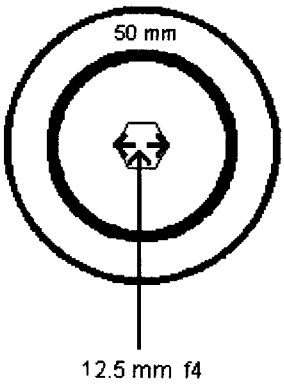


Figura 6.4 - Com a abertura ajustada em f/4, o diâmetro da abertura do diafragma íris de uma objetiva de 50 mm mede 12,5 mm.

A figura 6.4 acima mostra que se utilizarmos uma lente com distância focal 50 mm e pudéssemos medir o diâmetro da abertura do diafragma quando o anel de abertura da lente estivesse posicionado em 4, encontraríamos para ele o valor de 12,5 mm. Assim, confirmando a relação entre distância focal e diâmetro da abertura da lente, se dividirmos 50 por 12,5, obtemos o 4 que corresponde ao número **f**. Logo, o número 4 no anel de abertura indica que naquele ponto a distância focal da lente equivale a quatro vezes o diâmetro da abertura ou, inversamente, que o diâmetro da abertura é um quarto da distância focal.

As lentes trazem gravados números que exprimem a relação entre o diâmetro da abertura e a distância focal da lente. Essa relação é constante para todas as lentes.

A título de entendimento, podemos observar que quando o diâmetro da abertura do diafragma da íris da lente é 1/5.6 (f/5.6) da distância focal, a quantidade de luz que passa pela lente é a metade da que entra quando a abertura é 1/4 (f/4) da distância focal.

A superfície ou plano em que os raios de luz que atravessam a lente convergem para formar uma imagem nítida é chamada de plano focal. Neste plano focal de uma câmera estão localizados os sensores de imagem.

O comportamento dos raios de luz que passam por uma lente, é tal que os raios provenientes de um objeto próximo e de outro distante não convergem num mesmo plano.

Para que os objetos próximos possam aparecer focalizados, é necessário que a distância entre a lente e o sensor (plano focal) se torne maior do que a usada para focalização de objetos distantes. Quanto mais próximo o objeto, maior terá que ser esta distância. Logo, para que objetos próximos sejam focalizados com nitidez, deve-se aumentar a distância entre a lente e o sensor.

O anel de focalização é um anel normalmente estriado que funciona através de um mecanismo semelhante ao de uma rosca. Ao girá-lo, a lente se movimenta para dentro ou para

fora, conforme o sentido de giro. Quanto mais próximo da câmera estiver o objeto, mais se deverá girar o anel de focalização num sentido a fim de aumentar a distância entre a lente e o sensor, a fim de focalizar o objeto com nitidez. Para focalizar objetos localizados a distâncias cada vez maiores, deve-se girar o anel de focalização na direção oposta a fim de diminuir a distância entre a lente e o sensor, até que o anel de focalização não possa mais ser girado. A partir deste ponto, nota-se que todos os objetos distantes estarão focalizados independente da distância em que se localizam.

Este ponto a partir do qual todos os objetos estão focalizados, por maior que seja a sua distância da câmera, é chamado de foco infinito ou simplesmente infinito.

Na câmera, do lado do anel de focalização, encontramos uma seta ou marca (losango ou ponto) colocada à frente de uma escala de distâncias que se compõe de duas seqüências de números. Quando o anel de focalização é girado, os números, que compõem essa escala passam pela marca. As duas seqüências de números correspondem à distância em que a lente está focalizada, medida em metros (m) e pés (ft).

Ambas as seqüências, ou escalas, se iniciam com a lente focalizada no infinito, o qual é representado por um número oito (8) deitado: "∞". A medida, em que, se gira o anel de focalização, os números se tornam progressivamente menores, indicando uma aproximação cada vez maior entre o ponto no qual a lente está focalizada e a câmera. Uma escala típica em pés apresentaria, em ordem decrescente, da seguinte forma:

∞ 30 15 10 7 5 4 3.5 3 2.5 2.25 2 1.75

E abaixo desta, uma escala típica em metros apresentaria, em ordem decrescente, a seguinte seqüência:

∞ 10 5 3 2 1.5 1.2 1.0 0.9 0.8 0.7 0.65 0.6 0.55

A título de exemplo temos abaixo a figura 6.5 que é a representação de uma objetiva, com a escala de distância.

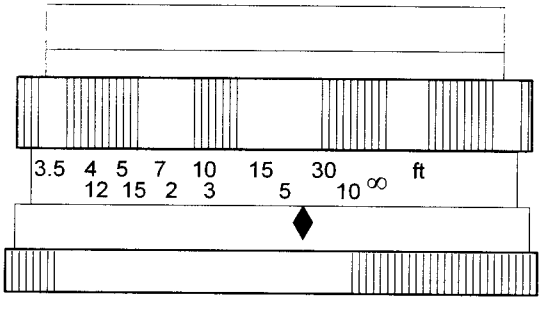


Figura 6.5 - Escala de distância em pés (ft) e metros (m). O losango indica a distância em que a câmera está focalizada.

Quando a lente estiver focalizada, com o losango (ou ponto) na posição ∞ , a distância entre o sensor e a lente será a menor distância necessária para que a maioria dos objetos próximos e distantes fique nítidos. Essa distância mínima entre a lente e o sensor, define o que é chamado de distância focal da lente.

A distância focal das lentes das câmeras é sempre dada em milímetros (3,6 mm, 6 mm, 8mm, 12 mm, etc.)

6.2.1.2.3 - ÂNGULO DE VISÃO OU DE COBERTURA

Uma mudança na distância focal equivale a uma mudança no ângulo de visão: quanto menor a distância focal, maior o ângulo. A seguir, na tabela, é apresentado o ângulo de visão aproximado correspondente à distância focal.

Distância focal (mm)	Ângulo de visão (graus)
220	
8	180
15	110
18	100
28	74
35	62
50	46
85	28
135	18
200	12
300	8
600	4
1200	2

6.2.1.2.4 - PROFUNDIDADE DE CAMPO

Quando se focaliza a lente, por exemplo, a um metro, todos os objetos focalizados a um metro de distância da lente são reproduzidos nitidamente. Os objetos mais afastados ou mais próximos da lente aparecem menos nitidamente e o grau de nitidez cai gradualmente à medida que aumenta a distância em relação ao plano focalizado.

O termo "Campo" se refere à parte da imagem na qual os objetos aparecem definidos com nitidez e a palavra "Profundidade" corresponde à extensão deste campo. Logo, "Profundidade de Campo" é a distância total entre o ponto mais próximo e o mais afastado da cena em frente da lente e que aparecem reproduzidos nitidamente na imagem.

Uma maneira prática, para se calcular o valor aproximado da profundidade de campo é dada pela formula:

$$\mathbf{Dp} = \frac{\mathbf{D}}{\mathbf{1 + D.c.f/F^2}} \quad \mathbf{e} \quad \mathbf{Da} = \frac{\mathbf{D}}{\mathbf{1 - D.c.f/F^2}} \tag{6.9}$$

onde:

Dp = limite próximo da profundidade de campo

Da = limite afastado da profundidade de campo

D = distância focalizada.

c = círculo de confusão limite.

f = número f ou abertura da lente.

F = distância focal da lente.

Das variáveis acima, temos que círculo de confusão, teoricamente, é a distância em que uma lente corretamente focalizada reproduz um ponto como um ponto, o que na prática isto não é conseguido, mas quando uma lente não se acha corretamente focalizada, diz-se que é reproduzido o ponto como sendo um disco. No entanto, temos que uma imagem parecerá nítida, contanto que a lente reproduza os pontos como discos que não ultrapassem determinado tamanho, o qual é chamado de círculo de confusão limite. Logo, as tabelas e fórmulas de profundidade de campo funcionam baseadas no princípio de que o tamanho do círculo de confusão pode ser previsto com precisão.

A seguir, é apresentado o efeito na profundidade de campo causado pela alteração do tamanho da abertura da lente, distância do objeto à câmera e da distância focal.

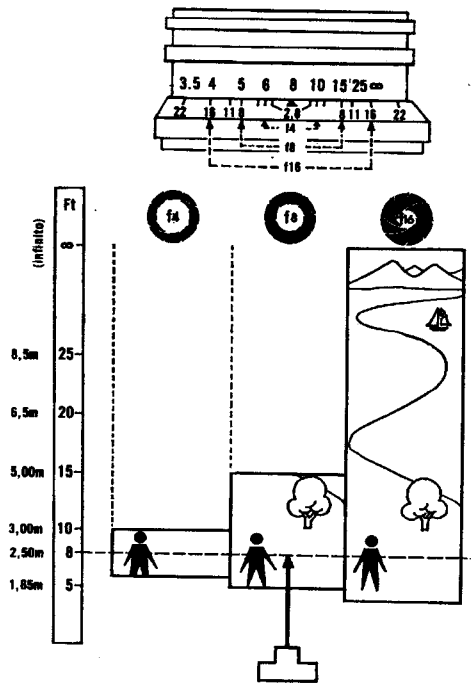


Figura 6.6 - Profundidade de campo, causado pela alteração do tamanho da abertura da lente, distância do objeto à câmera e da distância focal

A figura 6.7 mostra a profundidade de campo, do plano próximo ao plano distante, conforme a abertura da lente.

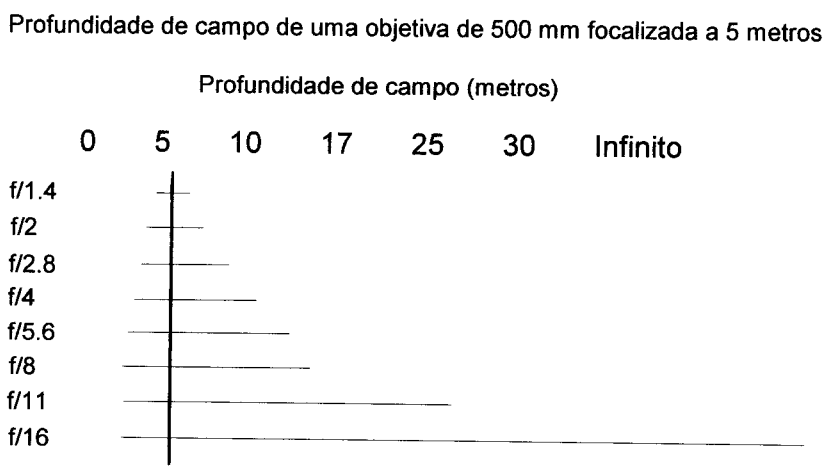


Figura 6.7 - Profundidade de campo de uma objetiva focalizada a distância fixa.

Observando a figura 6.7, verificamos duas características importantes da profundidade de campo:

1. A profundidade de campo é maior em direção ao lado mais afastado do plano em que a câmera está focalizada do que do lado mais próximo. Nota-se que o deslocamento estende-se para frente em relação ao plano na pro porção de um terço e para trás deste na proporção de dois terços.
2. Quanto menor o diâmetro da abertura da lente, maior é o aumento da profundidade de campo em direção ao lado mais afastado.

A seguir temos o exemplo de um gráfico onde a profundidade de campo varia em função da variação da distância em que uma câmera é focalizada, mantendo-se a abertura fixa.

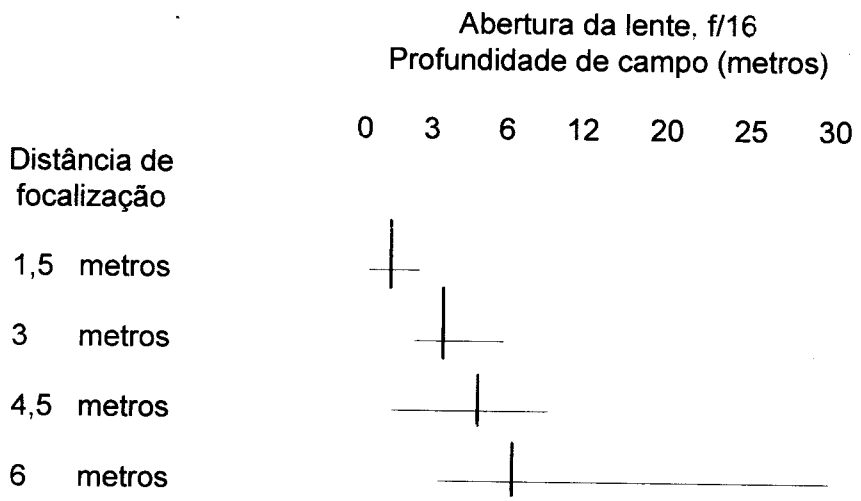


Figura 6.8 - Profundidade de campo em função da variação da distância em que a câmera é focalizada.

Da figura 6.8, podemos extrair as seguintes características da profundidade de campo:

1. Quanto menor a distância de focalização da câmera, menor será a profundidade de campo.
2. Á medida em que a distância de focalização é aumentada, a profundidade de campo aumenta também até que os objetos focalizados no infinito apareçam todos focalizados.

Por último, podemos observar, na tabela abaixo, a variação da profundidade de campo, quando variamos a distância focal (ou de objetiva), mantendo a mesma distância do objeto e a mesma abertura da lente

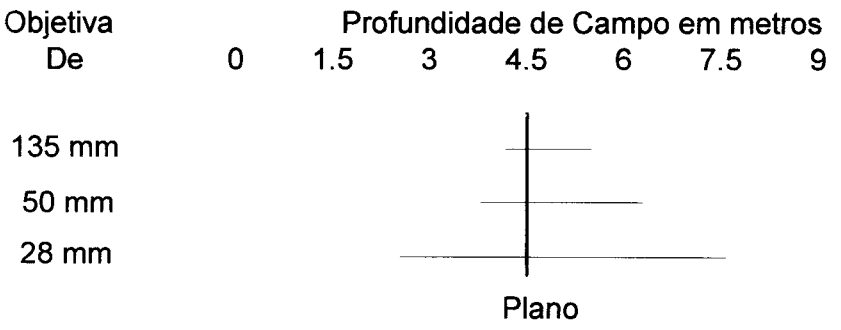


Figura 6.9 - Profundidade de campo quando varia a distância focal

Podemos observar na figura 6.9 mais duas características em relação à profundidade de campo:

1. Para uma mesma abertura de lente, quanto maior for a distância focal, menor será a profundidade de campo.
2. maior será a profundidade de campo para uma mesma abertura de lente, quanto menor for a distância focal.

6.2.1.2.4.1 - CONTROLE DA PROFUNDIDADE DE CAMPO

Observando uma cena ou objetos através de uma câmera e fechando gradativamente o diafragma íris até chegar às menores aberturas da lente, podemos estimar a profundidade de campo, analisando a nitidez da área de interesse da imagem. Entretanto, através deste método, temos a desvantagem de que com a diminuição da abertura do diafragma íris, reduz a quantidade de luz que atravessa a objetiva, o que a prejudica a nitidez, a menos que a cena ou objetos estejam muito bem iluminados.

Uma outra forma de se estimar a profundidade de campo é a utilização da escala de profundidade de campo existente na objetiva da câmera. A escala de profundidade de campo se compõe de uma série de linhas finas que se distribuem à esquerda e à direita de uma marca (ponto, losango, ou seta) central, de acordo com as quais as distâncias de focalização, em metros ou pés, são lidas. Essas linhas correspondem às regulagens dos números f da objetiva. Elas são

utilizadas para indicar a distância entre os planos próximos e distantes da zona de profundidade de campo com a câmera focalizada numa distância qualquer dada (fig. 6.10).

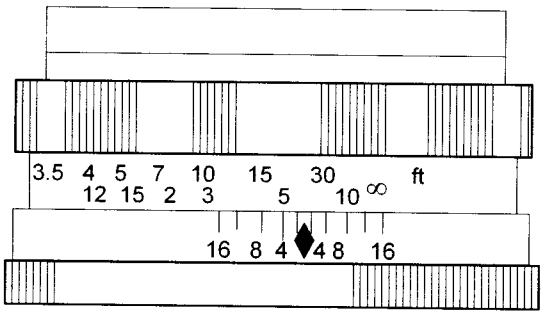


Figura 6.10 - Escala típica de profundidade de campo.

Em alguns casos esta escala numérica é substituída por cores, sendo a abertura da lente indicada pela cor correspondente como: f/16, azul; f/11, amarelo; f/8, rosa; f/5.6, branco; f/1.4, verde.

Para um melhor entendimento deste método, façamos uma análise da fig. 6.11, abaixo:

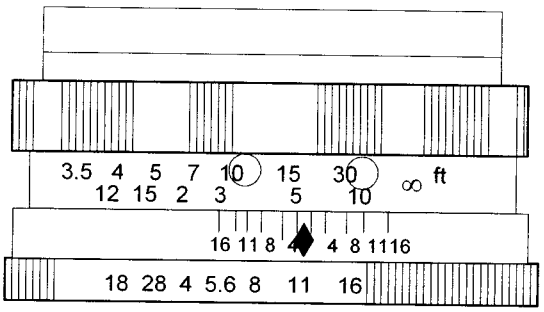


Figura 6.11 - Leitura da profundidade de campo com abertura f/11 quando a objetiva da câmera está focalizada a 5 m.

Com a seta apontando para a marca de 5 m na escala métrica, podemos verificar que as marcas 3 e 10 m se encontram em frente a f/11, uma em cada lado da seta. Logo, temos que se a objetiva estiver focalizada em um objeto a 5 m da câmera, tudo o que estiver entre 3 e 10 m estará na zona de profundidade de campo. Da análise, temos que, tudo o que estiver entre 3 e 10 m, estará focalizado.

Por último, poderemos obter a profundidade de campo através do uso de tabela fornecida pelo fabricante junto com a objetiva. Estas tabelas fornecem números bem precisos sobre o plano próximo e o plano distante de acordo com algumas distâncias de focalização, com

precisão de milímetros para distâncias de 50cm ao infinito. No entanto, assim como ocorre com a escala de profundidade de campo que está impressa na objetiva, para valores não constantes, a profundidade de campo terá que ser estimada.

Para cada abertura da lente, há uma distância em que a câmera pode ser focalizada, de modo que o plano distante esteja no infinito e o plano próximo esteja em uma distância mínima possível da câmera. Neste caso, obtêm-se a profundidade de campo máxima que cada abertura pode proporcionar. Estas distâncias são conhecidas como distâncias hiperfocais. Se um objeto for focalizado numa distância menor, os objetos localizados no infinito ficarão desfocados. Se focalizarmos a uma distância maior, o plano próximo ficará muito afastado da câmera. Portanto, as distâncias hiperfocais são úteis quando uma distância de profundidade de campo máxima é exigida a fim de se conseguir uma imagem nítida de objetos próximos e distantes.

Para ajustar a objetiva à distância hiperfocal correspondente à abertura desejada, é simplesmente necessário que se coloque o símbolo de infinito da escala de distância em frente à linha do número f que se pretende utilizar. A distância do plano próximo pode, então, ser estimado olhando na escala de distâncias. Analisando a figura 6.12 abaixo onde o símbolo de infinito está colocado em frente a linha do f/16, a profundidade de campo de uma objetiva com distância focal de 55 mm irá de 3 metros ao infinito.

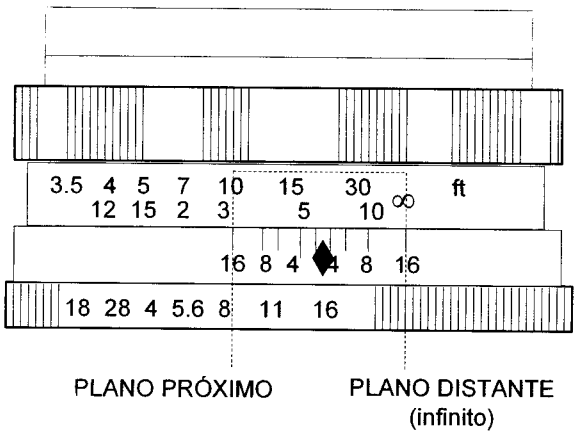


Figura 6.12 - Ajuste da objetiva numa distância hiperfocal (profundidade de campo máxima) com abertura de f/16.

USP SERVIÇO DE BIBLIOTECA
INFORMAÇÃO

6.2.1.2.5 - SUPERNITIDEZ

Uma supernitidez resulta no fato de que uma boa objetiva é capaz de registrar detalhes mínimos muito alem do que a vista consegue perceber. O que chamamos de profundidade de campo é, na verdade, o produto notável das objetivas das câmeras, que superam em muito a capacidade humana de perceber detalhes delicadíssimos. Assim, somente quando os objetos já estão a uma distância significativa do plano em que a objetiva da lente está focalizada é que o olho humano começa a perceber que estão fora de foco.

Para se conseguir uma nitidez máxima devemos evitar tanto as maiores quanto as menores aberturas das lentes. Para cada objetiva é recomendável consultar o fabricante.
Podemos, na prática estimar a região próxima ao plano que contém o ponto no qual a câmera está focalizada dentro da zona de profundidade de campo, como sendo a área onde se pode dizer ter uma supernitidez.

6.2.1.3 - DETECÇÃO DA IMAGEM

6.2.1.4 – CÂMERA

A imagem capturada consiste em intensidades relativas da luz que corresponde às várias porções da cena representadas por valores analógicos contínuos que serão convertidos na forma digital.

Para criar um sistema de visão por computador, podemos utilizar diversos componentes e tecnologias. Podemos utilizar câmeras com sofisticados transdutores para converter uma cena em um sinal elétrico (vídeo) a fim de ser analisado através do computador. Existem câmeras específicas para determinados tipos de aplicações.

As câmeras atualmente utilizadas em sistemas de visão são: dispositivos por injeção de carga (Charge injector device – CID), dispositivos a carga acoplada (Charge-coupled device – CCD) e com sensor bipolar de silício, sendo atualmente esta última a mais utilizada. A arquitetura das câmaras CCD, mais utilizadas em sistemas de visão são as do tipo câmeras de área (array câmera) e câmeras de varredura de linha (Line Scan Camera).

As câmeras CCD de área são constituídas por uma matriz de milhares ou de milhões de elementos fotossensíveis que capturam os pontos da cena de uma imagem estática de uma só vez, podendo apresentar resoluções, por exemplo, de 320x240, 640x480, 1280x960, 1280x1024, etc., enquanto que as câmeras de varredura de linha são recomendadas para o uso em aplicações que ocorrem com cenas em movimento, nas quais exigem alta velocidade e alta resolução.

Nos sensores de leitura que utilizam dispositivos de carga acoplada CCD, a imagem capturada é projetada sobre o CCD que detecta, armazena e fornece uma informação sobre a

carga acumulada gerada pela luz em cada ponto ou porção da imagem. A detecção da luz é efetuada pela absorção da mesma através de um substrato fotocondutor. Cada sensor CCD é um elemento fotosensitivo do estado sólido correspondente a um pixel, que gera e armazena uma carga elétrica quando a luz incide sobre o mesmo. Conforme pode ser visto na fig. 6.13, as cargas se acumulam na parte inferior dos eletrodos de controle carregados positivamente em poços isolados como resultado das tensões aplicadas aos eletrodos centrais. Assim, temos que cada depressão isolada representa um pixel e pode ser transferida para registros de armazenamento de saída variando as tensões sobre os eletrodos metálicos de controle.

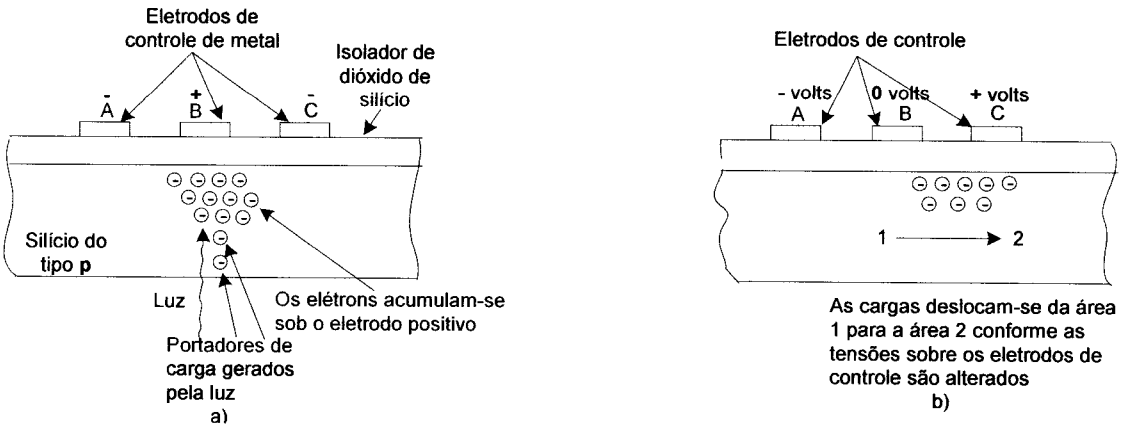


Figura 6.13 - Princípio básico de um dispositivo de carga acoplada.

(a) acumulação de uma carga de elétrons num elemento pixel;

(b) movimento de carga acumulada através de silício pela mudança de tensões sobre os eletrodos A, B, e C.

Um típico sensor inclui um circuito que armazena e transfere esta carga para um registrador de deslocamento, o qual converte a matriz espacial de cargas, em um sinal de vídeo analógico ou digital.

Na figura 6.14 abaixo, podemos verificar que as cargas ficam acumuladas durante o tempo necessário para completar uma única imagem, e a seguir são transferidas linha por linha para um registro de armazenamento. Notamos que o registro A acumula a carga do pixel produzida pela imagem luminosa. Após acumulada para uma única imagem, as cargas são transferidas linha por linha para o registro B. A seguir, as cargas de pixels são lidas e transmitidas linha por linha através de um registro horizontal C para um amplificador de saída. No transcorrer da transmissão, o registro A está acumulando mais pixels, sendo que este ciclo se repete em aproximadamente a cada 1/60 segundos.

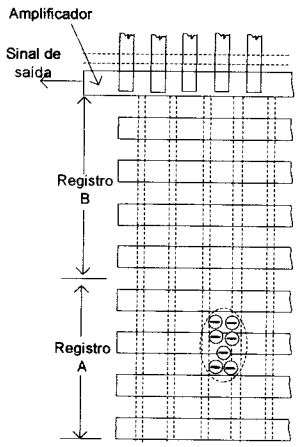


Figura 6.14 Um dispositivo de imagem a carga acoplada.

O registro A acumula as cargas de pixels produzidas por fotocondutividade gerada pela imagem luminosa. O registro B armazena as linhas de cargas de pixels e transfere cada linha por vez para o registro C. O registro C lê e transmite as cargas lateralmente conforme indicado, no amplificador.

Devido às características dos sensores CCD, que são, o seu baixo custo, a baixa potência de consumo de energia, boa resolução, dimensão reduzida, durabilidade e de fácil operação, fizeram com que os mesmos fossem adotados como uma tecnologia padrão para o uso em câmeras.

Os dois tipos de fotosensores são fotodiodos e fotogate. Existem também dois tipos principais de mecanismos de transferência: transferência chaveada digitalizada de tensão ou pacotes de carga e transferência de pacotes analógicos de fotocarga. Estes fotosensores combinam-se para formar os quatro tipos primários de arquitetura de leitores de saída: CID, CCD, CCPD, e SSPD, conforme pode ser visto na fig. 6.15.

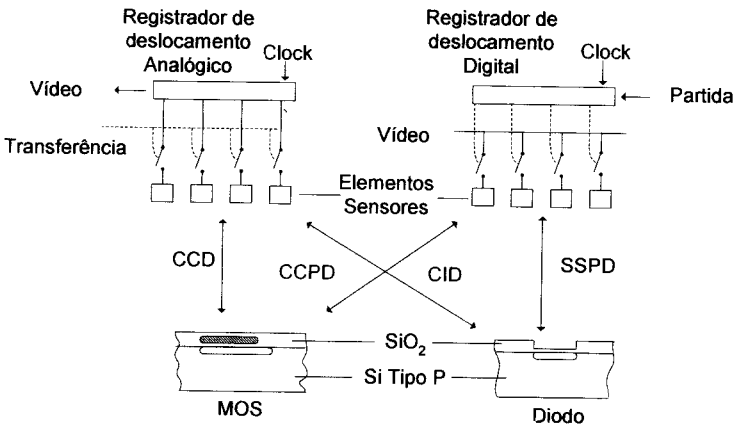


Figura 6.15 - Os tipos primários de sensores de imagem definidos pela combinação de fotosensores.

Na implementação deste projeto de sistema de visão, a câmera CCD foi escolhida por apresentar as seguintes características:

- opera livre de manutenção, não susceptível à degradação com o tempo.
- Não precisa de aquecimento.
- Não apresenta deslocamento de imagem.
- Resistência ao campo magnético estático.
- Leve e de tamanho reduzido.
- Não apresenta distorção geométrica de imagem.
- Alta resistência a impacto e vibração.
- Não apresenta envelhecimento quanto à imagem de entrada.
- Mínimo atraso na imagem.
- Baixa potência de consumo.

As seguintes características e propriedades foram levadas em consideração quando escolhemos a câmera CCD para aplicação do projeto de visão de máquina:

- resolução espacial/MTF.
- Sensibilidade/resposta.
- Detectividade (mínima resposta).
- Linearidade.
- Atraso/inércia.
- Alcance (amplitude) dinâmico.
- Sensibilidade espectral.
- Fluorescência.
- Ruído.
- Distorção geométrica.
- Longa estabilidade.
- Tempo de varredura/taxa de exibição.
- Dados mecânicos e elétricos.

6.2.1.5 - PLACA DE CAPTURA DE VÍDEO

Os equipamentos que produzem imagem ajudam a mostrar e analisar a informação visual representada pelo sinal de vídeo. A mostra eletrônica é executada por dispositivos de captura de vídeo freqüentemente chamados de placa de captura de imagens ou de vídeo (Frame

Grabber), que consiste em uma placa colocada em um microcomputador (PC). A placa de captura de imagem converte uma imagem (ou imagens) fornecida(s) por uma fonte de vídeo em uma matriz de pontos de dados, a qual pode ser armazenada digitalmente, processada e otimizada, e subseqüentemente analisada ou projetada em um monitor de vídeo.

Quando eletronicamente se mostra um sinal de vídeo, o processo refere-se comumente a uma aquisição de imagem, do qual é primeiramente necessário conhecer o tipo do sinal de vídeo que se irá trabalhar.

A placa de captura de vídeo é, portanto, um dispositivo que captura e converte sinais de vídeo analógico em dados digitais (uma exceção são as placas de captura digital as quais interfaceia a câmera digital, entretanto não converte dados).

Uma simples e mais importante tarefa que a placa de captura desempenha é converter dados de vídeo. Dependendo dos fabricantes e projetos fim, uma grande variedade de circuitos e técnicas pode-se ter sido usada no desenvolvimento de uma placa de captura. Dois tipos básicos de placa de captura são disponíveis: aquelas projetadas para aplicações industriais e científicas ou aquelas projetadas para aplicações em multimídia.

A placa de captura industrial e científica converte dados de vídeo o mais fiel possível, ou seja, sem alterá-los. Placas de captura para multimídia convertem dados, portanto, por propósitos estéticos, alterando o dado para ser visto melhor. Devido à vasta diferença de projetos fim, as placas de captura normalmente não pode ser usadas permutavelmente. No entanto, muitas placas de multimídia são divulgadas como sendo universal para todas as aplicações de imagem.

Os circuitos projetados para placa de captura a fim de serem utilizadas em multimídia podem induzir uma soma de variação de fase (flutuação de fase de alta freqüência) e ruídos. Em alguns casos elas, têm outros circuitos que podem significativamente alterar os dados de vídeo. Estas alterações, as quais não estão presentes nos dados originais, podem causar erros de medida, quando os dados de imagem forem posteriormente processados e analisados. Quando usado em aplicações que requerem alta exatidão (por exemplo, aferição, microscopia ou inspeção de disco) estas alterações podem ter efeitos danosos.

Quando avaliamos uma placa de captura para aplicações de uso industrial, é importante distinguir aquelas projetadas para uso industrial daquelas projetadas para aplicações em multimídia. As especificações típicas e critérios de projetos a considerar quando se seleciona uma placa de captura para aplicações industriais e científicas são:

- Impedância do canal de entrada.
- Filtro de entrada de alta qualidade.
- Entrada de Baixo Ruído e Conversão de circuitos.
- Exatidão de digitalização e velocidade do circuito de resposta.
- Relação do aspecto.

- Múltiplas entradas.
- Seleção das fontes de sincronismo.
- Relação espacial programada de clock e pixel.
- Entrada de sinal flexível.
- Correção do pulso de sincronismo.
- Tabelas de consulta LUT.
- Mostra de dados da imagem e sobreposição de vídeo.
- Medida de imagem e recorte.
- Controle da câmera.
- Entrada/Saída digital.
- Arquitetura de transferência de dados.
- Suporte de software.
- Desenvolvimento de software.

Na implementação do projeto desta tese foi utilizada uma placa de captura de vídeo STUDIO PCTV da Pinnacle Systems, destinada ao uso em multimídia, a qual apresentou bons resultados no uso desta aplicação.

6.2.2 - PRÉ-PROCESSAMENTO

Devido à qualidade da imagem capturada pela câmera, é necessário o estágio de pré-processamento da imagem. Para suavizar e corrigir pequenas distorções da imagem utilizamos filtros e binarização através da escolha de um limiar (*threshold*), possibilitando reduzir, assim, a sensibilidade quanto às condições de iluminação.

6.2.2.1 - FILTRAGEM

No estágio de pré-processamento utilizamos técnicas de realce com a finalidade de processar a imagem, de modo que a imagem final ficasse apropriada para a aplicação.

A técnica de realce é um dos tipos de transformação radiométrica, pois trabalha a modificação do valor dos níveis de cinza dos pixels que compõem a imagem.

Dentro das transformações radiométricas podemos distinguir dois tipos: a restauração e o realce, o primeiro visa corrigir alguma distorção sofrida pela imagem, enquanto que o segundo procura enfatizar alguma característica de interesse da imagem.

Os objetivos da restauração e do contraste podem produzir procedimentos coincidentes, como que, por exemplo, uma imagem que tenha sofrido uma distorção que tenha provocado uma diminuição do seu contraste, e se aplicado sobre a mesma uma operação de realce, as bordas dos objetos da imagem serão restauradas, restaurando assim a imagem.

O realce pode ser tratado dentro do domínio espacial ou no domínio da freqüência.

No domínio espacial, os procedimentos operam diretamente sobre os pixels da imagem. Já no domínio da freqüência, a abordagem é baseada na modificação das Transformadas de Fourier das imagens.

No processamento das imagens deste trabalho foram utilizadas máscaras espaciais para filtragem, usualmente chamadas de filtragem espacial.

As técnicas de filtragem no domínio espacial são aquelas que atuam diretamente sobre a matriz de pixels que formam a imagem digitalizada.

Para o realce de imagem podemos utilizar filtros espaciais lineares e não-lineares.

Os filtros lineares especificam que a função de transferência e o impulso ou função de espalhamento pontual de um sistema linear são transformadas inversas de Fourier uma da outra.

Os três principais tipos de filtros espaciais são: filtro passa-baixas, filtro passa-altas e filtro passa-faixas.

O filtro passa-baixas é utilizado para suavização e redução de ruídos de uma imagem, mas provoca um leve borramento na imagem, pois elimina as componentes de alta freqüência correspondente às regiões de bordas ou detalhes finos da imagem apresentada. Já o filtro passa-altas tem como objetivo o realce das regiões de alta freqüência, destacando assim, as bordas e os detalhes finos da imagem. E interessante notar que nos filtros passa-altas a soma algébrica dos coeficientes de sua máscara é zero, o que quer dizer que quando esta é aplicada a regiões homogêneas de uma imagem, o resultado será zero ou um valor mínimo, o que torna consistente com o princípio da filtragem passa-altas. Por fim, temos os filtros passa-faixa que são capazes de remover ou atenuar componentes acima de sua freqüência de corte superior e abaixo de sua freqüência de corte inferior, embora existam são de pouca utilidade prática, com exceção de algumas tarefas específicas de restauração de imagens.

Os filtros espaciais não-lineares baseiam-se diretamente nos valores dos pixels na vizinhança considerada. Estes podem ser usados para redução de ruído, computando o valor médio dos níveis de cinza na vizinhança em que o filtro está posicionado. Como exemplos temos o filtro de max, destinado a se encontrar os pontos mais claros de uma imagem, e o filtro de min, usado para encontrar os pontos escuros. Dentro deste conceito também podemos citar o filtro de mediana.

O filtro de mediana, vem para resolver a principal dificuldade dos filtros de suavização, que é a incapacidade de preservar as bordas e detalhes finos da imagem. Seu objetivo é reduzir o ruído, mas sem borrar a imagem. Neste processo, o nível de cinza de cada pixel é substituído pela mediana dos níveis de cinza da vizinhança daquele pixel ao invés da média. Este método é particularmente bom em situações nas quais a imagem está contaminada por ruído impulsivo.

Temos ainda os filtros por derivadas, os quais agudizam uma imagem. O aguçamento tem por finalidade enfatizar detalhes finos numa imagem ou realçar detalhes que tenham sido borrados em conseqüência de erros ou efeito natural do método de aquisição da imagem. O aguçamento tem uso em aplicações variadas como, por exemplo, impressão eletrônica, imagens médicas, inspeção industrial, detecção automática de alvos em armas inteligentes, etc.

No desenvolvimento deste trabalho, para realçar as bordas foram realizados testes com filtros lineares, através do uso de suas máscaras de convolução, tais como as de: Roberts, Sobel, Prewitt e Frei-Chen. Destas, a máscara de Sobel é a que apresentou melhores resultados, por mostrar bordas com melhor definição.

Pelo exposto, podemos verificar o motivo pelo qual no desenvolvimento deste trabalho optamos pelo uso do filtro de Sobel; ele se baseia no conceito de utilização do gradiente para diferenciação da imagem, o qual oferece a vantagem de fornecer, a um só tempo, os efeitos de diferenciação e de suavização, obtendo também como resultado uma imagem de gradientes do mesmo tamanho que a imagem original.

6.2.2.2 - FALSOS CONTORNOS

Na seqüência para a diminuição do fenômeno de falsos contornos, reduzimos o número de níveis de cinza de quantização, fazendo, assim, um processo de normalização do brilho, através da função de auto escala, implementada calculando para cada pixel com tom de cinza f, o nível resultante g, pela equação:

$$g = \frac{N}{f_{máx} - f_{mín}} (f - f_{mín}) \qquad (6.2)$$

onde $f_{máx}$ e $f_{mín}$, são, respectivamente, os níveis máximo e mínimo de cinza presentes na imagem original e N o limiar desejado (geralmente 32 – 1, ou 64 -1), a seguir, temos o algoritmo de execução da transformação linear dos níveis de cinza para 64 níveis:

1. Início
2. Encontrar os valores dos níveis de cinza máximo (f_{max}) e mínimo (f_{min}) da imagem velha

3. Calcular a nova imagem

$$g(i, j) = \frac{63}{f_{máx} - f_{mín}}(f - f_{mín})$$

4. Fim.

6.2.2.3 - LIMIARIZAÇÃO (THRESHOLDING)

No processo de limiarização, a iluminação tem influência significativa, pois esta provoca alterações no histograma da imagem que pode eliminar ou modificar região de vale entre dois picos proporcionando a definição de um limiar global.

No estudo de classificação, podemos levar em consideração o estudo da função de limiarização, que quando aplicada a uma cena produz uma imagem binária. Devido a este fato - de produzir uma imagem binária - o processo é denominado muitas vezes, binarização.

O processo de limiarização pode ser descrito em termos matemáticos, ou seja, considere uma função I(x,y) para representar uma imagem de entrada de N níveis de cinza e L(x,y) a imagem final limiarizada. A imagem final L(x,y) apresentará dois níveis de cinza, de tal modo que

$$L(x,y) = \begin{cases} 1 \text{ se } I(x,y) \geq T \\ 0 \text{ se } I(x,y) < T \end{cases} \tag{6.3}$$

Os pixels rotulados com 1 (ou qualquer outro nível de cinza conveniente) correspondem ao objeto, e os rotulados com 0 correspondem ao fundo.

Quando o limiar T depender apenas de I(x,y), ele é chamado de limiar global. Caso T, dependa de ambas as funções L(x,y) e I(x,y), recebe o nome de limiar dinâmico.

O valor do limiar T pode ser encontrado através de: do uso do próprio histograma; de forma arbitrária e subjetiva; uso de técnicas de cálculo do valor ótimo de limiar com base nas propriedades estatísticas da imagem [GONZALES & WOODS, 1992][CASTLEMAN, 1995][HARALICK e SHAPIRO, 1992]; método de seleção interativa [PARKER, 1994].

Com a intenção de se utilizar um valor ótimo de limiar e atender melhor aos objetivos desta tese, foi adotado o método de seleção interativa do limiar, pois ele apresenta os mesmos resultados, independentemente da iluminação, quando esta não satura o objeto da imagem.

O método de seleção interativa do limiar tem a finalidade de prover um estimado valor médio de nível de cinza da imagem entre o fundo T_b e o objeto T_o . O uso da média destes dois níveis é: $T = T_b + T_o /2$. Inicialmente, estes valores são estimados baseando-se nas propriedades da imagem. Se estas são conhecidas, e os objetos são escuros e o fundo é claro, o valor inicial pode ser $T_o = 0$ e $T_b = 255$. O passo seguinte é redefinir os valores de T_b e T_o usando o limiar T. Assumindo que as regiões são escuras, T_o é recalculado para ser o valor médio de todos os pixels menores do que T. Similarmente, os novos valores de T_b são a média dos valores de todos os pixels com os valores acima de T. Este processo produzirá uma melhor estimativa do nível médio e este é repetido para produzir um melhor estimado valor de limiar, o qual é depois recalculado, como antes, assim $T = T_b + T_o /2$, usando o novo valor de T_b e T_o.

Este processo é repetido até que o mesmo valor de limiar T seja encontrado em duas interações consecutivas, presumindo-se que neste ponto T seja o melhor limiar para a imagem.

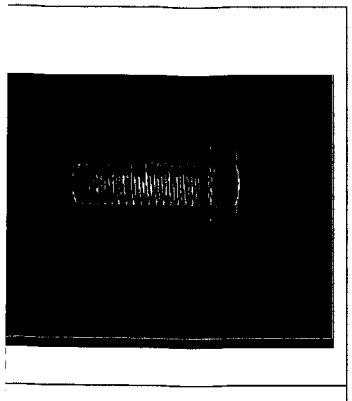
A seguir temos o algoritmo para o método de seleção interativa do limiar

1 - Selecionar um valor inicial do limiar T. Um bom valor inicial é a intensidade da média da imagem.
2 – Dividir a imagem em dois grupos G1 e G2 usando o limiar T.
3 – Calcular o valor médio do nível de cinza T_b e T_o dos grupos G1 e G2.
4 – Selecionar o novo limiar $T = T_b + T_o /2$.
5 – Repetir os passos 2 a 4 até os valores em sucessivas interações não mais se alterar.

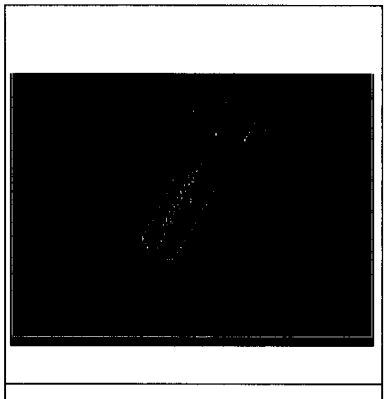
6.2.3 - EXTRAÇÃO DE CARACTERÍSTICAS

Para extração de características foi utilizado como base o cálculo do histograma, cálculo da média, cálculo do desvio padrão, cálculo da curtose, cálculo da energia, cálculo da entropia, cálculo de área, medidas de diâmetro (eixo maior, eixo menor) e elongação. Nesta etapa, para atender aos objetivos propostos nesta tese, procuramos escolher características determinantes de informações a serem extraídas da imagem que viessem atender a invariância geométrica pela translação, escala e rotação, visando também o não comprometimento da velocidade de processamento e a taxa de reconhecimento, reduzindo o dimensionamento da informação para proporcionar uma redução no tempo de cálculo dos atributos na fase de pré-processamento.

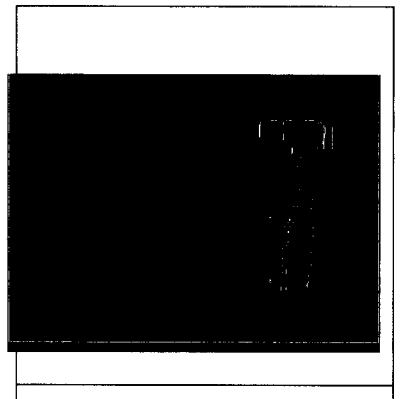
Podemos verificar pela tabela a seguir de extração de características a invariância quanto à posição, rotação e translação do objeto, onde suas características praticamente se mantiveram constantes.



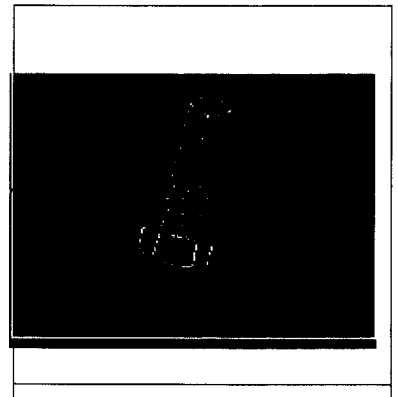
2 - 0°



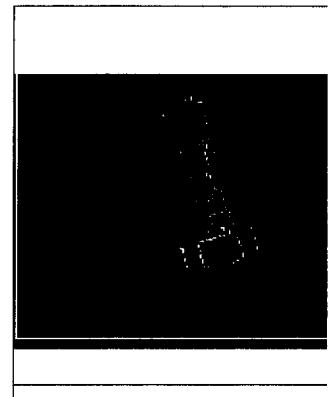
9 - 60°



49 - 90° deslocado



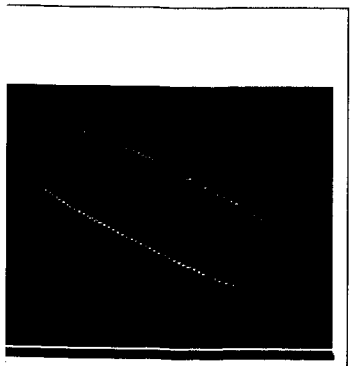
37 - 255°



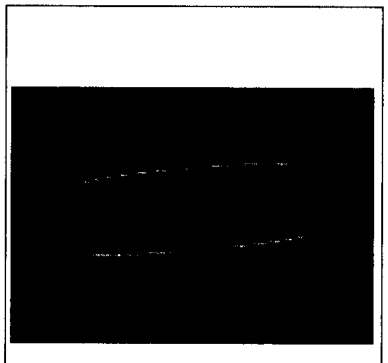
41 - 285°

PARAFUSO

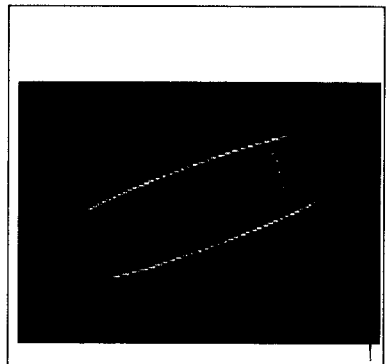
ça – Rot.	Média	DesvPadr	Curtose	Energia	Entropia	Área	EixoMaior	EixoMenor	Elongação
- 0°	4,881	8,983	298,192	72001,1	15,9	73200	187,265277026319	140,121897428421	1,33410409598938
- 60°	3,766	9,043	277,344	74159,8	16,1	71535	189,419816755436	139,219841528006	1,36945892604316
- 90° desl	3,766	9,043	277,344	74159,8	16,1	71970	184,713439638394	140,130305244998	1,32352395305655
- 255°	3,766	9,043	277,344	74159,8	16,1	72568	188,640961022026	139,531283537692	1,35008963580497
- 285°	3,766	9,043	277,344	74159,8	16,1	72617	188,580034854181	139,56266944568	1,34340056503021



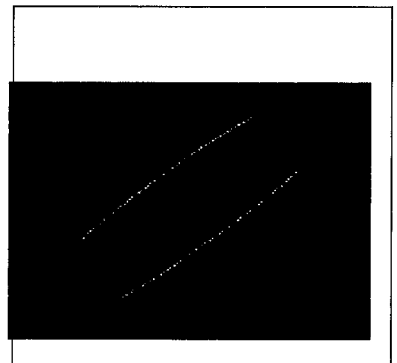
55 - 345°



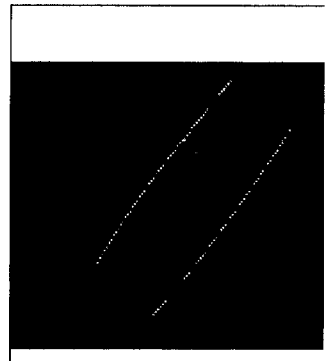
59 - 15°



61 - 30°



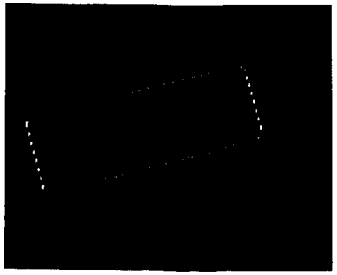
63 - 45°



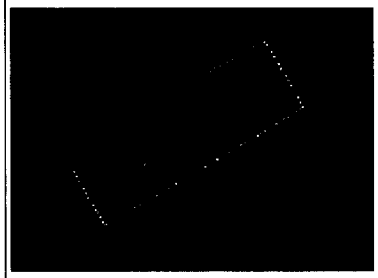
66 - 60°

FITA ISOLANTE

ça- Rot.	Média	DesvPadr	Curtose	Energia	Entropia	Área	EixoMaior	EixoMenor	Elongação
- 345°	5,417	8,954	342,884	65435,1	15,5	69674	190,345276675806	142,392666658968	1,32040180036998
- 15°	5,589	8,945	339,505	64693,6	15,4	70168	189,509775607478	142,368390990109	1,32979987872398
- 30°	5,517	8,949	302,769	63889,1	15,3	69912	190,133688809406	142,629828548497	1,33958594219987
- 45°	5,578	8,946	341,476	64978,4	15,4	69675	190,764363582503	141,611835199006	1,35472013207975
- 60°	5,270	8,962	350,030	68221,1	15,6	69540	191,447154153204	140,903039738233	1,36648620668973

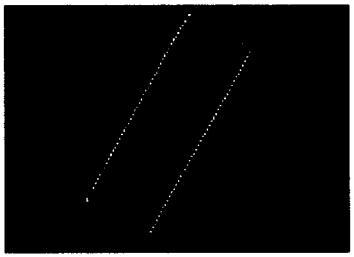


39 - 15°

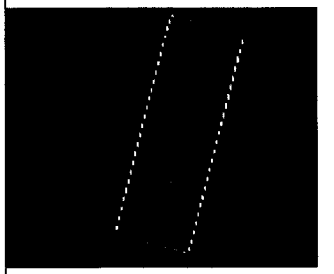


42 - 30°

43 - 45°



46 - 60°



47 - 75°

PARALELEPÍPEDO

ça - Rot.	Média	DesvPadr	Curtose	Energia	Entropia	Área	EixoMaior	EixoMenor	Elongação
- 15°	6,352	8,904	382,815	63024,8	15,3	70324	189,833867966481	142,024133234297	1,33707549444321
- 30°	6,139	8,915	366,300	63557,6	15,3	70232	190,257021738451	141,445769885401	1,34922459100276
- 45°	5,524	8,948	369,591	68689,3	15,7	69606	191,759825595548	140,021435363403	1,37423615390884
- 60°	5,524	8,948	369,591	68689,3	15,7	69606	191,759825595548	140,021435363403	1,37423615390884
- 75°	5,222	8,965	383,634	71869,0	15,9	69154	192,589994904701	139,59960168093	1,37922545558247

6.2.4 - RECONHECIMENTO DE PADRÕES

Nesta tese, procurando atender aos seus objetivos, optamos pelas redes neurais para se efetuar o reconhecimento de padrões como base para o desenvolvimento adaptativo dos coeficientes das funções de decisão através de apresentações sucessivas de padrões de treinamento.

Analisando-se as diversas redes neurais existentes, bem como as suas aplicações utilizadas em diversos trabalhos, optamos pelo uso da abordagem da Invariância pelo treinamento, aliado a atributos invariantes, visando atender a invariância geométrica e possibilitando o reconhecimento de padrões (objetos) invariante à transformação de escala, rotação e translação do padrão de entrada apresentado à rede neural.

Utilizamos a rede "*perceptron*" múltiplas camadas (MLP) com algoritmo de "*backpropagation*' para o treinamento por ser a mais recomendada pela literatura para este tipo de aplicação, verificando ser necessário um grande volume de dados para treinamento para se ter um melhor desempenho. O treinamento do sistema foi realizado preocupando-nos em obter um ambiente mais próximo possível das condições de operação de trabalho, preocupando-se com os parâmetros físicos como localização das câmeras, ajuste e abertura da lente, posicionamento das peças e sistema de iluminação adequado.

As Redes Neurais surgiram para atender às modelagens de problemas que não são representados através da programação convencional, como por exemplo, o reconhecimento de imagens, reconhecimento da fala, etc. Uma rede neural corresponde a um processador massivamente paralelo que possibilita armazenar conhecimento experimental adquirido através de um processo de treinamento e aprendizado. No desenvolvimento de uma aplicação de um projeto de Redes Neurais, deve-se seguir os seguintes passos:

1 – Estudar o problema.

2 – Escolher a arquitetura:

. Número de entradas/saídas.

. Recorrentes ou não.

3 – Escolher um algoritmo de aprendizado.

4 – Construir o conjunto de treinamento.

5 – Treinar a rede.

6 – Efetuar os testes.

Como o objetivo deste trabalho é desenvolver um sistema de visão em tempo real para operar em ambientes de trabalho sujeito a diversos tipos de ruídos, bem como, às invariâncias de rotação, translação e escala de objetos a serem manipulados, optou-se pelo uso da rede "*perceptron*" múltiplas camadas (MLP) com uso do algoritmo de treinamento "backpropagation" (retropropagação), a qual permite utilizar funções de decisão necessárias em

problemas de reconhecimento de padrões multiclasses, independentemente de as classes serem separáveis ou não, o que permite o uso desta arquitetura em soluções complexas. Esta rede, também conhecida como Rede Backpropagation, quando treinadas, possibilitam obter respostas satisfatórias quando são apresentadas entradas desconhecidas pela rede. Esta propriedade de generalização torna possível treinar uma rede utilizando um conjunto de entrada/saída e conseguir bons resultados de reconhecimento, quando novas entradas sem treinamento forem apresentadas à rede. O algoritmo Backpropagation utiliza o método do gradiente para minimizar uma função de erro, que é igual a diferença média entre as saídas desejadas e atuais da rede.

Redes com múltiplas camadas ou multicamadas utilizam redes feed-forward com uma ou mais camadas de neurônios entre os neurônios de entrada e saída. As camadas são constituídas de elementos estruturalmente idênticos chamados de neurônios, estruturadas de maneira que a uma camada alimente a entrada de cada neurônio da camada seguinte. Nesta rede temos, portanto, a camada de entrada, a camada de saída e as camadas intermediárias, escondidas ou ocultas. Não existe uma regra para definir a quantidade de camadas ocultas, no entanto, geralmente utiliza-se apenas uma, pois um maior número vai proporcionar um maior tempo de processamento. Na camada de entrada o número de neurônios corresponde à dimensionalidade dos vetores de padrões de entrada. A quantidade de neurônios na camada de saída é equivalente ao número de classes de padrões que a rede neural foi treinada para reconhecer. O número de neurônios na camada oculta da rede neural não obedece a uma regra específica, sendo este geralmente baseado tanto em experiência prévia como em uma escolha arbitrária, seguido de um refinamento por testes. Uma rede neural irá reconhecer um vetor de padrões x como pertencente a uma classe B_m se a *m*-ésima saída de rede é "alta", enquanto as outras saídas são "baixas".

Os neurônios utilizados na rede Backprogation têm a forma do modelo "*perceptron*", enquanto que a função de ativação por limiar é do tipo "*sigmoid*", pois para o desenvolvimento da regra de retropropagação é requerida uma diferenciabilidade ao longo de todos os caminhos da rede neural. O termo retropropagação é devido ao fato de que o erro obtido na saída é transferido para as camadas anteriores, e da necessidade de ajuste dos neurônios que não têm contato com a saída, os quais necessitam de algum parâmetro de atualização dos pesos.

Resumidamente, o cálculo do erro (iniciado a partir da saída) é realizado da seguinte forma:

$$\varepsilon S(t) = S(t)(1 - S(t)(D(t) - S(t)) \qquad (5.4)$$

onde S é a saída e D a saída desejada. Após encontrado o erro na saída, este é utilizado para ajustar os pesos da última camada, dado por:

$$W(t+1) = W(t) + \beta\varepsilon S(t)En(t) \tag{6.5}$$

onde W é o vetor de pesos, β é o coeficiente de aprendizado e En o vetor resultante da camada anterior.

O cálculo do erro nas camadas ocultas é feito a partir do erro da camada de saída, dado por:

$$\varepsilon_i = En(t)(1 - En(t)\sum_k \varepsilon_k w_{ik}(t) \tag{6.6}$$

onde En é o vetor resultante da saída da camada anterior até a camada intermediária, k é o número de nodos conectados a seguir do atual, ε_k é o erro do nodo k e w é o peso correspondente à conexão do nodo i atual com o nodo k. Calculado este erro, utiliza-o para o cálculo dos pesos pela seguinte fórmula:

$$W(t+1) = W(t) + \beta\varepsilon_i(t)En(t) + \mu(W(t) - W(t-1)) \tag{6.7}$$

onde μ é um coeficiente utilizado para acelerar a convergência da rede denominado *momentum*.

Na fase de aprendizado da rede neural, os vetores dos padrões de entrada devem ser preparados adequadamente em função do padrão exigido pela rede. Os vetores dos padrões de entrada devem ser apresentados à rede de forma alternada ou aleatória. A seguir, deve-se realizar algumas interações para definir os parâmetros ou coeficientes. Através do coeficiente de aprendizado, determina-se o intervalo de ajuste dos pesos, sendo que, quanto maior seu valor, maior será a diferença dos valores de uma iteração a outro. Temos também que, quanto maior este valor, maior será o intervalo de ajuste, e. por conseqüência, maior será a rapidez de convergência. Mas, pode ocorrer que para o caso de um valor muito alto, o sistema nunca chegue numa convergência e entre em um laço infinito. Este fato quando ocorre é porque os valores nunca serão pequenos o suficiente para se ter uma saída confiável, próxima de 0 ou 1. O valor deste coeficiente é arbitrário, definido pelo projetista da aplicação dentro do intervalo 0.01 a 1.

O ajuste do coeficiente de limiar pode acompanhar ao do peso ou ser definido pelo projetista. Este ajuste é responsável pelo descolamento, na função limiar, do ponto central de decisão da quantização da saída.

A fase de treinamento terminará quando as saídas desejadas estiverem numa margem segura de proximidade à saída da função. Considera-se esta aproximação devido à dificuldade de se chegar a um valor exato da saída desejada; logo, deve-se estimar uma margem de erro para que se alcance a convergência da rede. Para se chegar à convergência é necessário realizar várias interações. Quando ocorre a convergência, termina-se a fase de treinamento, obtendo, assim, uma rede treinada.

Por último, temos a fase de reconhecimento, que consiste em apenas uma iteração do algoritmo para verificação da saída. Coloca-se o vetor do padrão que se quer reconhecer na entrada da rede, executa-se a rede já treinada e compara a saída com as saídas desejadas aprendidas, verificando-se assim a qual classe o padrão apresentado pertence.

A seguir, temos o algoritmo para treinamento e teste para redes neurais MLP com "Backpropagation"

A) ALGORITMOS PARA TREINAMENTO DE REDES NEURAIS MLP COM "BACKPROPAGATION"

1. Início
2. Escolher o número de nós das camadas de entrada, escondida e de saída
3. Escolher o erro e o número de interações total para a parada do treinamento
4. Inicializar pesos w_{ij} com valores pequenos [-1,1], fazer interação = 0 e erro da rede Ep = 0
5. Selecionar um par de treinamento $\{x^P, t^P\}$ (entrada, saída) e computar para cada camada q

   $S^q_j = f[\Sigma_i S^{q-1}_i * w^q_{ij}]$ se f for sigmoidal: $Sqj = 1/(1+\exp\{-k[\Sigma_i S^{q-1}_i * w^q_{ij}]\})$
   Notando que $S^0_i = x_i$

6. Usar o valor S^q_j computado na camada final e seu correspondente valor alvo t^P_j para computar a quantia delta e o erro do modelo

   $\delta^Q = (S^q j - t^P_j) f'(H^Q_j)$ se f for sigmoidal: $\delta^Q = k.S^Q_j(1 - S^Q_j)(t^P_j - S^q_j)$
   para todos j usando o modelo P.
   Erro do modelo $Ep = Ep + ½ * (t^P_i - S^q_i)^2$

7. Computar a regra delta para cada uma das camadas precedentes usando

   $\delta^{q-1}_j = f'(H^{q-1}_j)\, \Sigma_i \delta^q_i w^q_{ji}$ se f for sigmoidal: $\delta^{q-1}_j = K.S^{q-1}_j(1-S^{q-1}_j)\, \Sigma_i \delta^q_i w^q_{ji}$
   para todos os nós j nas camadas q.

8. Atualizar os pesos

   $w_{ij} = w_{ij} + \Delta w^q_{ij}$ onde $\Delta w^q_{ij} = \eta \delta^q_i S^{q-1}_j$

9. Retornar ao passo 5 e repetir para cada amostra do conjunto de treinamento

10. Incrementar interações em uma unidade e fazer
    Ep = Ep/número_de_amostras
11. Se o erro total do conjunto for menor que o erro escolhido ou se o número de interações for maior que o número_de_interações_total parar, caso contrário reapresentar novamente o modelo partindo do passo 5.
12. Salvar os pesos w_{ij}
    Fim

B) ALGORITIMO PARA TESTE DA REDE MLP "BACKPROPAGATION"

1. Início
2. Carregar os pesos w_{ij}
3. Apresentar o vetor de características e computar as saídas nas camadas

$S^q_j = f[\Sigma_i S^{q-1}_i * w^q_{ij}]$ se f for sigmoidal: $Sqj = 1/(1+\exp\{-k[\Sigma_i S^{q-1}_i * w^q_{ij}]\})$
Sendo que $S^0_i = x_i$

4. Fim.

6.2.5 - APLICAÇÃO

A aplicação foi realizada procurando simular um ambiente o mais próximo possível do real de trabalho. Este projeto teve como uma das finalidades o seu uso em um robô (manipulador mecânico) industrial localizado no chão de fábrica do laboratório da engenharia mecânica da EESC-USP (Escola de Engenharia de São Carlos – Universidade de São Paulo).

No desenvolvimento do sistema foi necessário fazer um critério de separação de classes, para se ter um ajuste mais preciso no reconhecimento dos objetos, preocupando-se com o posicionamento dos mesmos e adequar o sistema de iluminação para cada tipo. Nesta separação de classe, devemos nos preocupar também com a forma e textura (coloração e tipo de superfície do objeto). Quanto a orientação e disposição do objeto a ser reconhecido, procuramos atender ao objetivo proposto quanto à invariança geométrica.

CAPÍTULO 7

7 - DESENVOLVIMENTO EXPERIMENTAL

7.1 - SISTEMA DE ILUMINAÇÃO

No primeiro momento, foi necessário analisar os possíveis sistemas de iluminação que viesse a atender os requisitos exigidos para simular um ambiente o mais próximo do real. Pode ser verificado o quanto é importante o sistema de iluminação em função do ambiente de trabalho e o objeto a ser reconhecido. Ao se realizar testes com iluminação natural e artificial, verificamos ser necessário, para cada situação de local e tipo de objeto, adotar sistema com difusor para espalhar e difundir a luz, bem como o controle de sua intensidade, e, se necessário, utilizar o recurso de filtros polarizadores, para eliminar as sombras e ruídos, com a finalidade de se obter contornos mais nítidos, melhorando, assim, a qualidade da imagem capturada. Pode ser utilizada iluminação direta ou indireta, com a finalidade de reduzir a quantidade de fontes de luz.

7.2 - SUPORTE PARA FIXAÇÃO DA CÂMERA

No experimento, foi utilizou-se um suporte com trilho para deslizar a câmera, permitindo o uso de apenas uma câmera e não duas como descrito para o projeto.

7.3 - AQUISIÇÃO DA IMAGEM

Para capturar as imagens foi utilizada uma câmera CCD, acoplada a um suporte deslizante, de forma a obter vistas transladadas (esquerda, central e direita) e rotacionada da imagem, uma placa de captura de imagem e um microcomputador. A coordenada z (distância da câmera até o objeto) foi medida manualmente, podendo a mesma ser lida por sistemas laser, ultra-som ou pelo processo de geração de imagens em estéreo.

Foram capturadas inúmeras imagens de diversos objetos, e algumas, foram selecionadas para formar o conjunto de padrões a ser utilizado para teste neste trabalho. A escolha do conjunto para

teste deve ser bem analizado para que possamos obter o tempo de aprendizagem da rede neural reduzido.

As imagens utilizadas para a realização deste trabalho são constituídas por 320x240 pixels cada uma. No apêndice A são apresentadas algumas das imagens utilizadas.

7.4 - PROCESSAMENTO PARALELO

7.4.1 - PRÉ-PROCESSAMENTO, TREINAMENTO E TESTE

Para realização do pré-processamento paralelo foi utilizado algoritmo paralelo para o nó mestre e escravo, apresentado no item 7.4.3 a seguir, desenvolvido levando-se em consideração o peso do custo de comunicação entre os processos em relação ao tempo de execução efetivo. Para torná-lo mais eficiente do que o seqüencial, bem como pensando também na arquitetura do SPP3, na qual será executado, utilizamos a biblioteca MPI (Message Passing Interface) como ambiente para programação por passagem de mensagem.

O multicomputador SPP3 utilizado possui oito nós (P0, P1, P2, P3, P4, P5, P6 e P7), porém o nó P5 não estava em funcionando; logo trabalhamos com os sete nós disponíveis.

Na implementação final utilizamos as RNAs (Redes Neurais Artificiais) treinadas de forma supervisionada, utilizando o algoritmo "*Backpropagation Momentum*". Para diminuir os esforços a serem gastos na manipulação da rede neural durante toda a fase de aprendizagem, validação e teste, procuramos utilizar um ambiente simples e flexível para treinamento que permitisse visualizar de forma rápida o desempenho da rede, optamos, então, pelo uso do software SNNS (Stuttgart Neural Network Simulator), desenvolvido pela Universidade de Stuttgart – Alemanha [SNNS,1995].

No teste final foram utilizados cento e sessenta e oito (168) padrões correspondentes a três (3) conjuntos de imagens, sendo que cada conjunto corresponde a imagens de um objeto em várias posições, tamanhos e condições de iluminação diferente.

O treinamento e aprendizagem da rede neural foram realizados utilizando cento e sessenta e oito (168) padrões os quais foram divididos em oitenta e quatro (84) padrões para o conjunto de treinamento, quarenta e dois (42) padrões para o conjunto de validação e quarenta dois (42) padrões para o conjunto de testes correspondente a objetos a serem reconhecidos, atendendo às instruções propostas em PROBEN 1 [PRECHELT, 1994]. Cada padrão é constituído por um vetor composto de nove elementos obtidos pelo cálculo da média, desvio padrão, curtose, energia, entropia, área, eixo maior, eixo menor e elongação, extraído da imagem que o representa, cujos resultados se encontram na tabela de características das imagens constante no apêndice B. Assim, ficou definido a utilização de redes neurais com nove unidades de entrada, ***n*** unidades intermediárias e três unidade de saída.

Na fase de treinamento e teste o paralelismo foi implementado utilizando a estratégia processo "farm". Neste processo, o sistema fica dividido entre mestre (master) e escravos (slave), conforme pode ser visto através da figura 7.1. As tarefas de filtragem, eliminação da falsos contornos, limiarização, histograma e extração de característica estão distribuídas separadamente em cada nó escravo, enquanto que o nó mestre (processador P0) fica responsável pelo controle geral para

supervisionar, controlar e distribuir as tarefas para os nós escravos. Desta forma temos a ocorrência de processamento simultâneo em cada nó e um balanceamento automático da distribuição de carga, o que vem a dispensar qualquer tipo de balanceamento.

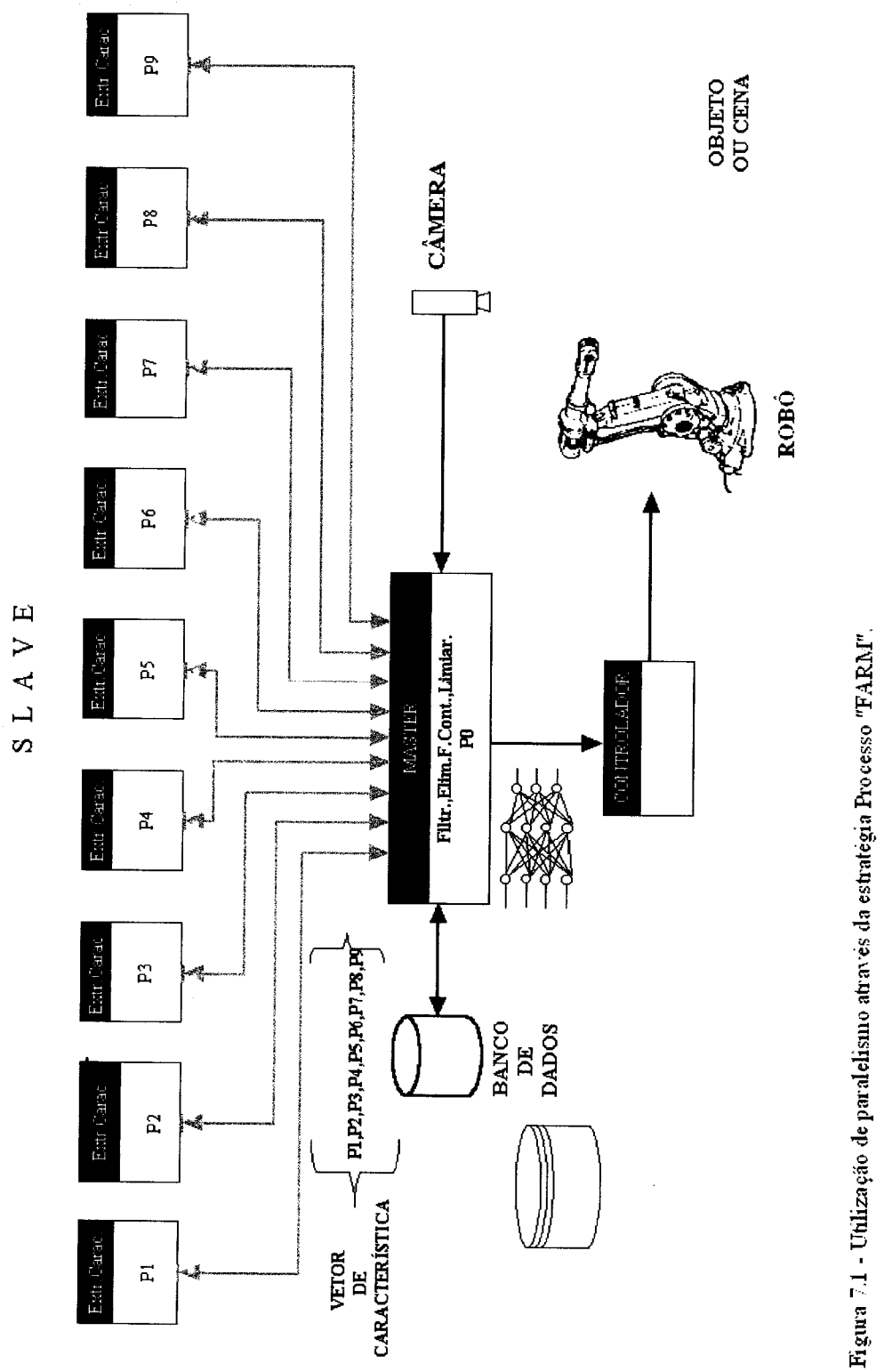


Figura 7.1 - Utilização de paralelismo através da estratégia Processo "FARM".

Fig. 7.1 - Utilização de paralelismo através da estratégia processo "FARM".

IFSC-USP SERVIÇO DE BIBLIOTECA INFORMAÇÃO

Durante a fase de treinamento e teste é enviada pelo nó mestre para cada nó escravo a quantidade total de padrões dividida pelo número de nós, em uma seqüência de faixas (lotes) de imagens, porém, o nó mestre deve controlar este envio, verificando se existe algum nó vago (livre), na seqüência de execução da filtragem, binarização e extração de características. Cada nó escravo, após extrair as características de cada padrão, enviará para o nó mestre um vetor (nove (9) características mais identificação do padrão), que constituirá uma linha da matriz de dez (10) colunas x número de linhas de padrões utilizado.

O nó mestre, após verificar que todos os nós estão vagos (término de pré-processamento), normalizará as nove primeiras colunas da matriz recebidas, embaralhará as suas linhas e em seguida salva esta matriz com a extensão. pat, construindo, assim, um arquivo que será a entrada da rede neural, que se encontra no mesmo, para fornecer o resultado do processo de treinamento ou teste. Alguns desses arquivos podem ser vistos no apêndice B.

A seguir, realizamos teste de reconhecimento com imagens que não pertenciam ao conjunto de teste de treinamento de aprendizagem.

7.4.2 - UTILIZAÇÃO DO SISTEMA PARA RECONHECIMENTO E APLICAÇÃO

Na fase de reconhecimento foi explorado o paralelismo utilizando a divisão do módulo de extração de característica, através de uma estrutura mestre/escravo, conforme pode ser visto na figura 7.2. Nesta figura é apresentado o sistema no qual pode ser verificado, a exploração do paralelismo, pois, em cada nó (escravo) é realizado de forma concorrente a extração das características da imagem do objeto a ser reconhecido. O módulo mestre é responsável pela aquisição, filtragem, eliminação de falsos contornos, limiarização, histograma da imagem e controle geral do sistema. Neste módulo é que se encontra a rede neural já treinada utilizada para identificação do objeto a ser reconhecido.

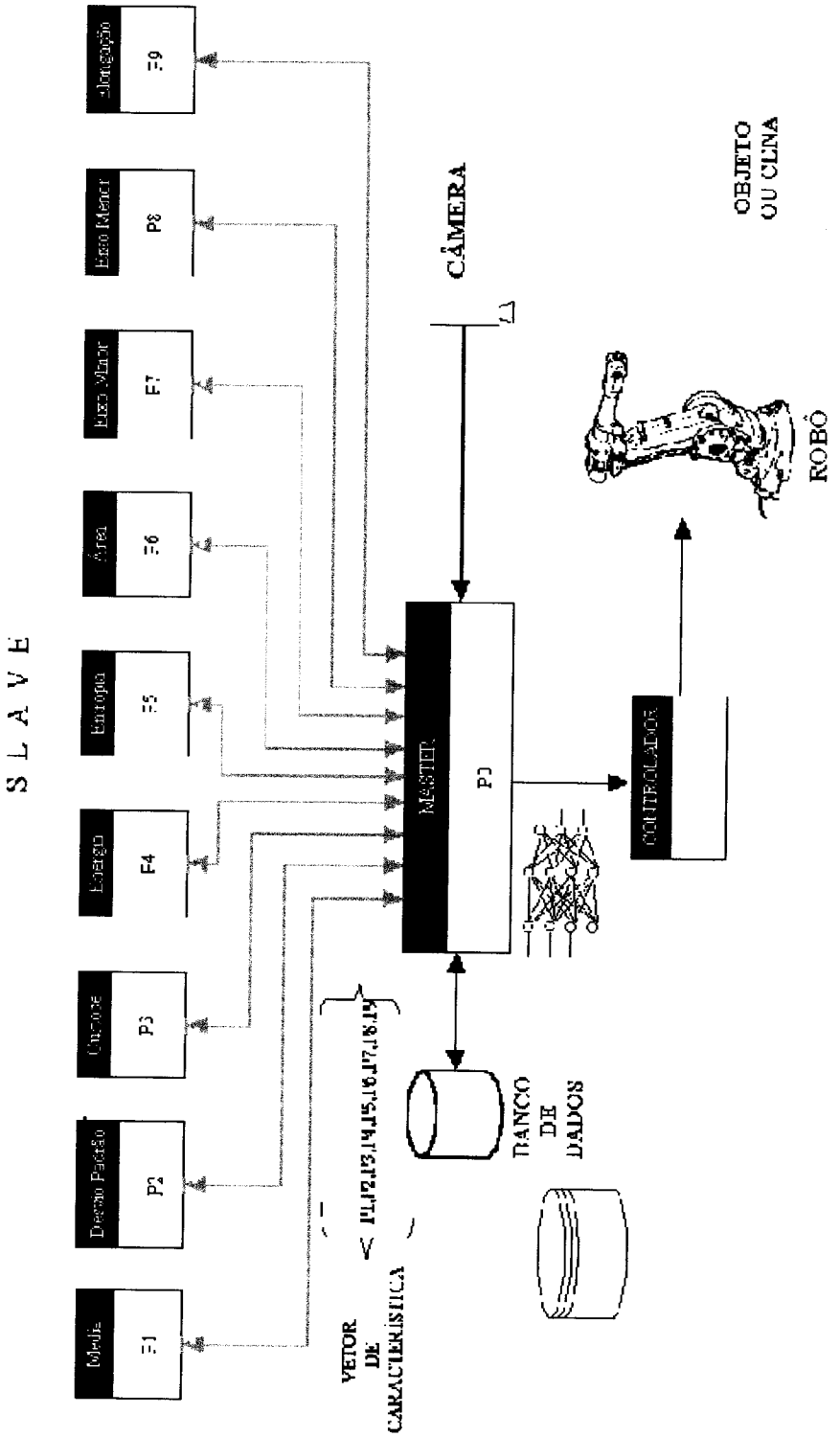


Figura 7.2 - Paralelismo utilizando divisão do módulo de extração de característica

Figura 7.2 Paralelismo utilizando a divisão do módulo de extração de característica.

Foi utilizado o nó P0 como mestre (fig. 7.2) contendo a rede neural treinada e os programas de aquisição e tratamento da imagem e como escravos os nós P1 a P7 (menos o nó P5). O nó mestre após o processo de aquisição da imagem do objeto realiza os processos de filtragem, eliminação de falsos contornos, limiarização, histograma da imagem a envia para cada nó escravo para que estes realizem as tarefas (Média, Desvio Padrão, Curtose, Energia, Entropia, Área, Eixo Maior, Eixo Menor e Elongação) distribuídas em cada nó. Após realizar as referidas tarefas os nós escravos às enviarão para o nó mestre, o qual montará uma matriz constituída por uma (1) linha x nove (9) colunas que será normalizada e em seguida a salvará com a extensão .pat. Na seqüência esta matriz será enviada para a entrada da rede neural treinada para identificação do objeto.

O objeto identificado é enviado para um banco de dados, que contém as características geométricas dos objetos utilizados. Feito a consulta, se este é encontrado, o banco de dados, envia o vetor característica geométrica para a saída, que será lida por um programa instalado no módulo de controle responsável pela movimentação do braço mecânico (robô), cuja função é analisar as variáveis da imagem de entrada e fornecer as instruções que serão utilizadas pelo mecanismo de manipulação da peça. Estes dados referem-se à posição real da peça em relação à plataforma de montagem, ângulo de posicionamento da peça na plataforma e medidas da peça a ser manipulada. Recebido estas grandezas, o módulo de controle gera como saída os parâmetros que controlarão o posicionamento da garra para manipular o objeto.

7.4.3 - PROGRAMAÇÃO

Para a programação paralela desenvolvida foi utilizada a biblioteca MPI (Message Passing Interface), utilizando um conjunto de primitivas para a troca de mensagens.

A figura 7.3 apresenta as principais primitivas de troca de mensagem do MPI e sua respectivas funções.

MPI_init	Inicializa um processo
MPI_Comm_rank	Identifica um processo MPI
MPI_Comm_size	Retorna o número de processo
MPI_Send	Rotina para envio de mensagens
MPI_Recv	Rotina para recepção de mensagens
MPI_Finalize	Finaliza um processo

Fig. 7 3 – Primitivas básicas do MPI

A seguir, é apresentado o algoritmo paralelo para o processo Mestre e Escravo, utilizado no desenvolvimento do sistema proposto nesta tese.

ALGORITMO PARALELO PARA O NÓ MESTRE

```
1. Gerar a lista de Imagens
   Total_Escravos = Número_de_Escravos
   Total_Faixas = Total_de_Imagens/10
   i = 1;
   Rank = 1;

2. WHILE(i < Total_Faixas and Total_Escravos <> ∅)
      SEND (i , 1 , MPI_INT, Rank , Tag , MPI_COMM_WORLD)
      i = i + 1;
      Rank = Rank + 1;
      Atualiza Tabela
      Total_Escravos = Total_Escravos – 1;

3. WHILE (Total_Escravos < Número_de_Escravos)
      Recv (Vetor[90], 10 , MPI_FLOAT, MPI_ANY_SOURCE, TAG,
            MPI_COMM_WORD, Status)
      Verifica quem enviou Resultados(MPI_Status) → P1=MPI_Source
      Total_Escravos = Total_Escravos + 1;
      Atualiza Matriz;
      IF (i < Total_Faixas)
            SEND( i , 1 , MPI_INT, P1, TAG, MPI_COMM_WORD)
            i = i + 1;
            Atualiza Tabela;
            Total_Escravos = Total_Escravos –1;

4. MPI_FINALIZE;

      Tratamento da Matriz
```

ALGORITMO PARALELO PARA O NÓ ESCRAVO

1. RECV (Faixa, 1 , MPI_INT, ∅ , Tag, MPI_COMM_WORD)

   Processamento da Imagem

2. SEND (Vetor, 9 , MPI_FLOAT, ∅ , Tag, MPI_COMM_WORD)

A base computacional do algoritmo pode ser dividida em duas etapas: transmissão dos dados e processamento. Na etapa de transmissão de dados temos duas fases, que são aquisição e distribuição das imagens a serem processadas e a transmissão dos resultados (características) para o nó mestre. A demanda computacional para o processamento aumenta à medida que se tenha um maior volume de padrões, enquanto que a demanda computacional requerida pela transmissão de dados pouco se altera, devido à alta banda de transferência do SPP3. Desta forma, observamos que a utilização do multicomputador e o método utilizado para o processamento paralelo veio a contribuir sensivelmente na solução do processamento de imagem e visão computacional, pois o seu grande gargalho se encontra justamente no tempo de processamento da imagem, justificando a implementação com a utilização do multicomputador de alto desempenho.

A performance da implementação paralela frente à seqüencial aumenta à medida que se utiliza um número maior de padrões.

Para análise dos resultados (quanto ao comportamento da performance), foi utilizada a medida de Speed-Up, dada pela razão entre o tempo de execução seqüencial e o tempo de execução paralela.

Logo, a performance do sistema será maior, quanto maior for a quantidade de nós processadores no multicomputador, o que vem justificar a forma modular construtiva do SPP3, que possibilita a atualização dos processadores e do número de módulos (nós) de processamento, em função da aplicação a ser realizada.

7.4.4 - ANÁLISE DOS RESULTADOS

De um conjunto de cento e oitenta (180) padrões constituídos por imagens, formada por 320x240 píxel, foram utilizados cento e sessenta e oito (168) padrões subdivididos em três (3) conjuntos de dados constituindo um conjunto de oitenta e quatro (84) padrões para treinamento, quarenta e dois (42) padrões para validação e quarenta e dois (42) padrões para teste. Os conjuntos de treinamento e validação **A** foram obtidos partir do embaralhamento das linhas da matriz formada pelos 126 padrões que constituirão os dois conjuntos. A seguir realizam-se mais dois embaralhamentos para formar os conjuntos **B** e **C** respectivamente. Os arquivos correspondentes a estes três conjuntos de treinamento e validação podem ser vistos no apêndice B. Após formados estes conjuntos de treinamento e validação é então realizado a verificação do desempenho da rede na fase de treinamento e validação, como se pode verificar pelos resultados obtidos e transcritos nos parágrafos que se seguem.

Como pode ser visto na fig. 7.4 a rede neural definida para o uso neste trabalho é uma rede do tipo *Perceptron* Múltiplas Camada (MLP) utilizando nove (9) neurônios de entrada que correspondem às características extraídas das amostras de cada padrão a ser analisado, ***n*** neurônios para a camada intermediária e três (3) neurônios pertencentes à camada de saída correspondente à quantidade de padrões a serem identificados.

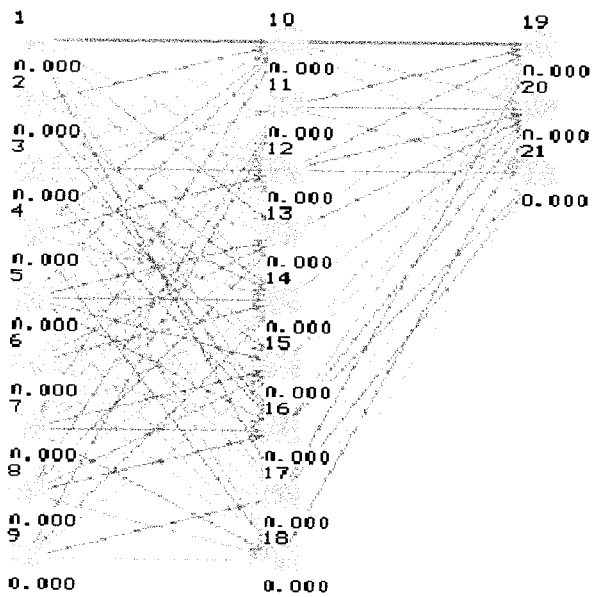


Figura 7.4 Rede Neural Perceptron Múltiplas Camadas (MLP) com nove neurônios na camada de entrada, 9 neurônios na camada Intermediária e 3 neurônios na camada da saída.

Podemos-se verificar a seguir os resultados do desempenho obtidos através de testes realizados durante o treinamento das redes neurais (RNA) com variações do número de elementos da camada intermediária, parâmetros de aprendizagem e momento, função de treinamento e o número de padrões dos objetos para o treinamento, validação e teste utilizados na escolha da arquitetura implementada neste trabalho.

Na realização dos testes de treinamento e validação foram utilizadas dezoito (18) arquiteturas diferentes, variando-se o número de neurônios na(s) camada(s) intermediária(s). A descrição do tipo de arquitetura utilizada é representada na seqüência das quantidades de neurônios existentes nas camadas de entrada, intermediária(s) e de saída.

Os resultados obtidos da média de dez (10) treinamentos e validação do desempenho dos conjuntos de padrões **A**, **B** e **C**, utilizando uma rede neural com (**9**) neurônios de entrada, três (3) neurônios de saída e ***n*** neurônios na camada intermediária, função de aprendizagem **Backpropagation Padrão** com coeficiente de aprendizagem 0,4 para sessenta mil ciclos (60.000), podem ser observados nas tabelas 7.1 (a), (b) e (c) a seguir:

Arquitetura	Treinamento	DesvPadrão	Validação	DesvPadrão
9-3-3	0,043006	0,129252	0,278810	0,017395
9-6-3	0,019282	0,060975	0,218771	0,082166
9-9-3	0,021265	0,062309	0,237578	0,06586
9-12-3	0,032741	0,092152	0,228668	0,036642
9-15-3	0,019728	0,062385	0,206317	0,037795
9-18-3	0,197060	0,062316	0,203745	0,075809
9-30-3	0	0	0,208781	0,022847
9-3-3-3	0,146517	0,158355	0,297354	0,040142
9-6-3-3	0,050990	0,107382	0,226421	0,119211
9-6-6-3	0	0	0,219536	0,031687
9-9-9-3	0	0	0,194203	0,042078
9-9-6-3	0	0	0,171368	0,037372
9-9-3-3	0,100640	0,154512	0,249444	0,149468
9-15-3-3	0,066079	0,116148	0,229594	0,097552
9-18-3-3	0,049476	0,105868	0,213166	0,091323
9-18-6-3	0,030041	0,089724	0,193489	0,111525
9-9-9-9-3	0,121680	0,157875	0,281122	0,079359
9-18-9-6-3	0,307524	0,300817	0,427068	0,267704

Tabela 7.1 (a) Desempenho do teste de treinamento e validação utilizando o conjunto A.

Arquitetura	Treinamento	DesvPadrão	Validação	DesvPadrão
9-3-3	0,00983	0,020443	0,250533	0,018031
9-6-3	0	0	0,234884	0,015659
9-9-3	0	0	0,202635	0,034922
9-12-3	0	0	0,221236	0,032129
9-15-3	0	0	0,212473	0,02432
9-18-3	0	0	0,216963	0,016518
9-30-3	0	0	0,213952	0,019975
9-3-3-3	0,223444	0,199931	0,313439	0,149906
9-6-3-3	0,005641	0,012618	0,20885	0,018827
9-6-6-3	0,005123	0,0162	0,198479	0,029815
9-9-9-3	0	0	0,203856	0,032081
9-9-6-3	0	0	0,20096	0,016021
9-9-3-3	0	0	0,198256	0,023989
9-15-3-3	0	0	0,195588	0,019331
9-18-3-3	0,026457	0,083664	0,20534	0,042716
9-18-6-3	0,028882	0,084762	0,214251	0,034481
9-9-9-9-3	0,297568	0,262609	0,435246	0,19806
9-18-9-6-3	0,143788	0,199345	0,305244	0,153345

Tabela 7.1 (b) Desempenho do teste de treinamento e validação utilizando o conjunto B.

Arquitetura	Treinamento	DesvPadrão	Validação	DesvPadrão
9-3-3	0	0	0,263840	0,008580
9-6-3	0	0	0,240591	0,033605
9-9-3	0	0	0,233621	0,030063
9-12-3	0,001855	0,104278	0,178388	0,028079
9-15-3	0	0	0,200921	0,024758
9-18-3	0	0	0,194870	0,020489
9-30-3	0	0	0,213792	0,017330
9-3-3-3	0,104741	0,147605	0,259915	0,103358
9-6-3-3	0	0	0,186011	0,016890
9-6-6-3	0	0	0,197513	0,026040
9-9-9-3	0	0	0,194418	0,039382
9-9-6-3	0	0	0,187222	0,023511
9-9-3-3	0	0	0,182964	0,021524
9-15-3-3	0	0	0,178177	0,016370
9-18-3-3	0	0	0,166393	0,009829
9-18-6-3	0	0	0,178489	0,018500
9-9-9-9-3	0	0	0,197154	0,016093
9-18-9-6-3	0	0	0,192900	0,016522

Tabela 7.1 (c) Desempenho do teste de treinamento e validação utilizando o conjunto C.

Tabela 7.1 Resultado do desempenho encontrado para o teste de treinamento e validação dos conjuntos: (a) A, (b) B e (c) C, utilizando a função de aprendizagem **Backpropagation Padrão** com coeficiente de aprendizagem 0,4 após o cálculo da média de dez (10) testes.

Analisando os resultados obtidos registrados nas tabelas acima verificamos que a arquitetura de rede que apresentou melhor resultado é a que contém nove (9) neurônios na primeira camada intermediária e seis (6) neurônios na segunda camada intermediária.

Após os testes realizados e o resultados acima obtidos, verificou-se que, para uma melhor análise seria necessário mudar a função de aprendizagem, optando-se então, pela **função Backpropagation Momentum.** Assim procurando encontrar a melhor arquitetura realizaram-se novos testes de treinamento e validação com os conjuntos **A**, **B** e **C** contendo cada um oitenta e quatro (84) padrões para o conjunto de treinamento e quarenta e dois (42) para o conjunto de validação e três (3) objetos a serem reconhecidos, utilizando uma rede neural com (**9**) neurônios de entrada, três (3) neurônios de saída e ***n*** neurônios na camada intermediária, modificando a função de aprendizagem para **Backpropagation Momentum,** utilizando o coeficiente de aprendizagem 0,4 e com o fator de momento 0,5 mantendo-se o número de ciclos de treinamento e foram encontrando os resultados apresentados na tabela 7.2 (a), (b) e (c) a seguir após o cálculo da média de dez (10) treinamentos e validação.

Arquitetura	Treinamento	DesvPadrão	Validação	DesvPadrão
9-3-3	0,126776	0,182122	0,326286	0,02214
9-6-3	0,040124	0,094039	0,260663	0,012452
9-9-3	0,041721	0,11658	0,278250	0,30296
9-12-3	0	0	0,237246	0,019757
9-15-3	0,000471	0,001489	0,239729	0,000453
9-18-3	0	0	0,223621	0,010133
9-30-3	0	0	0,226480	0,011618
9-3-3-3	0,391225	0,263838	0,511582	0,194285
9-6-3-3	0,530653	0,137296	0,590161	0,152884
9-6-6-3	0,465404	0,102354	0,553101	0,051548
9-9-9-3	0,401464	0,198452	0,485541	0,137172
9-9-6-3	0,511112	0,09751	0,597682	0,021362
9-9-3-3	0,549217	0,109658	0,237294	0,152671
9-15-3-3	0,464131	0,18065	0,555738	0,042667
9-18-3-3	0,467430	0,227751	0,539986	0,264988
9-18-6-3	0,534772	0,100723	0,608082	0,1259
9-9-9-9-3	0,554690	0,092933	0,593872	0,088346
9-18-9-6-3	0,619812	0,120925	0,657776	0,126226

Tabela 7.2 (a) Desempenho do teste de treinamento e validação utilizando o conjunto A.

Arquitetura	Treinamento	DesvPadrão	Validação	DesvPadrão
9-3-3	0,227759	0,22555	0,434395	0,165348
9-6-3	0,046	0,117293	0,29838	0,098756
9-9-3	0,002207	0,006979	0,234503	0,067381
9-12-3	0,010849	0,023221	0,232828	0,041139
9-15-3	0	0	0,233194	0,047367
9-18-3	0	0	0,215603	0,039543
9-30-3	0	0	0,214336	0,039888
9-3-3-3	0,421609	0,063632	0,526274	0,069903
9-6-3-3	0,465780	0,186535	0,548739	0,128704
9-6-6-3	0,583452	0,118763	0,674528	0,093646
9-9-9-3	0,473804	0,099875	0,557053	0,178742
9-9-6-3	0,423560	0,148740	0,466307	0,046787
9-9-3-3	0,465702	0,092884	0,378739	0,148684
9-15-3-3	0,557825	0,207456	0,594379	0,068942
9-18-3-3	0,438976	0,167439	0,496721	0,137854
9-18-6-3	0,477848	0,088793	0,558942	0,147655
9-9-9-9-3	0,480056	0,107862	0,575389	0,057845
9-18-9-6-3	0,564386	0,118734	0,626752	0,113856

Tabela 7.2 (b) Desempenho do teste de treinamento e validação utilizando o conjunto B.

Arquitetura	Treinamento	DesvPadrão	Validação	DesvPadrão
9-3-3	0,106538	0,171995	0,374915	0,200599
9-6-3	0,041129	0,118826	0,265115	0,023568
9-9-3	0,001389	0,002957	0,235222	0,026911
9-12-3	0	0	0,231023	0,021164
9-15-3	0	0	0,228406	0,035825
9-18-3	0	0	0,220126	0,031427
9-30-3	0	0	0,228448	0,051073
9-3-3-3	0,496663	0,121077	0,52087	0,03165
9-6-3-3	0,49984	0,132322	0,56004	0,077175
9-6-6-3	0,462247	0,03721	0,57191	0,57191
9-9-9-3	0,268019	0,276911	0,370455	0,194405
9-9-6-3	0,47798	0,111007	0,479233	0,020305
9-9-3-3	0,46043	0,088399	0,522426	0,139519
9-15-3-3	0,376888	0,221581	0,401555	0,006484
9-18-3-3	0,512027	0,103525	0,579647	0,019431
9-18-6-3	0,456817	0,042252	0,54675	0,02564
9-9-9-9-3	0,55669	0,137172	0,657255	0,046266
9-18-9-6-3	0,578153	0,103497	0,65976	0,042723

Tabela 7.2 (c) Desempenho do teste de treinamento e validação utilizando o conjunto C.

Tabela 7.2 Resultado do desempenho encontrado para o teste de treinamento e validação dos conjuntos (a) A, (b) B e (c) C, utilizando a função de aprendizagem **Backpropagation Momentum** com coeficiente de aprendizagem 0,4 e fator de momento 0,5 após o cálculo da média de dez (10) treinamentos e validação.

Analisando os resultados obtidos registrados na tabela acima verifica-se que a arquitetura de rede que apresentou melhor resultado é a que contém dezoito (18) neurônios na camada intermediária.

Repetindo a seqüência acima com o mesmo objetivo de encontrar a melhor arquitetura realizaram-se novos testes de treinamento e validação utilizando os mesmos conjuntos de treinamento e validação e a mesma função de aprendizagem, variando apenas coeficiente de aprendizagem de 0,4 para 0,2 encontramos os resultado apresentados na tabela 7.3 (a), (b) e (c) a seguir após o cálculo da média de dez (10) treinamentos e validação.

Arquitetura	Treinamento	DesvPadrão	Validação	DesvPadrão
9-3-3	0,162454	0,170137	0,352004	0,132752
9-6-3	0,004229	0,008917	0,246753	0,062989
9-9-3	0	0	0,191720	0,003847
9-12-3	0	0	0,180363	0,044845
9-15-3	0	0	0,216588	0,022168
9-18-3	0	0	0,207305	0,031502
9-30-3	0	0	0,211320	0,005811
9-3-3-3	0,409112	0,127376	0,501152	0,052333
9-6-3-3	0,341107	0,165532	0,447440	0,006668
9-6-6-3	0,153640	0,224141	0,320240	0,000658
9-9-9-3	0,068079	0,139273	0,264623	0,203696
9-9-6-3	0	0	0,252402	0,027054
9-9-3-3	0,237294	0,198876	0,348665	0,234668
9-15-3-3	0,066549	0,158102	0,263361	0,018965
9-18-3-3	0,281311	0,252123	0,398401	0,02798
9-18-6-3	0	0	0,225917	0,002899
9-9-9-9-3	0,539545	0,136977	0,546822	0,000728
9-18-9-6-3	0,637783	0,061367	0,665875	0,025062

Tabela 7.3 (a) Desempenho do teste de treinamento e validação utilizando o conjunto A.

Arquitetura	Treinamento	DesvPadrão	Validação	DesvPadrão
9-3-3	0,110619	0,14075	0,304117	0,076041
9-6-3	0,034223	0,097325	0,370725	0,190883
9-9-3	0	0	0,223805	0,035151
9-12-3	0	0	0,194218	0,024892
9-15-3	0	0	0,213568	0,023463
9-18-3	0	0	0,203168	0,023272
9-30-3	0	0	0,213733	0,018549
9-3-3-3	0,315027	0,125943	0,419371	0,120384
9-6-3-3	0,285648	0,186384	0,387356	0,108563
9-6-6-3	0,189254	0,208743	0,358924	0,086389
9-9-9-3	0,019688	0,042147	0,259732	0,033145
9-9-6-3	0,057878	0,168374	0,242639	0,0351
9-9-3-3	0,228528	0,167834	0,297845	0,147839
9-15-3-3	0,107847	0,178357	0,248732	0,047692
9-18-3-3	0,266741	0,226574	0,346914	0,038548
9-18-6-3	0,017937	0,032159	0,26708	0,083632
9-9-9-9-3	0,486721	0,116842	0,494829	0,068528
9-18-9-6-3	0,564528	0,084658	0,558735	0,047845

Tabela 7.3 (b) Desempenho do teste de treinamento e validação utilizando o conjunto B.

Arquitetura	Treinamento	DesvPadrão	Validação	DesvPadrão
9-3-3	0,071271	0,149909	0,388535	0,212846
9-6-3	0,030427	0,096219	0,414030	0,263312
9-9-3	0	0	0,241120	0,040781
9-12-3	0	0	0,223750	0,030533
9-15-3	0	0	0,222295	0,025803
9-18-3	0	0	0,206528	0,017051
9-30-3	0	0	0,205445	0,021643
9-3-3-3	0,355488	0,196831	0,219245	0,029409
9-6-3-3	0,280588	0,201144	0,334855	0,183770
9-6-6-3	0,119524	0,145086	0,249265	0,065895
9-9-9-3	0,013853	0,037730	0,233010	0,027695
9-9-6-3	0,174076	0,204626	0,318998	0,004667
9-9-3-3	0,248812	0,157660	0,351785	0,084421
9-15-3-3	0,234377	0,210013	0,260760	0,028199
9-18-3-3	0,206003	0,206714	0,621980	0,090092
9-18-6-3	0,010082	0,031882	0,247730	0,058718
9-9-9-9-3	0,438687	0,067195	0,514955	0,050933
9-18-9-6-3	0,533033	0,033902	0,604625	0,030681

Tabela 7.3 (c) Desempenho do teste de treinamento e validação utilizando o conjunto (c)

Tabela 7.3 Resultado do desempenho encontrado para o teste de treinamento e validação dos conjuntos (a) A, (b) B e (c) C, utilizando a função de aprendizagem **Backpropagation Momentum,** coeficiente de aprendizagem 0,2 e fator de momento 0,5 após o cálculo da média de dez (10) treinamentos e validação.

Analisando os resultados obtidos registrados na tabela acima verifica-se que as arquiteturas de rede que apresentaram melhores resultados são as que contém doze (12) e dezoito (18) neurônios na camada intermediária.

Da analise dos resultados médios obtidos e conclusões extraídas das tabelas 7.1, 7.2, 7.3 referente ao treinamento de aprendizagem e validação utilizando cento e vinte e seis (126) padrões e três (3) objetos a serem reconhecidos, seguiu-se para a fase de testes de verificação de aprendizagem, selecionando-se as arquiteturas em que foram obtidos os menores erros sobre o conjunto de validação, pois, o modelo selecionado pode estar por demais "ajustado" ao conjunto de validação, tornando-se necessário utilizar um terceiro conjunto, o de teste, para confirmar a estimativa do erro.

O conjunto de teste é formado por quarenta e dois (42) vetores não pertencentes aos conjuntos de treinamento e validação, onde cada vetor é constituído de nove (9) elementos, os quais correspondem a nove características apresentada por cada padrão. A partir deste conjunto de teste formado por uma matriz com quarenta e duas (42) linhas foram criados três (3) conjuntos de testes **A**, **B** e **C**, obtidos a partir do embaralhamento de suas linhas. Os arquivos correspondentes a estes três conjuntos de teste podem ser vistos no apêndice B. Depois de formados os conjuntos de testes, são então realizados a verificação dos desempenhos da rede na fase de testes, como se pode verificar pelos resultados que se seguem.

Com os melhores resultados obtidos na realização de testes com cada um dos conjuntos **A**, **B** e **C** correspondentes ao treinamento e validação foram realizados novos testes com seus respectivos conjuntos de testes para verificar o desempenho, utilizando uma rede neural já treinada com (**9**) neurônios de entrada, três (3) neurônios de saída e ***n*** neurônios na camada intermediária, função de aprendizagem **Backpropagation Padrão** com coeficiente de aprendizagem 0,4 para sessenta mil ciclos (60.000). Nas tabelas 7.4 (a), (b) e (c) a seguir podem ser observados os resultados obtidos da média de dez (10) testes realizados, utilizando os conjuntos de testes A, B e C, respectivamente.

Arquitetura	**Treinamento**	**Validação**	**DesvPadrão**	**Teste**	**DesvPadrão**
9-30-3	0	0,208781	0,022847	0,132603	0,058533
9-6-6-3	0	0,219536	0,031687	0,099644	0,038316
9-9-9-3	0	0,194203	0,042078	0,085276	0,036854
9-9-6-3	0	0,171368	0,037372	0,072731	0,035179

7.4 (a) Resultado do desempenho utilizando o conjunto de teste A.

Arquitetura	Treinamento	Validação	DesvPadrão	Teste	DesvPadrão
9-6-3	0	0,234884	0,015659	0,143872	0,037345
9-9-3	0	0,202635	0,034922	0,156831	0,037255
9-12-3	0	0,221236	0,032129	0,128099	0,036575
9-15-3	0	0,212473	0,024320	0,150661	0,017617
9-18-3	0	0,216963	0,016518	0,128794	0,043010
9-30-3	0	0,213952	0,019975	0,163680	0,023453
9-9-9-3	0	0,203856	0,032081	0,084568	0,024095
9-9-6-3	0	0,200960	0,016021	0,085393	0,012893
9-9-3-3	0	0,198256	0,023989	0,114250	0,053768
9-15-3-3	0	0,195588	0,019331	0,070546	0,010634

7.4 (b) Resultado do desempenho utilizando o conjunto de teste B.

Arquitetura	Treinamento	Validação	DesvPadrão	Teste	DesvPadrão
9-3-3	0	0,263840	0,008580	0,161781	0,038741
9-6-3	0	0,240591	0,033605	0,186892	0,030221
9-9-3	0	0,233621	0,030063	0,150906	0,055255
9-12-3	0,001855	0,178388	0,028079	0,117315	0,094884
9-15-3	0	0,200921	0,024758	0,119154	0,071620
9-18-3	0	0,194870	0,020489	0,116915	0,056767
9-30-3	0	0,213792	0,017330	0,123811	0,061139
9-3-3-3	0,104741	0,259915	0,103358	0,152597	0,174665
9-6-3-3	0	0,186011	0,01689	0,181351	0,105196
9-6-6-3	0	0,197513	0,02604	0,125694	0,032952
9-9-9-3	0	0,194418	0,039382	0,128579	0,050314
9-9-6-3	0	0,187222	0,023511	0,110126	0,041160
9-9-3-3	0	0,182964	0,021524	0,141248	0,085689
9-15-3-3	0	0,178177	0,016370	0,126211	0,056083
9-18-3-3	0	0,166393	0,009829	0,142426	0,052768
9-18-6-3	0	0,178489	0,018500	0,196309	0,147814
9-9-9-9-3	0	0,197154	0,016093	0,190230	0,086099
9-18-9-6-3	0	0,192900	0,016522	0,226530	0,238548

7.4 (c) Resultado do desempenho utilizando o conjunto de teste C.

Tabela 7.4 Desempenho encontrado durante os testes de treinamento, validação e testes realizados utilizando os conjuntos de testes (a) A, (b) B e (c) C com função de aprendizagem **Backpropagation Padrão** com coeficiente de aprendizagem 0,4 após o cálculo da média de dez (10) testes.

Analisando os resultados obtidos registrados na tabela 7.4 (a), (b) e (c) acima utilizando os conjuntos de testes A, B e C, verifica-se que a arquitetura de rede que apresentou melhor resultado é a que contém nove (9) neurônios na primeira camada intermediária e seis (6) neurônios na segunda camada intermediária com o conjunto de teste A.

Após os testes acima realizados, para se fazer uma melhor análise, verificou-se o desempenho com a utilização dos conjuntos de testes **A**, **B** e **C** mudando a função de aprendizagem para **Backpropagation Momentum**, utilizando uma rede neural com (**9**) neurônios de entrada, três (3) neurônios de saída e ***n*** neurônios na camada intermediária, utilizando o coeficiente de aprendizagem 0,4 e com o fator de momento 0,5, mantendo-se o número de ciclos para a realização dos testes foram encontrando os resultados apresentados nas tabelas 7.5 (a), (b) e (c) a seguir após o cálculo da média de dez (10) testes.

Arquitetura	Treinamento	Validação	DesvPadrão	Teste	DesvPadrão
9-12-3	0	0,237246	0,019757	0,090271	0,024012
9-18-3	0	0,223621	0,010133	0,085347	0,028423
9-30-3	0	0,226480	0,011618	0,091196	0,021458

7.5 (a) Resultado do desempenho utilizando o conjunto de teste A.

Arquitetura	Treinamento	Validação	DesvPadrão	Teste	DesvPadrão
9-15-3	0	0,233194	0,047367	0,157838	0,058543
9-18-3	0	0,215603	0,039543	0,151487	0,051487
9-30-3	0	0,214336	0,039888	0,124562	0,055135

7.5 (b) Resultado do desempenho utilizando o conjunto de teste B.

Arquitetura	Treinamento	Validação	DesvPadrão	Teste	DesvPadrão
9-9-3	0,001389	0,235222	0,026911	0,157927	0,058764
9-12-3	0	0,231023	0,021164	0,161999	0,068683
9-15-3	0	0,228406	0,035825	0,181147	0,052266
9-18-3	0	0,220126	0,031427	0,195081	0,078786
9-30-3	0	0,228448	0,051073	0,173278	0,043849

7.5 (c) Resultado do desempenho utilizando o conjunto de teste C.

Tabela 7.5 Desempenho encontrado durante os testes de treinamento, validação e testes realizados utilizando os conjuntos de testes (a) A, (b) B e (c) C com função de aprendizagem **Backpropagation Momentum,** coeficiente de aprendizagem 0,4 e fator de momento 0,5 após o cálculo da média de dez (10) testes.

A partir dos resultados obtidos registrados nas tabelas 7.5 (a), (b) e (c) acima verificam-se que a arquitetura de rede que apresentou melhor resultado é a que contém dezoito (18) neurônios na camada intermediária, utilizando o conjunto de teste A.

Repetindo a seqüência acima para finalizar a seqüência de testes com o objetivo de definir a melhor arquitetura de rede neural a ser utilizada neste trabalho, realizaram-se novos testes utilizando os mesmos conjuntos de testes A, B e C. Na realização dos próximos testes permaneceu

com o uso da mesma função de aprendizagem, variando apenas coeficiente de aprendizagem de 0,4 para 0,2 e foram encontrados os resultado apresentados nas tabelas 7.6 (a), (b) e (c) a seguir após o cálculo da média de dez (10) testes.

Arquitetura	Treinamento	Validação	DesvPadrão	Teste	DesvPadrão
9-9-3	0	0,191720	0,003847	0,097684	0,052735
9-12-3	0	0,180363	0,044845	0,110238	0,051426
9-15-3	0	0,216588	0,022168	0,154413	0,052132
9-18-3	0	0,207305	0,031502	0,104669	0,062538
9-30-3	0	0,211320	0,005811	0,104882	0,071388
9-9-6-3	0	0,252402	0,027054	0,201125	0,154487
9-18-6-3	0	0,225917	0,002899	0,10800	0,02996

7.6 (a) Resultado do desempenho utilizando o conjunto de teste A.

Arquitetura	Treinamento	Validação	DesvPadrão	Teste	DesvPadrão
9-9-3	0	0,223805	0,035151	0,147819	0,065156
9-12-3	0	0,194218	0,024892	0,108554	0,04348
9-15-3	0	0,213568	0,023463	0,116357	0,051238
9-18-3	0	0,203168	0,023272	0,121816	0,046702
9-30-3	0	0,213733	0,018549	0,097103	0,030973

7.6 (b) Resultado do desempenho utilizando o conjunto de teste B.

Arquitetura	Treinamento	Validação	DesvPadrão	Teste	DesvPadrão
9-9-3	0	0,24112	0,040781	0,196399	0,062831
9-12-3	0	0,22375	0,030533	0,221367	0,068582
9-15-3	0	0,222295	0,025803	0,198112	0,068361
9-18-3	0	0,206528	0,017051	0,15586	0,064261
9-30-3	0	0,205445	0,021643	0,173026	0,059936

7.6 (c) Resultado do desempenho utilizando o conjunto de teste C.

Tabela 7.6 Desempenho encontrado durante os testes de treinamento, validação e testes realizados utilizando os conjuntos de testes (a) A, (b) B e (c) C com função de aprendizagem **Backpropagation Momentum,** coeficiente de aprendizagem 0,2 e fator de momento 0,5 após o cálculo da média de dez (10) testes.

Nota-se que, pelos resultados apresentados nas tabelas 7.6 (a), (b) e (c) a rede neural que apresentou melhor resultado foi a arquitetura com nove (9) neurônios na camada intermediária, utilizando o conjunto de teste A, a qual apresentou grande estabilidade durante as duas fases, a de treinamento e validação e a de teste.

Através da análise detalhada dos resultados obtidos nos vários testes de treinamento, validação e teste observados nas tabelas 7.2 a 7.6 apresentadas neste capítulo concluímos que foi encontrados um melhor desempenho e estabilidade quando se utilizou a arquitetura de rede neural *Perceptron* Múltiplas Camadas (MLP) com nove (9) neurônios na camada de entrada, nove (9) neurônios na camada intermediária e três (3) neurônios na camada de saída, com uma taxa de aprendizagem de 0,2 e fator e momento 0,5, optando-se pelo uso da função de aprendizagem **Backpropagation Momentum**, utilizando o número total de nós do SPP3.

Continuando na fase de testes foi apresentado à rede neural um padrão de cada vez, os quais também não pertenciam ao conjunto de treinamento para verificar o seu reconhecimento. Verificamos que quando da apresentação de um único padrão semelhante aos existentes no banco de dados estes foram reconhecidos, apresentando um taxa de reconhecimento de 95% em tempo real, confirmando os resultados obtidos na fase de teste indicado na tabela 7.6 acima.

Com relação ao tempo de processamento verificamos que este está proporcionalmente ligado ao número de processos (tarefas), pois o ideal para o processamento paralelo é que o número de tarefas a serem realizadas, seja igual ao número de nós de processamento. Logo, devemos analisar a viabilidade de se desenvolver um hardware (aumentar o número de nós de processamento), caso seja viável dentro da aplicação.

Analisando o tempo de treinamento verificamos que este está diretamente ligado ao número de ciclos previstos para o processo.

Verificamos que a maior parte do tempo gasto na realização do processamento é gasto na fase de pré-processamento e extração de características, pois se trabalha com um número consideravelmente elevado de padrões.

Quanto ao algoritmo paralelo apresentado no item 7.4.3 que foi implementado utilizando o modelo mestre/escravo de computação paralela. Para desenvolver as tarefas de computação utilizando este modelo divide-se o processo em mestre e escravo. O processo mestre atua como controle, sendo responsável pela criação e envio de processos e pela coleta e fornecimento dos resultados. Os processos escravos efetuam os processamentos de acordo com as suas cargas alocadas pelo processo mestre.

O tempo de treinamento das implementações seqüencial e paralela, variando-se as quantidades de padrões apresentados para treinamento pode ser verificado através da tabela 7.7 a seguir.

IFSC-USP SERVIÇO DE BIBLIOTECA INFORMAÇÃO

Número de Amostras	Número de Ciclos	Seqüencial (Seg)	Paralelo 3 nós (Seg)	Speed-up	Efici-ência	Paralelo 7 nós (Seg)	Efici-ência	Speed-up
168	120.000	1510,00	540,76	2,79	0,93	225,32	0,95	6,7
118	120.000	666,37	237,00	2,81	0,93	98,80	0,95	6,7
84	60.000	147,00	52,85	2,78	0,93	21,90	0,95	6,7
42	60.000	95,54	32,13	2,97	0,93	13,39	0,95	6,7

Tabela 7.7 Resultado do speedup para os testes de treinamento variando-se o número de amostras e número de nós de processamento.

O desempenho ou speed-up de um sistema é proporcional ao número de nós de processamento, ou seja, quanto mais próximo o número de nós em relação ao número de tarefas a serem realizadas, melhor será o speed-up, logo podemos ser constatar analisando a tabela 7.7 acima que o resultado alcançado pelo algoritmo utilizado neste trabalho alcançou um bom resultado.

CAPÍTULO 8

8 - CONCLUSÕES

8.1 - CONSIDERAÇÕES FINAIS

- O sistema apresentou características de invariância geométrica ao tratar com imagens de entrada, reconhecendo peças transladadas, escalonadas e rotacionadas em relação à posição original dos exemplos utilizados para treinamento. Logo, permitindo o seu uso em ambientes onde as peças a serem manipuladas apresentem posição ou forma diferente daquelas apresentadas para o treinamento.
- Apresentou características para tarefa de manipulação em tempo real, contribuindo para a melhora da automação de montagem e manipulação de peças.
- Os tempos de processamento envolvidos demonstraram a viabilidade e aplicação e utilização do sistema para aplicações em tempo real, utilizando o multicomputador SPP3, atendendo a tempos de processamento que justificam a implementação.
- O sistema vem a atender a aplicações nos casos em que o tempo de realização das tarefas venha a ser de grande importância.
- O sistema desenvolvido comprovou ser muito superior aos até então desenvolvidos em sistemas distribuídos (clusters), uma vez que a taxa de transferência de dados e troca de mensagem é bem melhor.

8.2 - CONTRIBUIÇÕES

Neste trabalho, foi possível mostrar que ao se efetuar uma escolha criteriosa de poucas características aliadas à utilização de redes neurais é possível se obter uma alta taxa de reconhecimento, e quando processada paralelamente, atingir excelentes resultados no tempo de processamento, quando comparado com máquinas seqüenciais.

A contribuição inédita e de grande relevância deste trabalho foi ter mostrado a possibilidade de se obter um alto desempenho no reconhecimento de padrões independente do tamanho da amostra (imagem ou objeto), pois não se trabalha com a dimensão da imagem em termos de pixel e sim com o mínimo de características extraídas da mesma, criteriosamente selecionada de forma que estas venham a caracterizar precisamente o objeto a ser reconhecido. Desta forma vem contribuir para a solução de situações complexas em que até então se tem trabalhado com uma massa excessiva de dados, dificultando a obtenção da informação em tempo real.

Outra grande contribuição foi apresentar mais uma possibilidade de uso dos multicomputadores na realização de tarefas de reconhecimento de padrões.

Outra contribuição apresentada pelo trabalho é a possibilidade do sistema trabalhar em tempo real no reconhecimento de imagens (objetos) e possibilitar o trabalho concorrente para o desenvolvimento de novos projetos.

Este trabalho vem a chamar a atenção dos desenvolvedores de projetos que envolvam o reconhecimento de padrões, quando da implantação de situações onde envolve grande poder computacional.

8.3 - SUGESTÕES PARA TRABALHOS FUTUROS

- Continuar a implementação utilizando o robô (manipulador mecânico) industrial localizado no chão de fábrica do laboratório da engenharia mecânica da EESC-USP (Escola de Engenharia de São Carlos – Universidade de São Paulo).
- Utilizar o sistema implementado fazendo uso do multicomputador SPP3 para projetar sistemas de alto desempenho para o uso em mecanismos de manipulação ou em sistemas de inspeção visual automática, pois estes mecanismos tem despertado grande interesse na área industrial por prover aumento na produtividade, ganhos econômicos, eficiência e um melhor processo de controle de qualidade na fabricação de um produto.
- Implementar o sistema para operações em 2D, trabalhando com algoritmos para o processamento de pontos conjugados em estéreo. Utilizar os recursos de computação gráfica que permitam trabalhar em 3D, objetos translúcidos e com pontos não visíveis de objetos opacos, para que estes não venham a ser projetado nos planos da imagem.
- Estender a aplicações sistemas utilizando redes neurais fuzzy. A utilização da lógica *fuzzy* proporcionará uma grande vantagem, pois o seu procedimento é paralelo, o que permite a sua implementação em processadores paralelos. Em aplicações dedicadas onde a combinação complexa do relacionamento entre as variáveis *fuzzy* sejam necessárias, aplicando-se a lógica *fuzzy é* possível extrair cada atributo (perceptivas de tonalidade, distância, tamanho e orientação, bem como o relacionamento entre elas, etc.) de uma imagem, montar suas respectivas funções de pertinência, relacioná-las e agregá-las de modo a aplicar idéias inspiradas na percepção humana em um sistema de visão computacional. A combinação com cada antecedente da

regra com um fato de entrada quando computado de forma paralela torna o procedimento de inferência *fuzzy* apropriada para implementação em processadores paralelos. A flexibilidade proporcionada pela lógica *fuzzy* permite que vários critérios de decisão possam ser usados para unir diferentes objetivos e com sua implementação é possível simplificar o desenvolvimento, manutenção e experimentação.

- Implementar o sistema de manipulação para comando remoto de manipulação, via internet, para área industrial, médica e militar.
- Desenvolver projetos que exijam a ampliação dos módulos do SPP3, utilizando o conceito de redes neurais associadas às estruturas de paralelismo implementado nesta tese.
- Implementar este sistema para aplicações que exijam alto poder computacional. A utilização de imagens com alta densidade de pixel acarreta um aumento considerável do tempo de processamento dos dados no estágio de pré-processamento. Com o uso das redes neurais e da implementação paralela este tempo de pré-processamento da imagem é reduzido drasticamente.
- Realizar trabalhos que envolvam classificação e análise de imagens na área médica em diagnóstico por imagens.
- Desenvolver trabalhos em sistema de segurança utilizando identificação baseada em padrões de íris ou impressões digitais.
- Projetar sistemas de inspeção visual de tarefas como: soldagem, pintura, controle de qualidade, furação, montagens, etc.
- Desenvolver trabalhos para área militar, na localização e identificação de alvos. Nesta área o fator tempo de processamento é o de maior importância vindo a ser imprescindível a utilização de recursos que venham a contribuir para a redução do mesmo.
- Fazer um trabalho explorando a teoria de momentos invariantes de Hu. A escolha adequada dos invariantes de momento de Hu referente às características de um objeto proporciona a redução do tempo de processamento. Durante o processo de digitalização, transformações realizadas devido à rotação, translação, escalonamento, e à razão de aspecto da imagem, surgem deformações, que leva os invariantes de momento de Hu extraídos apresentarem uma distribuição dentro de um valor máximo e mínimo, com uma média e desvio padrão. Como os invariantes de Hu são construídos através da combinação de momentos centrais, sua soma faz com que várias características de um mesmo objeto sejam recombinadas dentro de uma mesma característica, favorecendo dessa maneira uma representação única do objeto.

- Implementar um hardware dedicado para uma implementação específica. Desenvolver nova versão do SPP3 com uma estrutura de hardware dedicada a uma nova implementação que exija uma maior velocidade de processamento.
- Realizar esta implementação em outros computadores paralelos. Com a realização de uma análise comparativa do desempenho entre arquiteturas, um novo projeto poderá ser proposto para uma nova arquitetura com um melhor desempenho.
- Fazer esta implementação sem o uso de biblioteca de comunicação de passagem de mensagens MPI (Message Passing Interface) e PVM (Paralell Virtual Machine) e levantar um parâmetro de comparação. Através do uso de uma plataforma específica para programação paralela a ser utilizada no desenvolvimento de programas paralelos que agilizem o desenvolvimento de sistemas nas diversas áreas científicas que demandam soluções paralelas.
- Desenvolver um conjunto de ferramentas para facilitar o processo de paralelização utilizando o ambiente de desenvolvimento Delphi para Linux, o Kylix.

BIBLIOGRAFIA

[ALMASI, 1994] ALMASI, G.S.; GOTTLIEB, A. Highly parallel computing. The Benjamin Cummings Publishing Company, Inc., 1994.

[AMORIM, 1998] AMORIN, C.L.; BARBOSA, V.C.; FERNANDES, E.S.T. Uma introdução à computação paralela e distribuída. VI Escola de Computação, Campinas, UNICAMP, 1988.

[ANAND, et al, 1993] ANAND, R.; MEHPOTRO, K.; MOHAN, C. K.; RANKA, S. Analysing images containing multiple sparse patterns with Neural networks. Pattern Recognition, v. 26, n. 11, 1993.

[ANDERSON, 1998] ANDERSON, D.;CHASE, J.; GADDE, S.; GALLATIN, A.; YOCUM, K. and FEELLEY, M. Cleating the I/O Bottleneck: Network Storage with Trapeze/Myrinet. Proceeding of the 1998 USENIXTechinical Conference, New Orleans, June, 1998.

[ANDREW, 1983] ANDREWS, G.R.; SCHNEIDER, F. B. Concepts and Notations for Concurrent Processing. ACM Computing Surveys, v. 15, p. 3-43, 1983.

[ANSI, 1998] ANSI/VITA 26-1998. , 1988.

[BALLARD & BROWN, 1982] BALLARD, D. H.; BROWN, C. M. "Computer Vision" Prentice-Hall INC, New Jersey, 1982.

[BARAK, 1998] BARAK A.; LA`ADAN, O. The MOSIX Multicomputer Operating System for High Performance Cluster Computing. Journal of Future Generation Computer Systems, vol. 13, n.4-5, p. 361-372, march, 1998.

[BEALE and JACKSON, 1990] BEALE, R., JACKSON, T. " Neural Computing: an Introduction" - Adam Hilger,Bristol, 1990.

[BERKELEY, 1998] Berkeley Network of Workstations (NOW). , 1998.

[BLECH, 1994] BLECH, R.A., "Na overview of parallel processing", Slides, Parallel Computing with PVM Workshop, Nasa Lewis Research Center, http://www.lerc.nasa.gov/Other _Groups/IFMD/2620/tutorialPP.html, 1994.

[BODEN, 1995] BODEN, N.J. Myrinet – A Gigabit-per-Second Local Area Network myricom Inc., , 1995.

[BOGNI & MARRONE, 1991] BOGNI, C. J.; MARRONE, L. Arquitectura de Computadores. Rio de Janeiro, V Escola Brasileira – Argentina de Informática, 1991.

[BORGES, et al, 1994] BORGES, D.L.; ORR, M.J.; FISHER, R. B. A Radial Basis Function Neural Networks for Parts Identification of three dimensional shapes. VII SIBGRAPI, Anais, 1994.

[BROWN & HARRIS, 1994] BROWN, M.; HARRIS, C. "Neurofuzzy Adaptative Modelling and Control", Prentice Hall, New York, 1994.

[BRONSON, 1990] BRONSON, E. C.; CASAVANT, T.L.; JAMIESON, L. H. Experimental Application-driven Architecture Analysis of na SIMD/MIND Parallel Processing System, IEEE Transaction on Parallel and Distributed, v. 1, n.2, april, 1990.

[CALLAHAN, 1988] CALLAHAN, D.; COOPER, K.; TOREZONI, L. Parascope, a parallel programming environment. J. Supercomputing Application, v.5, Winter, 1988.

[CARVALHO, 1995] CARVALHO, A. C. P. L. F. A modular neural architecture for pattern recognition. II Simpósio Brasileiro de Redes Neurais, Anais, 1995.

[CHEN et al, 1995] CHEN, Y.Q.; NIXON, M. S.; THOMAS, D. W. "Statical Geometrical Features for Texture Classification", Pattern Recognition, vol.28, nº 4, pp.537-552, 1995.

[CLARK, 1997] CLARK, E. The PCI Connection. Network Magazine, p. 69-73, september, 1997.

[CHIN & DYER, 1986] CHIN, R. T.; DYER, C. R. Model-based recognition in robot vision. Computing Survey, v. 18, n. 1, 1986.

[CHTIOUI, et al, 1996] CHTIOUI, Y.; BERTRAND, D.; DEVAUX, M. F.; BARBA, D. Aplication of a hybrid neural network for discrimination of seeds by artificial vision. Proceeding of the International Conference on Tools with Artificial Inteligence, 1996.

[CORNELL, 1996] CORNELL Theory Center, "Introduction to parallel processing", Virtual Workshops, 1996.

[CRAY, 1995] CRAY T3D system architecture overview. Cray Research Inc., , 1995.

[DAS, 1998] DAS – The Distributed ASCI Supercomputer. , 1998.

[DAVIES & GILL, 1993] DAVIES, J. L.; GILL, K. F. Machine Vision and Automated Assembly. Mechatronics, Vol 3, n. 4, 1993.

[DUDA el al, 1979] DUDA, R.O.; NITZAN, D. and BARRET, P. "Use of Range and Reflectance Data to Find Planar Surface Regions" , IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-1, nº 3, July, pp. 259-71, 1979.

[DUDA & HART, 1973] DUDA, R.O.; HART, P.E.H. "Pattern Classification and Scene Analysis", John Wiley & Sons, New York, 1973.

[DUNCAN, 1990] DUNCAN, R.A. A Survey of Parallel Computer Architectures. IEEE

Computing Magazine, v. 23, n. 2, p. 5-16, february, 1990.

[FELDERMAN, 1994] FELDERMAN, R. ATOMIC: A High-Speed Local Comunication Architecture. Journal of High Speed Networks 1, p 1-28, 1994.

[FLORISI, 1994] FLORISI, D. and YEMINI,Y. Protocols for Loosely Synchronous Networks. IFIP/IEEE Protocols for High-Speed Networks, 1994.

[FLYNN, 1966] FLYNN,M.J.,Very High Speed Computing Systems Proc. IEEE, vol. 54, p. 1901-1909, 1966.

[FLYNN, 1972] FLYNN, M.J., "Some computer organizations and their effectiveness", IEEE Transactions on Computers, vol. C-21, p. 948-960, 1972.

[FOSTER, 1995] FOSTER, I. Designing and Building Parallel Programas: concepts and tools for parallel software engineering. Addison-Wesley Publishing Company, Inc., 1995.

[FU, 1974] FU, K.S. "Syntactic Methods in Pattern Recognition", Academic Press, New York, 1974.

[FUKUSHIMA, 1983] FUKUSHIMA, K. Neocognitron: A neural network model for a mechanism of visual pattern recognition. IEEE Transactions on Systems, Man and Cybernetics, SM-13(5), p. 826-834, 1983.

[FUKUSHIMA, 1985] FUKUSHIMA, K. Cognitron: a self-organizing multilayered neural networks. Biological Cybernetics, 20, p. 121-136, 1985.

[FUKUSHIMA, 1989] FUKUSHIMA, K. Analysis of the process of visual pattern recognition by the neocognitron. Neural Networks, v. 2, n. 6, 1989.

[GALLANT, 1994] GALLANT, S. I. "Neural Network Learning and Expert Systems" The MIT Press, Massachusetts, 1994.

[GEIST, 1994] GEIST, G.; BEGUELIN, A.; DONGARRA, J.; JIANG, W.; MANCHECK, R.; SUNDERAM, V. PVM: Parallel Virtual Machine – A user`s guide and tutorial for network parallel computing, The MIT Press, 1994.

[GILES, et al, 1988] GILES, G. L.; GRIFFIN, R. D.; MAXWELL, T. Encoding geometric invariance in higher-order neural networks. Neural Information Processing Systems, 1988.

[GILMARC & KLEIN, 1986] GILMARC, C. E.; MILUTINOVIC, V. M. A survey RISC, 1986.

processors and computers in the mid-1980s. IEEE Computer, 20(9), p. 59-69, september, 1986.

[GONZALEZ & WINTZ, 1987] GONZALEZ, R. C.; WINTZ, P. "Digital Image Processing", Addilson-Wesley Publishing Company, 2nd. edition, 1987.

[GONZALEZ & WOODS, 1993] GONZALES, R.C.; WOODS, R.E. Digital Image

Processing, Addison-Wesley, 1993.

[GUPTA, et al, 1989] GUPTA, L.; SAYEH, M. R.; TAMMANA, R. A neural network approach to robust shape classification. Pattern Recognition, v. 23, n. 6, 1989.

[HARALICK et al, 1973] HARALICK, R. M.; SHANMUGAM, K.; DINSTEIN, It'shak, "Textural Features for Image Classification", IEEE Transactions on Systems, Man and Cybernetics, vol.SMC-3, n°. 6, November, pp.610-621, 1973.

[HARALICK, 1979] HARALICK, R. M. "Statical and Approaches to Texture", Proceedings of the IEEE, vol.67, n°. 5, May, pp.786-804, 1979.

[HERTZ et al, 1993] HERTZ, J.; KROGH, A.; PALMER, R. G. "Introduction to the Theory of Neural Computation", Addison_ Wesley Publishing Company, June, 1993.

[HOCKNEY & JESSHOPE, 1988] HOCKNEY, R. W.; JESSHOPE, C. R. Parallel Computers 2. Philadelphia, IOP, 1988.

[HWANG, 1987] HWANG, K.; BRIGGS, F.A. Computer Architecture and Parallel Processing, McGraw-Hill, 1984.

[HWANG,1993] HWANG, K. Advanced computer architecture: parallelism, scalability, programmability. MacGraw-Hill, Inc., 1993.

[HWANG & TSENG, 1993] HWANG, J. N.; TSENG, Y. H. 3D motion estimation using simple perspective sparse range data via surface reconstruction neural networks. International Conference on Neural Networks, Proceedings, 1993.

[HEBREW, 1998] Hebrew University of Jerusalem – Computer Science Institute. MOSIX-Multicomputer Operating System for UNIX. , 1998.

[IBM, 1993] IBM solutions: Scalable Power Parallel System 1. IBM Corporation, , 1993.

[IBM, 1995] IBM parallel environment for AIX version: operation and use, v. 2, p. 111-190, august, 1995.

[ILLIONOIS, 1998] Illinois Fast Messages (FM). , 1998.

[INTEGRATED, 1998] Integrated Concurrent and Distributed Computation – Mississipi State University, , 1998.

[JANG & SUN, 1995] JANG, J. R.; SUN, C-T. "Neuro-Fuzzy Modelling and Control" The Proceendings of the IEEE, vol. 83, n°. 3, pp. 378-406, March, 1995.

[KARATHANASSI, et al, 1996] KARATHANASSI, V.; IOSSIFIDIS, C.; ROKOS, D.

Application of machine vision techiques in the quality control of pharmaceutical solutions, Computers in Industry, v. 32, n. 2, 1996.

[KARVÉ, 1997] KARVÉ, A. ATM in the Fast Lane. Network Magazine, p.48-62, july, 1997.

[KENNEDY, 1992] KENNEDY, K. Software for supercomputers of the future. Journal of Supercomputing, v. 5, p. 251-262, september, 1992.

[KIRNER, 1989] KERNER, C., "Sistemas operacionais para ambientes paralelos", IX Congresso da SBC, Anais da VIII Jornada de Atualização em Informática, 1989.

[KIRNER, 1991] KIRNER, C. Arquiteturas de sistemas avançados de computação. In: Anais da Jornada EPUSP/IEEE em Sistemas de Computação de Alto Desempenho, p. 307-353, 1991.

[KOHL, 1997] KOHL, J. XPVM: A graphical console and monitor for PVM, , 1997.

[KOWALCZYK & FERRA, 1994] KOWALCZYK, A.; FERRA, H. L. Developing Higher-order Networks with empirically selected units. IEEE Transactions on Neural Networks, v. 5, n.5, 1994.

[LEJEUNE & SHENG, 1993] LEJEUNE, C.; SHENG, Y. Optoneural system for invariant pattern recognition, Journal of Physics, v. 71, n. 9-10, 1993.

[LIBOUDEC, 1992] LIBOUDEC, J.Y. The Assynchronous Transfer Mode: a Tutorial. Computer Network and ISDN System, n. 24, p. 279-309, 1992.

[LIEIRA, 1995] LIEIRA, J.; MOREIRA, E.D.S. Utilização de Som e Imagem em Sistemas de Gerenciamento de Redes de Computadores, 1995.

[LIN & CUNNINGHAM III, 1995] LIN, Y,; CUNNINGHAM III, G. A. - "A New Approach to Fuzzy-Neural System Modeling" - IEEE Trans. On Fuzzy Systems, vol.3, nº 2, May, pp. 190-197, 1995.

[LIN & LU, 1995] LIN, C-T.; LU, Y-C. "A Neural Fuzzy Systems with Linguistic Teaching Signals", IEEE Trans. On Fuzzy Systems, vol.3, nº 2, May, pp. 169-189, 1995.

[LINUX,1998] LINUX Parallel Howto. , 1998.

[MACDOWELL, 1989] MACDOWELL, C.; HELMBOLD, D. Debugging concurrent programs. ACM Computing Magazine, v.21, p. 593-622, dezembro, 1989.

[MACHADO, 1993] MACHADO, A. B. M. Neuroanatomia funcional, São Paulo, Atheneu, 1993.

[MARTINS, 1994] MARTINS, W. The Improvement of Classifiers based on Weightless Neural Networks, Phd thesis, University of York, 1994.

[MASTERS. 1993] MASTERS. T. "Practical Neural Network Recipes in C++ Academic Press, New York, 1993.

[MASTERS, 1994] MASTERS, T. "Signal and Image Processing with Neural Networks a C++ Sourcebook", John Wiley & Sons, Inc., New York, 1994.

[McCULLOCH & PITTS, 1943] McCULLOCH, W.; PITTS, W. A logical calculus of the ideas immanent in nercous activity, Bulletin of Mathematical Biophysics, 1943.

[MERLIN, 1994] MERLIN, J. IDA – User's guide, Southampton: University of Southampton, 1994.

[MITZIAS & MERTZIOS, 1994] MITZIAS, D. A.; MERTZIOS, B. G. Shape recognition with neural classifier based polygon approximation tecnique. Pattern Recognition, v. 27, n.3, 1994.

[MYRINET, 1998] Myrinet Customer Projects. , 1998.

[MPI, 1995] EPCC MPI course. EPCC Training and Education Center. , 1995.

[NAJJARI, 1995] NAJJARI, H.; STEINER, S.S. Knowledge-based object recognition technique for robots assembly application. Proceedings of the SPIE, vol. 2588, USA, 1995.

[NAVAUX, 1989] NAVAUX, P.O.A. Introdução ao Processamento Paralelo. Revista Brasileira de Computação, v.5, n. 2, p. 31-43, out/dez, 1989.

[NAYAR, et al, 1996] NAYAR, S.K.; NENE, S.A.; MURASE, H. Subspace methods for robotic vision, IEEE Transactions on Robotic Automation, v. 12, n. 5, 1996.

[NETSCAPE, 1998] Netscape, Netscape Netcener, , 1998.

[NETSTATION, 1997] Netstation Project. University of Southern California. , 1997.

[OGÊ, 1999] OGÊ, M. F.; NETO, H.V. Processamento Digital de Imagens, Brasport Livro e Multimídia Ltda., Rio de Janeiro, 1999.

[PARKER, 1994] PARKER, J.R. "Practical Computer Vision Using C", John Wiley & Sons, Inc., New York, 1994.

[PEREZ, 1996] PEREZ, P.A.S. P2D – Um ambiente de auxílio à paralelização de aplicações Fortran, São Carlos: Universidade de São Paulo, 1996.

[PRECHELT, 1994] PRECHELT, L. "PROBEN1 – A set of Neural Network Benchemark Probens and Benchmarking Rules – Universidad de Karlsruhe Relatório Técnico – Alemanha, 1994.

[PRATT, 1991] PRATT, W. K. "Digital Image Processing", A Wiley-Interscience Publication, 1991.

[QUINN, 1987] QUINN, M.J., Designin Efficient Algorithms for Parallel Computers, McGraw Hill, 1987.

[RAUBER, 1997] RAUBER, T. W. "Pattern Recognition", XVII Congress of The Brazilian Computer Science Society, Brasília, DF, http://www.inf.ufes.br/~thomas 1997.

[REID, et al, 1989] REID, M. B.; SPIRKOVSKA, L.; OCHOA, E. Simultaneous position, scale and rotation invariant pattern classification using third-order neural networks International Journal Neural Networks, v. 1, 1989.

[RUMELHART, et al, 1986] RUMELHART, D.E.; HINTON, G.E.; WILLIAMS, R.J. Learning internal representations by Short-Term Goal Identification. International Journal of Advanced Manufacturing Technology, v. 4, 1986.

[SCHALKOFF, 1989] SCHALKOFF, R. J. "Digital Image Processing and Computer Vision", John Wiley & Songs, INC, New York, 1989.

[SEITZ, 1998] SEITZ, C.; COHEN, D. M2M-DUAL-SW8: Dual 8-Port Myrinet-SAN Switch. Myricon Inc., - sw8.html, 1998.

[SNOW, 1992] SNOW, C.R., Concurrent Programming, Cambrige University Press, 1992.

[SNNS, 1995] SNNS – Stuttgart Neural Network Simulator User Manual Version 4.1, 1995.

[SONG & CHANG, 1996] SONG, K. T.; CHANG, J. M. Experimental study on robot visual tracking using a neural controller, Industrial Eletronics Conference, vol. 3, 1996.

[SPIRKOVSKA & REID, 1990] SPIRKOVSKA, L.; REID, M. B. Connectivity strategies for higher-order neural networks applied to pattern recognition. International Joint Conference on Neural Networks, Proceedings, San Diego, CA, 1990.

[SPIRKOVSKA & REID, 1993] SPIRKOVSKA, L.; REID, M. B. Coarse Coarse Coading Higher-Order Neural Networks for PSRI Object Recognition. IEEE Transaction on Neural Networks, v. 4, n. 2, 1993.

[TANENBAUM, 1992] TANENBAUM, A.S. Modern operating systems, Prentice Hall International, 1992.

[TANENBAUM, 1997] TANENBAUM, A. S. Computer Network, Prentice-Hall, inc., 1997.

[TRINDADE, 1994] TRINDADE, O.; MARQUES, E.; JEUKENS, I. A Parallel

Architecture Based on Personal Computers – An Overview, University of São Paulo, USP, São Carlos, Brazil, 1994.

[TRINDADE, 1995] TRINDADE, O.; MARQUES, E.; JEUKENS, I. A parallel architecture based on personal computers – requeriments and definitions – na overveiw in: XV International Conference of the Chilean Computer Society, p. 479-490, November, 1995.

[TRUXEL, 1988] TRUXEL, S.K.; ROGERS, S.K.; KABRISKY, M. The use of neural networks in PSRI recognition. Proc. Joint International Conference Neural Networks, San Diego, CA, July, p. 24-27, 1988.

[TSAO, et al, 1993] TSAO, C. K. E.; LIN, W. C.; CHEN, C. T. Constraint satisfaction neural networks for image recognition.Pattern Recognition, v. 26, n. 4,1993

[VARELLA, et al, 1994] VARELLA, L.E.; PASSOS, E. P. L.; SANTOS, M. A.; ARAUJO, R. L. Uma solução back-propagation invariante a escala, rotação e translação para o reconhecimento de caracteres. I Congresso Brasileiro de Redes Neurais, Itajubá-MG, 1994.

[VASCONCELOS, 1995] VASCONCELOS, G. C. Redes Neurais e Reconhecimento de padrões, II Simpósio Brasileiro de Redes Neurais, São Carlos, 1995.

[WAARD, 1994] WAARD, W. P. Neural techiniques and postal code detection. Pattern Recognition Letters, v. 15, 1994.

[WANG, & COHEN, 1994] WANG, J. Y.; COHEN, F. S. 3-D object recognition and shape estimation from image countours using B-spline, shape invariant matching, and neural networks, IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 16, n 1, 1994.

[WELSH, 1994] WELSH, M. Linux installation and getting started. LDP – LINUX Documentation Project, , 1994.

[WHY, 1998] Why Myrinet? , 1998.

[WOLF, 1995] WOLF, S. A beginners's guide to the IBM SP2. Knoxville University of Tennesse – Joint Institute of Computational Science, september, 1995.

[YAN & WU, 1994] YAN, H.; WU, J. Character and line extraction from color map images using a multi-layer neural network. Pattern Recognition Letter, v. 15, 1994.

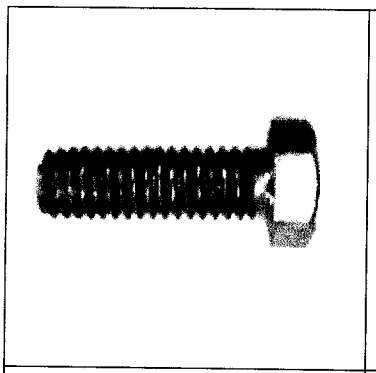
[YUCEER & OFLAZER, 1993] YUCEER, C.; OFLAZER, K. A rotation, scaling and translation invariant pattern classification system, Pattern Recognition, v. 26, n. 5, 1993.

[ZALUSKA, 1991] ZALUSKA, E.J., "Research lines in distributed computing systems and concurrent computation", Anais do Workshop em Programação Concor., Sist. Distribuídos e Engenharia de Software, p. 132-155,1991.

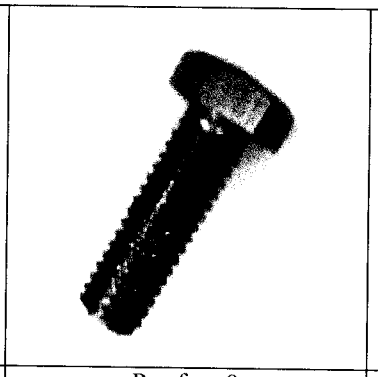
[ZHANG, et al, 1991] ZHANG, W.; HASEGAWA, A.; ITOH, K.; ICHIOKA,Y. Error backpropagation with minimum-entropy weights: a technique for better generalization on 2D shift invariants. Internation al Joint Conference on Proceeding, p. 645-Neural 648, 1991.Networks,

APÊNDICE A

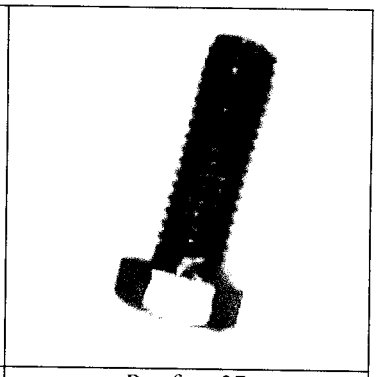
IMAGENS DE OBJETOS UTILIZADOS



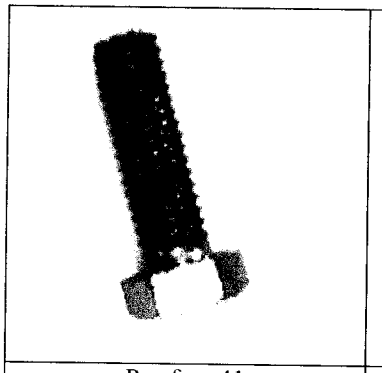
Parafuso 2



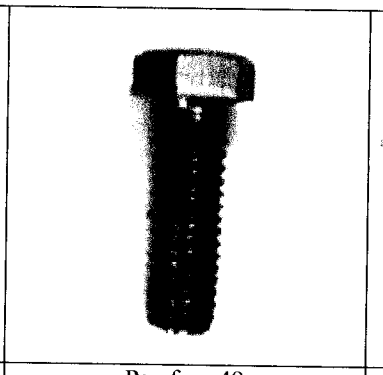
Parafuso 9



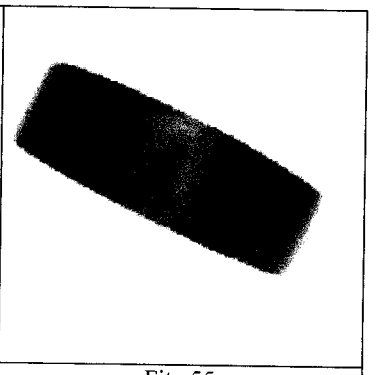
Parafuso 37



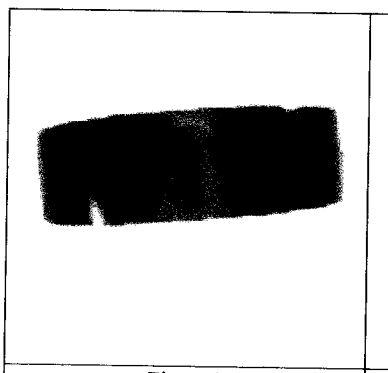
Parafuso 41



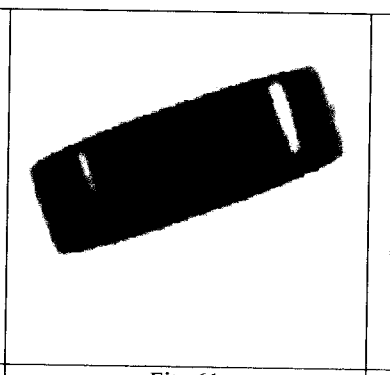
Parafuso 49



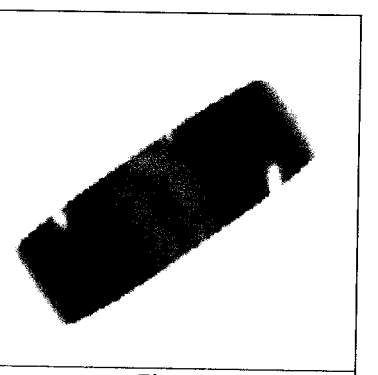
Fita 55



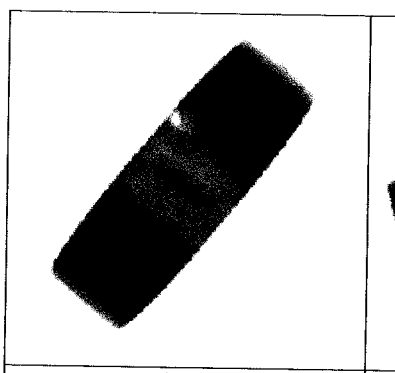
Fita 59



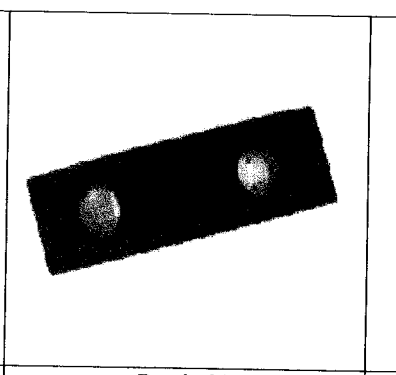
Fita 61



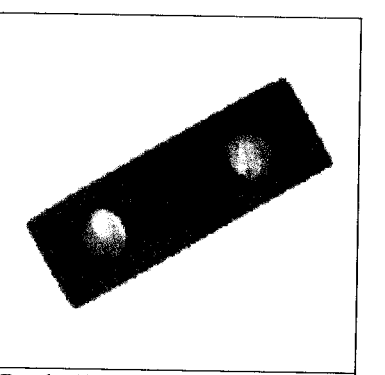
Fita 63



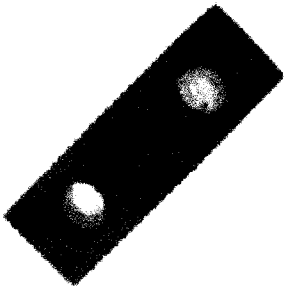
Fita 66



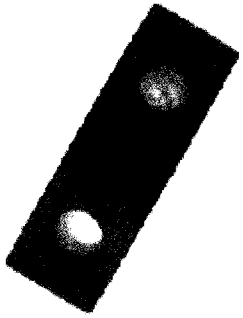
Parale 39



Parale 42



Parale 43

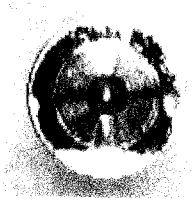


Parale 46

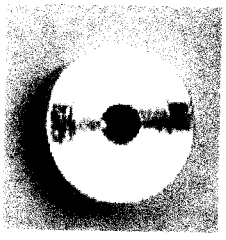
Parale1d69

Parale2e74

Parale2c72

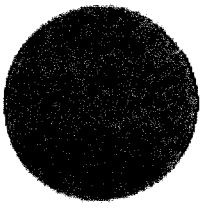


Cilcc32

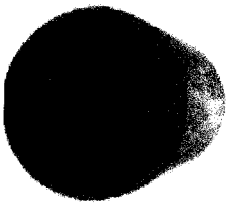


Cild35

Cilmad1c13

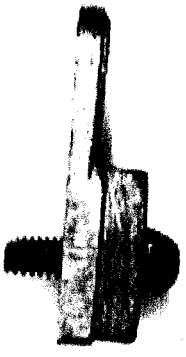


Cilmadc2

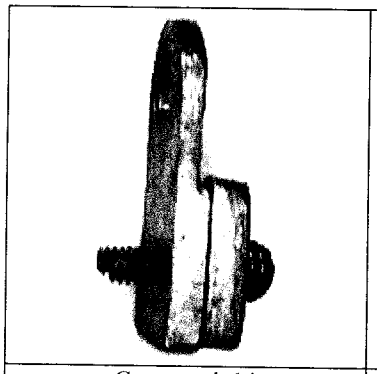


Cilmad7

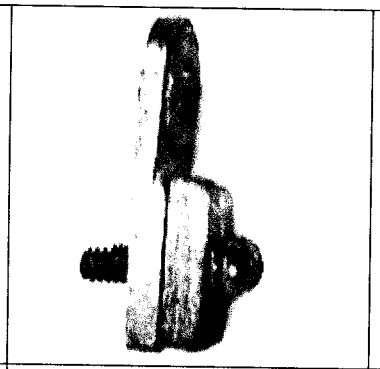
Conector



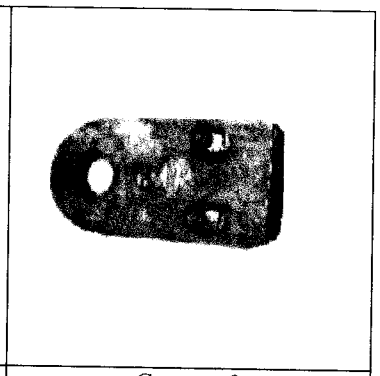
Conectorcp11



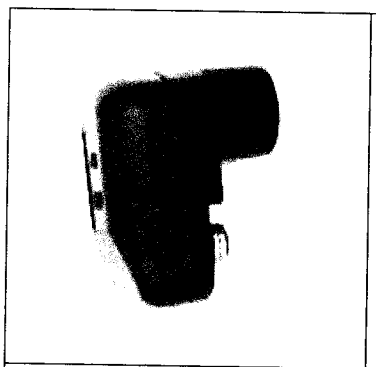
Conectordp14



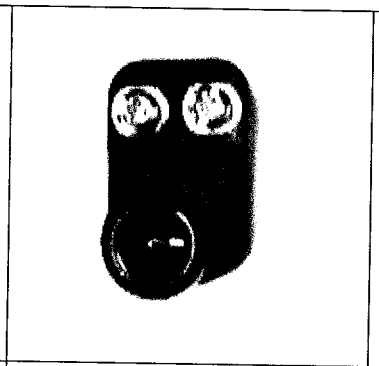
Conectorep17



Conector8

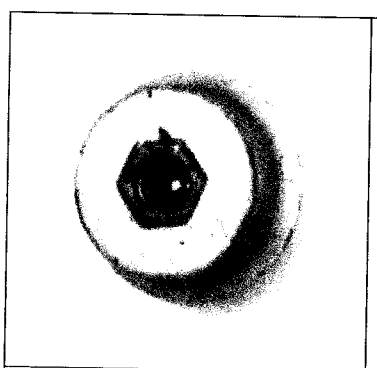


ConectorTVc23

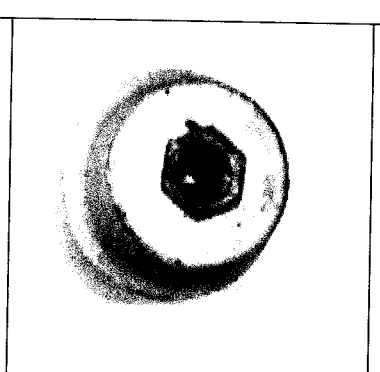


ConectorTVe41

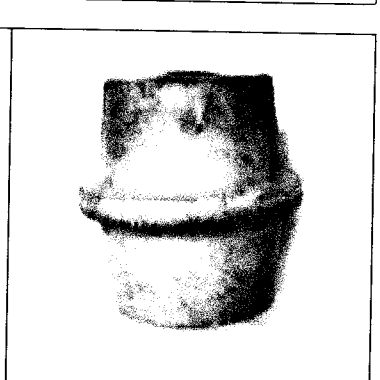
ConectorTVc



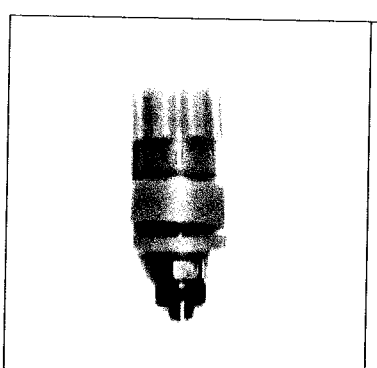
Isolador17e



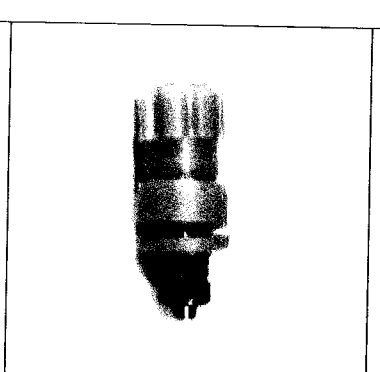
Isolador18d



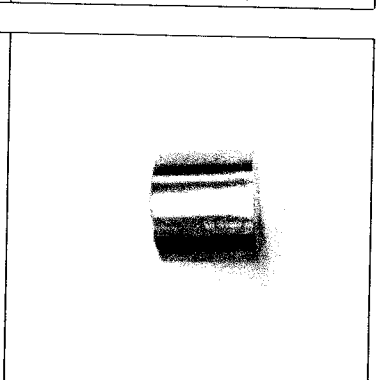
Isolador22e



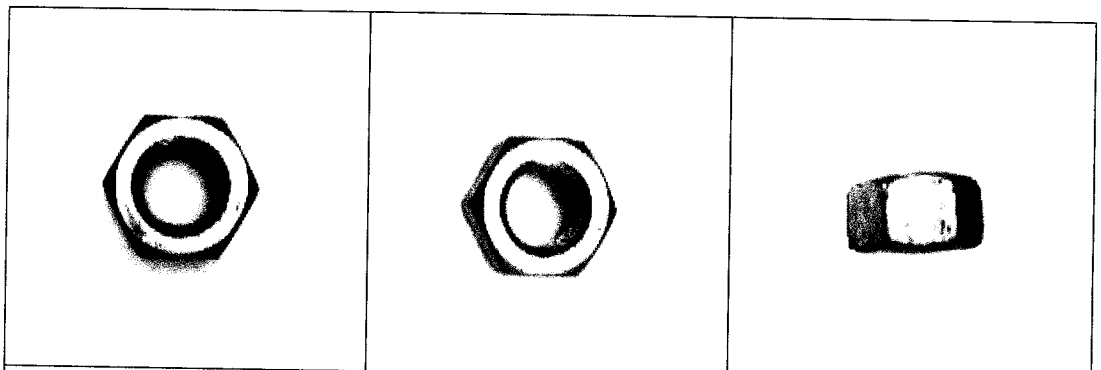
Polo4



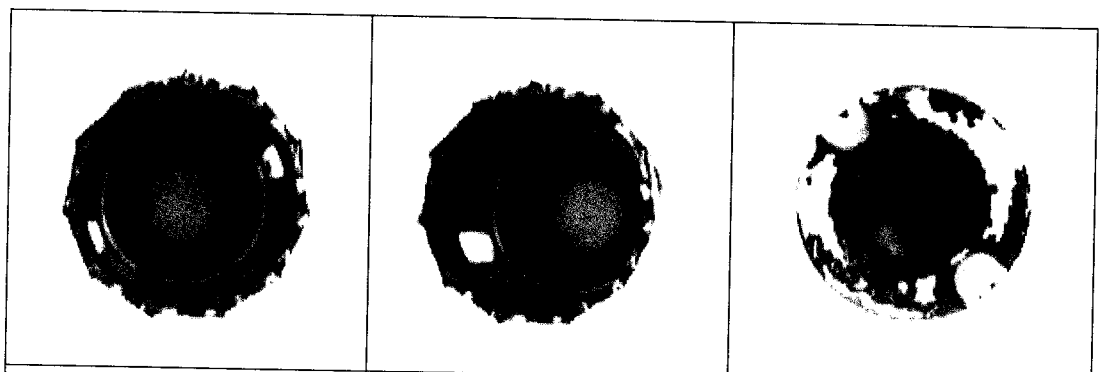
Polo7



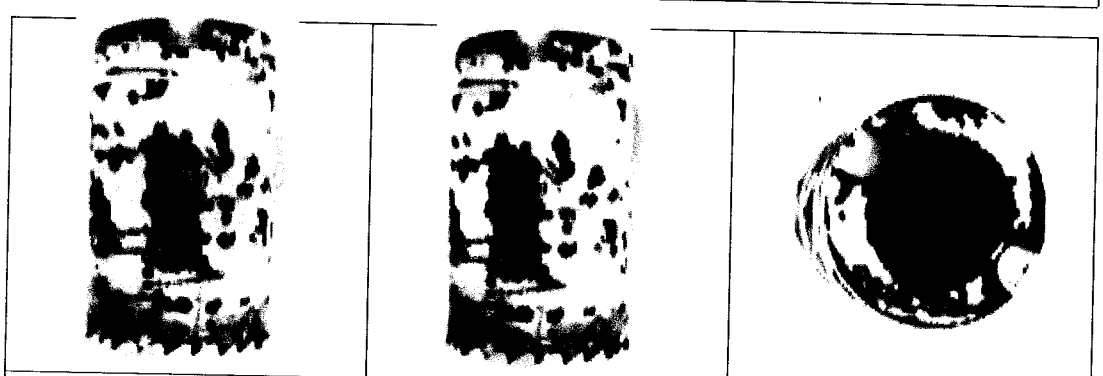
Cil14



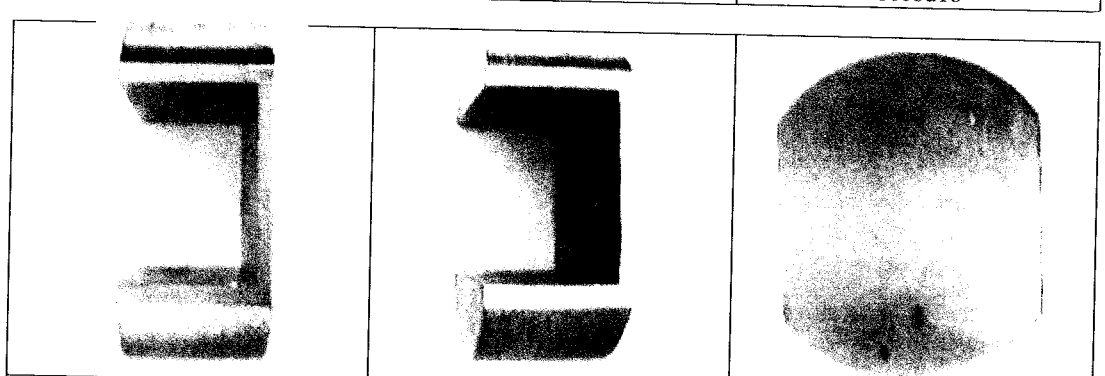
Porcac37 | Porcac40 | Porcap1



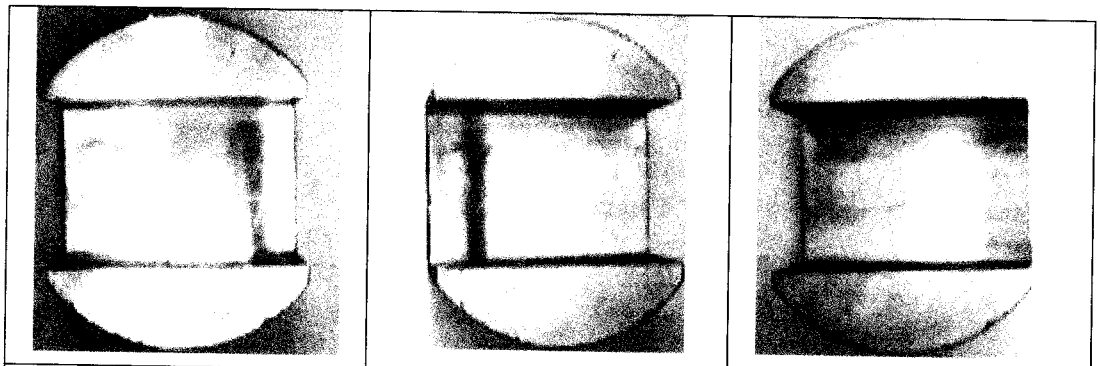
Seco1c36 | Seco1e32 | Seco2c39



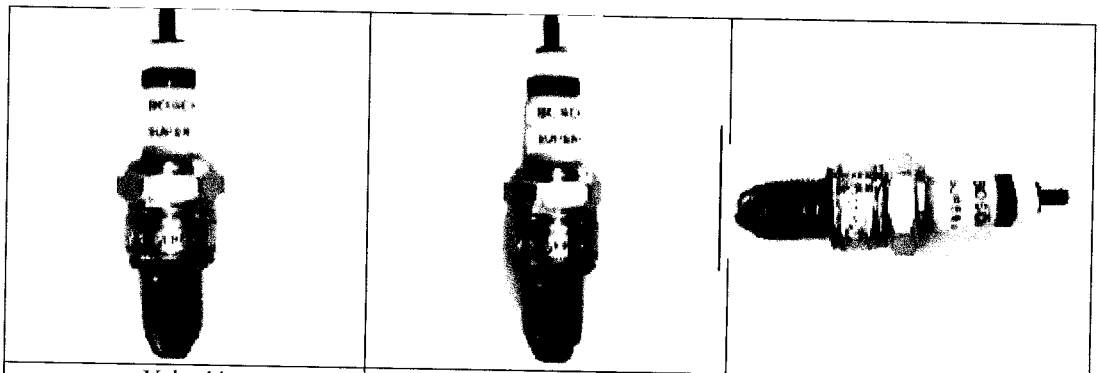
Secoc23 | Secoe27 | Secod18



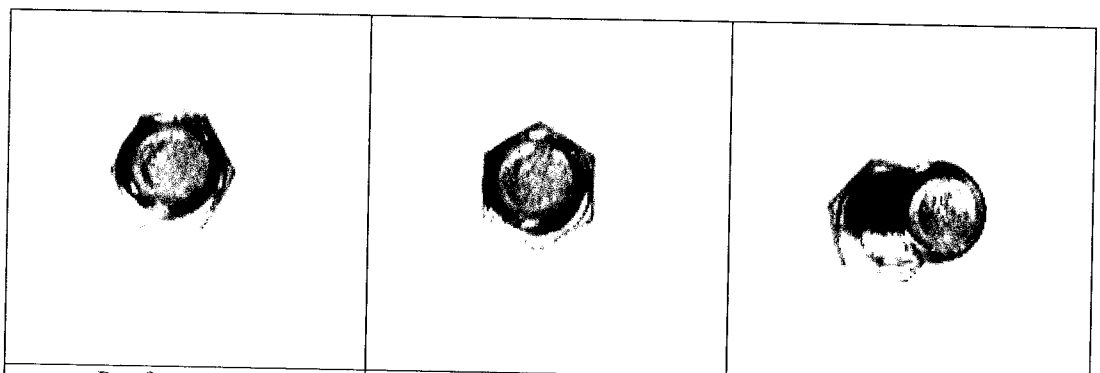
U1c40 | U1d33 | U3c2



Uc20	Ud25	Ue29



Velac44	Velad49	Vela4



ParafusoImage10	ParafusoImage13	ParafusoImage51

Parafuso122	Sc2	Isoladorc1

APÊNDICE B

TABELAS DE CARACTERÍSTICAS
E
RESULTADOS EXPERIMENTAIS

Peça	Média	DesvPadr	Curtose	Energia	Entropia	Área	EixoMaior	EixoMenor	Elongação
Cilc31	174,495	5,495	889,485	1126,6	10,0	26394	142,584094820546	96,678424	0,74068308
Cilc32	194,462	6,257	953,324	913,7	9,7	26387	142,591483541382	96,280774989978	0,727777798512395
Cilc33	173,896	5,475	1001,732	973,8	9,8	25870	142,664303037592	99,0946038236623	0,773213444405289
Cild34	110,415	4,759	360,331	840,8	9,4	27598	152,597996048426	108,012087566658	0,831095420704581
Cild35	136,141	4,668	801,738	844,1	9,4	26389	151,691477239845	106,473187561359	0,829178080375662
Cild36	86,568	5,301	93,829	1066,7	9,9	27626	152,670850636083	108,777529361481	0,834588664872845
Cilmad1c11	232,738	8,069	1127,763	45613,4	14,0	25657	140,044821132735	99,6443102274057	0,740800682064231
Cilmad1c13	242,711	8,588	789,938	63967,6	15,6	26536	142,518363552328	103,620978605474	0,778787180223085
Cilmad1d17	234,968	8,184	1118,635	45188,4	14,0	26303	140,951688483491	99,7495489538113	0,736357442875673
Cilmad1d18	241,420	8,520	925,278	64888,8	15,6	26534	142,505946277934	103,579494752635	0,778424620333128
Cilmad1d19	242,716	8,589	789,731	64374,6	15,6	26537	142,521259946147	103,624120898076	0,778744448332369
Cilmad1e14 *	182,379	5,776	1182,539	14607,2	11,9	15647	135,412723607894	112,693686094054	0,911511663881594
Cilmad1e15 *	151,528	4,876	3043,563	5731,3	10,6	19958	142,484065925577	107,642397512504	0,77190279974526
Cilmad1e16 *	126,977	4,637	5193,230	4251,5	10,2	14405	128,32694997181	109,586831507217	0,918560206335908
Cilmadc1	241,891	8,545	859,176	64242,5	15,6	26325	141,001083207813	100,664452169339	0,744761314225151
Cilmadc2	242,394	8,572	813,086	64757,3	15,6	26742	144,123608654254	105,816637507358	0,804399989751562
Cilmadc3	236,856	8,281	1205,788	41775,4	14,1	27805	153,388940507686	112,752027321974	0,851345672348708
Cilmadd10	232,769	8,070	1118,695	34554,5	13,3	26330	141,041433859052	100,834889029938	0,746428187293603
Cilmadd8	242,361	8,570	814,547	64382,4	15,6	26530	142,498715020622	103,423153354476	0,776796730195767
Cilmadd9	240,853	8,490	898,038	56751,8	15,0	26734	144,068544012019	105,671864462851	0,803211266311614
Cilmade4 *	154,322	4,933	7414,149	12329,8	11,6	24165	157,645550538562	113,804551619994	0,842645301164387
Cilmade5 *	159,205	5,046	5590,334	13673,4	11,5	15835	119,537830073461	113,432162958603	1,01198194172631
Conectorc1	239,552	8,422	1097,076	60849,7	15,3	26535	142,48997316892	103,516524928724	0,777534728040399
Conectorc2	239,212	8,404	1239,526	60946,6	15,4	26501	153,064314348515	111,6334849439	0,845432958103418
Conectorc3	238,722	8,379	1252,104	59900,6	15,3	28212	153,363593168432	113,446024784946	0,854514646414198
Conectorcp10	240,283	8,460	1198,670	63992,9	15,7	26714	152,344438515342	112,081406628929	0,849291721296996
Conectorcp11	240,283	8,460	1198,670	63992,9	15,7	28065	153,245071571154	114,102644369246	0,85774233110846
Conectorcp12	240,283	8,460	1198,670	63992,9	15,7	26568	153,094891923969	112,155142169338	0,847110580369544
Conectord4	240,282	8,460	1198,650	63928,1	15,7	28431	155,245320295679	119,998816507446	0,871730413210542
Conectord5	240,283	8,460	1198,670	63992,9	15,7	26797	154,947849086479	116,028378621019	0,852776623063149
Conectord6	240,475	8,470	1139,010	64347,9	15,6	26797	154,947849086479	116,028378621019	0,852776623063149
Conectordp13	240,283	8,460	1198,670	63992,9	15,7	28005	154,535247359387	116,419581378188	0,862614863533287

Conectordp14	240,901	8,493	1027,277	64357,5	15,6	25761	141,275988215884	103,553173149279	0,788375108155572
Conectordp15	240,901	8,493	1027,277	64357,5	15,6	25761	141,275988215884	103,553173149279	0,788375108155572
Peça	Média	DesvPadr	Curtose	Energia	Entropia	Área	EixoMaior	EixoMenor	Elongação
Conectore7	196,657	6,351	2083,566	14008,5	11,0	25761	141,275988215884	103,553173149279	0,788375108155572
Conectore8	194,323	6,251	2349,550	13618,0	11,0	17398	154,322718158443	107,566849504749	1,02311365889234
Conectore9	203,626	6,657	1666,299	15750,7	11,3	15233	129,011581515861	111,682453694874	1,03075726241031
Conectorep16	207,889	6,851	1689,033	24566,1	12,1	12249	135,07982475665	118,965380337646	0,963302939606537
Conectorep17	211,637	7,026	1469,356	15736,5	12,2	11924	137,587368659689	119,617553037174	0,957412562207008
Conectorep18	217,963	7,329	1190,327	26976,1	12,4	11580	138,682380367458	120,206358475897	0,973761464302111
Conectorep19	206,594	6,792	1734,779	24322,7	12,0	12111	128,889523261206	118,246910153988	0,971987862251818
Conectorep20	199,154	6,459	2237,686	23279,1	12,0	13976	162,532025854383	114,188850422212	0,932859926281003
ConectorTVc	240,283	8,460	1198,670	63992,9	15,7	26850	152,406663449574	112,570785320218	0,850809850666461
ConectorTVc21	241,213	8,509	961,016	64342,1	15,6	26129	140,137479616525	101,049532636171	0,754298141199584
ConectorTVc23	242,099	8,556	837,676	64374,6	15,6	25761	141,298669392987	103,469207645501	0,787541331148532
ConectorTVc8	242,505	8,577	802,976	64363,1	15,6	25654	140,148617318828	100,684222080474	0,751797528684204
conectorTVd	240,283	8,460	1198,670	63992,9	15,7	28400	154,743552985008	121,23405064403	0,878752347501473
ConectorTVd24	242,099	8,556	837,676	64374,6	15,6	25735	141,367876901797	102,173198697648	0,779611514817579
ConectorTVd25	241,892	8,545	859,191	64368,4	15,6	26088	140,176342271153	99,1281646115099	0,741866708401342
ConectorTVd26	242,395	8,572	813,100	64906,0	15,6	25833	142,793907424184	104,044396006282	0,806917053987186
ConectorTVd27	240,927	8,494	1021,300	64359,6	15,6	26706	144,235368763428	104,177949527872	0,795045923734684
ConectorTVd6	241,683	8,534	885,501	63835,3	15,6	25761	141,287267471578	103,512097077706	0,787961175516733
ConectorTVd7	243,015	8,604	777,576	64397,8	15,6	26736	144,101393910647	105,770382874541	0,804233910900972
ConectorTVe	168,636	5,307	5358,584	20769,7	12,0	14873	167,953972415007	107,673611822269	0,933253294935989
ConectorTVe28	183,003	5,799	3107,082	26753,5	12,7	14873	167,953972415007	107,673611822269	0,933253294935989
ConectorTVc22	241,683	8,534	885,501	63835,3	15,6	26436	142,067710237371	103,680172703952	0,782339551871864
Isolador16e	206,688	6,796	1517,306	12495,1	11,0	16233	123,434745209275	109,010272246868	1,01807386537439
Isolador17e	209,664	6,934	1506,014	18574,0	11,6	13926	128,241603302771	107,816651720301	1,10540469336086
Isolador18d	234,883	8,179	1114,495	35287,3	13,4	25648	140,219093494254	95,3706617747319	0,713402207400588
Isolador19d	237,367	8,308	995,438	45172,6	14,1	26329	142,000019934	100,745442006975	0,759923827473191
Isolador20d	239,287	8,408	899,465	45410,2	14,2	26639	144,169995075322	103,239887441106	0,787833316615901
Isolador20e	207,052	6,813	1377,323	15316,0	11,4	16627	173,805182623662	113,373385973693	0,773300800166689
Isolador25d	240,252	8,459	1101,422	61968,3	15,3	25660	140,276990299585	99,1759563844825	0,74030074019433
Isolador26d	236,792	8,278	1108,870	42808,9	14,0	26033	140,126610757737	98,1330846163595	0,73069098949547

Isoladorc	232,312	8,047	935,738	12713,4	11,9	25633	141,224647745814	81,2440913873229	0,625089056887728
Isoladorc15	236,506	8,263	1109,097	36931,0	13,6	25759	140,067059563246	100,604484001695	0,750706390482722
Isoladord	224,398	7,646	968,644	8370,2	11,1	26433	144,004812388211	94,3415029874325	0,72490873215221
Isoladord12	232,988	8,081	1076,036	20290,3	12,5	25858	141,336091734731	99,3396571411785	0,761809752713029
Parafusoc	241,292	8,513	926,433	64180,7	15,5	25753	142,181522550707	78,9133314680615	0,604608772983746
Parafusod	241,296	8,514	926,172	64420,8	15,5	25761	141,121284721076	79,2961025609848	0,614148779044794
Parafusoe	193,168	6,203	1575,922	19207,0	11,6	13504	138,282429717874	80,941937169455	0,701899332148424
Parale1c58	237,667	8,324	1157,860	61478,9	15,3	25868	142,723766375116	105,790426901421	0,817665988402258
Parale1c59	238,563	8,370	1013,147	61997,8	15,3	26436	142,112170048148	103,673122881504	0,782184348851162
Parale1d64	241,683	8,534	885,501	63835,3	15,6	26436	142,067710237371	103,680172703952	0,782339551871864
Parale1d67	240,738	8,484	1066,836	64363,2	15,6	25761	141,288459662023	103,438561430139	0,787085879815342
Parale1d68	241,154	8,506	971,748	64341,7	15,6	26223	140,571514241142	100,850844311353	0,7495910847364
Parale2c71	235,324	8,202	937,532	55374,5	14,8	25654	140,126883090045	100,522213966289	0,749825739459621
Parale2c72 *	237,654	8,323	822,057	55518,7	14,9	4809	193,720259663971	156,892866235315	1,24620089380319
Parale2d76	241,614	8,530	897,750	64387,7	15,6	26734	144,069055823107	105,67140299556	0,803206341371133
Parale2d77	241,275	8,513	948,987	64340,6	15,6	26533	142,53305610327	103,57058657805	0,778404988696888
Parlec43	241,742	8,537	822,259	63155,6	15,5	25868	142,701743490269	105,732937262472	0,816981497803132
Paralec44	241,699	8,535	822,684	63050,7	15,5	26740	144,120247619072	105,810312461606	0,804460291857722
Paralec47	241,437	8,521	848,235	62696,3	15,5	25761	141,298669392987	103,469207645501	0,787541331148532
Paraled51	241,420	8,520	925,278	64888,8	15,6	25761	141,306794317327	103,503269105742	0,787967949104593
Paraled52	241,420	8,520	925,278	64888,8	15,6	26436	142,112170048148	103,673122881504	0,782184348851162
Poloc3	239,152	8,401	1201,261	53924,8	14,9	27931	153,208499973903	112,780616939314	0,85217890902438
Poloc4	232,746	8,069	1212,517	56881,6	15,2	26540	151,858423556923	104,610896986638	0,820129941625695
Poloc5	239,419	8,415	1200,861	56766,8	15,1	27840	153,845573976611	113,21744201182	0,852082839403383
Poloc6	238,260	8,355	1203,130	50662,6	14,6	26536	152,945822895946	111,861269494489	0,847121569526306
Poloc7	240,283	8,460	1198,670	63992,9	15,7	27844	153,944158488002	113,275754091204	0,852022651472062
Poloc8	240,283	8,460	1198,670	63992,9	15,7	26433	153,226553233686	111,658868463724	0,845121072960296
Polod9	239,513	8,420	1201,270	59924,7	15,3	28039	154,313395683141	116,20891524937	0,86228123076757
Porcac37	232,915	8,078	1219,947	31125,6	13,2	27789	152,543508512753	112,354662174264	0,852223378450098
Porcac38	207,311	6,825	1260,839	61943,1	15,5	26787	151,120016553683	106,082622311353	0,829130018641133
Porcac39	241,540	8,526	906,160	63852,5	15,6	26740	144,100527205421	105,748863259421	0,803729529171398
Porcad40	241,335	8,516	905,301	60590,6	15,3	25889	142,670347655309	105,733457144029	0,816040735600977
Porcad41	234,989	8,185	1210,795	36346,1	13,6	27870	153,930189912502	115,532470228199	0,861445311969105

Porcad42	239,361	8,412	1200,079	56170,8	15,0	26586	153,23840754959	114,140326822673	0,854735118733086
Porcac40	240,422	8,468	1152,439	63806,2	15,6	26106	140,098707453791	99,681338906561	0,748926896079656
Porcac41	240,422	8,468	1152,439	63806,2	15,6	25661	140,123176891788	100,603005151286	0,752034297629649
Porcap1	240,422	8,468	1152,439	63806,2	15,6	26107	140,113777767883	100,16857366013	0,751179062538374
Porcap2	240,283	8,460	1198,670	63992,9	15,7	26368	153,226263887069	109,878010653101	0,837690971721776
Seco1c36	240,506	8,472	966,787	62840,8	15,4	25761	141,325811054649	103,495848742142	0,787975125498274
Seco1c38	241,094	8,503	888,093	62876,6	15,5	26730	144,066940860028	105,687932662128	0,803706572138158
Seco1d33	241,275	8,513	948,987	64340,6	15,6	25706	141,445630774081	100,850437798967	0,769443850924381
Seco1d34	241,275	8,513	948,987	64340,6	15,6	26478	142,651125945055	100,994832234041	0,760087122960398
Seco1d35	241,175	8,513	948,987	64340,6	15,6	25712	141,437557882833	101,142425986571	0,771473092031269
Secoc19	240,308	8,462	1155,781	63324,8	15,5	26126	140,103108645736	100,861776690292	0,754355377925796
Secoc21	241,780	8,539	869,040	63965,9	15,6	25761	141,301000926021	103,393326343898	0,786680866168594
Secoc22.	229,949	7,926	1225,063	20901,1	12,4	28350	151,96436548383	109,920996678848	0,843329791584649
Secoc23.	238,547	8,370	1203,560	52637,0	14,8	27114	151,708020434713	111,163800855952	0,847840568910232
Secod16	240,738	8,484	1066,836	64363,2	15,6	26383	142,168832837722	101,475787746887	0,766711299345517
Secod17	240,422	8,468	1152,439	63806,2	15,6	25618	140,245265125412	98,7641519462878	0,738067108171598
Secod18	240,420	8,468	1152,331	63782,8	15,6	26081	140,196468496724	99,0822306884478	0,740516720554025
U1c39	240,274	8,460	1198,490	63616,7	15,7	26367	152,530132470672	111,171255267508	0,846136819820355
U1c40	240,253	8,459	1198,084	63068,4	15,6	27694	152,737153559407	111,862295931886	0,848892857732596
U1c41	186,240	5,923	1259,665	1695,3	10,5	26348	151,71498045923	105,454834426039	0,824487085640314
U1c42	240,266	8,460	1198,336	63485,0	15,6	27722	152,854234170914	111,833012107156	0,848445987637503
U1d33	238,419	8,363	1202,863	61519,8	15,5	26412	153,155603198157	111,729164323043	0,84583748959477
U1d34	237,507	8,315	1204,601	52484,0	14,8	27739	153,192643992492	112,673748828993	0,851523523073634
U1d35	209,686	6,935	1280,039	2640,6	10,1	26675	150,921025262168	114,376746297186	0,866093489656106
U3d10	240,927	8,494	1021,300	64359,6	15,6	26742	144,14104468277	105,875248338446	0,805118621169063
U3d11	241,358	8,517	935,826	64369,0	15,6	25761	141,306794317327	103,503269105742	0,787967949104593
U3d12	241,614	8,530	897,750	64387,7	15,6	26734	144,069055823107	105,67140299556	0,803206341371133
U3d4	218,271	7,344	1259,617	20695,1	12,2	26494	153,028817846624	110,652300442828	0,841620532444723
Uc20 *	137,258	4,677	1093,078	1191,5	9,6	27520	152,729859984328	110,45862796724	0,84229070842769
Uc21 *	110,845	4,753	655,287	1189,4	9,6	26409	151,513042370558	109,093009194583	0,841396536193113
Uc22 *	172,763	5,438	1231,462	1308,8	9,8	25937	140,047989038066	97,0674116448622	0,719988245559809
Uc23 *	125,320	4,639	1197,805	1201,2	9,7	26480	151,579042321603	110,207690135258	0,845940415192374
Ud24	193,088	6,200	1289,041	941,2	9,6	27483	152,745291389936	107,664148193899	0,828987989769434

Ud25	224,871	7,670	1237,607	12561,5	11,6	26370	151,59899914362	105,58161182638	0,825435011594216
Ud26	206,909	6,806	1283,434	1677,6	9,9	26365	151,729918349144	106,035263777453	0,827055508729302
Velac42	237,822	8,332	1105,012	39134,7	13,9	25624	140,076427069021	96,3347380990898	0,722533483939202
Velac43	238,603	8,372	1103,756	42257,7	14,1	26019	140,21369935126	97,7282179142594	0,734750808564801
Velac44	239,747	8,432	1101,735	55068,4	14,9	25723	140,113902036029	100,132948201158	0,748722453082142
Velad48	232,023	8,032	1218,335	29498,6	13,0	27982	154,537172832454	115,8906583205	0,860814422250683
Velad49	237,316	8,305	1202,567	39526,1	14,0	26888	153,992379856099	115,04155114341	0,855350583760385
Velad50	230,943	7,977	1218,629	20433,1	12.4	27391	156,024702016959	117,428574538725	0,855197494296065
Velad51	231,695	8,015	1218,239	26516,8	12,8	28334	155,779394784054	118,665930459665	0,866639629049382
ConectorTV1	240,921	8,494	1029,558	68041,1	15,8	25868	142,996255451115	106,43764418954	0,825362912531131
Isolador2	228,224	7,839	1092,113	14580,5	12,0	26768	144,421062637996	106,632559739612	0,812386413770635
Isolador3	240,404	8,467	1068,666	63448,0	15,5	25868	142,996255451115	106,43764418954	0,825362912531131
Isolador4	240,694	8,482	1067,927	65370,1	15,6	26768	144,421062637996	106,632559739612	0,812386413770635
Parafuso25	238,069	8,345	1204,708	53038,7	14,8	27400	151,009956261442	113,65077559766	0,860800373978518
Parafuso26	240,312	8,462	1198,897	68134,5	15,9	28718	152,027470647483	115,055752674626	0,865698608891708
Parafuso27	240,301	8,461	1199,111	67773,9	15,8	27355	151,210529415794	112,935156892698	0,856775444450988
Parafuso28	240,035	8,447	1212,753	67851,9	15,8	28624	152,358436198761	114,288750461705	0,86137676276799
Parafuso30	234,542	8,162	1533,617	63396,3	15,5	26757	152,778353281148	112,409317388578	0,848587616069555
Parafuso31	233.409	8,103	1565,570	62818,5	15,4	28156	153,37262839505	114,091624632519	0,85724935307185
Parafuso32	232,904	8,077	1596,624	62060,1	15,3	26763	152,727818435395	112,816408296673	0,850534370096638
Parafuso33	233,989	8,1333	1531,356	63331,0	15,4	28054	153,544586175118	114,254032533108	0,857437415873593
Parafuso34	235,707	8,222	1472,593	64663,3	15,6	26749	152,573158044073	112,819149021636	0,851509404009641
Parafuso35	238,743	8,380	1296,759	66796,5	15,8	28531	152,38944015518	114,554636573715	0,86235685876806
Parafuso36	240,312	8,462	1198,897	68151,4	15,9	27126	151,792227945154	112,682724040127	0,853336206668365
Parafuso37	240,312	8,462	1198,897	68151,4	15,9	27435	151,771830793076	112,391127007136	0,851136412258199
Parafuso38	201,243	6,551	3368,841	39332,8	13,6	27784	154,184186223907	113,45011118038	0,85201321365496
Parafuso39	209,833	6,942	2050,959	41732,6	13,7	26768	144,421062637996	106,632559739612	0,812386413770635
Parafuso40	206,013	6,765	2635,606	42647,7	13,8	26768	144,421062637996	106,632559739612	0,812386413770635
Parafuso41	240,627	8,478	1103,742	68041,1	15,8	25868	142,996255451115	106,43764418954	0,825362912531131
Parafuso42	241,976	8,549	855,289	68041,1	15,8	25868	142,996255451115	106,43764418954	0,825362912531131
Polo11	240,447	8,469	1155,928	68041,1	15,8	26768	144,421062637996	106,632559739612	0,812386413770635
Polo12	240,447	8,469	1155,928	68041,1	15,8	25868	142,996255451115	106,43764418954	0,825362912531131
Polo13	240,435	8,468	1156,128	67511,9	15,8	25868	142,996255451115	106,43764418954	0,825362912531131

Polo14	240,447	8,469	1155,928	68041,1	15,8	26768	144,421062637996	106,632559739612	0,812386413770635
Polo15	240,446	8,469	1155,909	67990,4	15,8	25868	142,996255451115	106,43764418954	0,825362912531131
Polo16	240,447	8,469	1155,928	68041,1	15,8	26768	144,421062637996	106,632559739612	0,812386413770635
Polo17	240,426	8,468	1155,948	67164,3	15,8	25868	142,996255451115	106,43764418954	0,825362912531131
Polo18	240,425	8,468	1155,956	67455,6	15,8	26768	144,421062637996	106,632559739612	0,812386413770635
Porca10	240,411	8,467	1156,047	66761,8	15,7	25868	142,996255451115	106,43764418954	0,825362912531131
Porca5	238,918	8,389	1071,574	52403,2	14,7	26768	144,421062637996	106,632559739612	0,812386413770635
Porca6	238,854	8,386	1071,391	51848,4	14,7	25868	142,996255451115	106,43764418954	0,825362912531131
Porca7	239,680	8,429	1158,262	61300,2	15,3	26768	144,421062637996	106,632559739612	0,812386413770635
Porca8	240,425	8,468	1155,912	67097,2	15,8	25868	142,996255451115	106,43764418954	0,825362912531131
Porca9	239, 214	8,404	1159,938	57944,3	15,1	26768	144,421062637996	106,632559739612	0,812386413770635
Seco19	238,313	8,357	1265,347	66257,3	15,7	25868	142,996255451115	106,43764418954	0,825362912531131
Seco20	237,329	8,306	1311,696	65619,8	15,6	26768	144,421062637996	106,632559739612	0,812386413770635
Seco21	240,447	8,469	1155,928	68041,1	15,8	25868	142,996255451115	106,43764418954	0,825362912531131
Seco22	233,686	8,117	1519,840	62214,6	15,3	29506	151,159411778824	117,098479637094	0,875506265369705
Seco23	228,410	7,848	1687,708	56659,8	14,9	27830	151,079784590221	115,64173923335	0,864798617941436
Seco24	230,362	7,947	1520,099	56858,5	14,9	29327	150,924548660865	119,114566174957	0,884995771273532

TABELA DE CARACTERÍSTICAS DAS IMAGENS

SNNS pattern definition file V3.2
generated at Sat Apr 20 06:33:38 2002

No. of patterns: 168
No. of input units: 9
No. of output units: 3

0.55761483	0.99226946	0.82823440	0.96521582	0.98757764	0.95106261	0.98611059	0.98078307	0.96182486	1 0 0
0.55830551	0.99226946	0.75805318	0.96401311	0.98136646	0.99666740	0.96380946	0.97905582	0.95236867	0 0 1
0.68976632	0.98542242	0.92531158	0.86720177	0.95652174	0.96019996	0.97700660	0.98873446	0.96154992	1 0 0
0.52169909	0.99414688	0.74461282	0.97905237	0.98757764	0.99908490	0.96347520	0.97674173	0.95245115	0 0 1
0.66812478	0.98652678	0.81424587	0.86055586	0.95031056	0.96880463	0.97524127	0.98850119	0.94826770	1 0 0
0.49441694	0.99558255	0.82799585	0.99179138	0.99378882	0.94236233	0.99034411	0.97456184	0.98908314	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.98926464	0.97033955	0.97379561	0.96647412	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.96604567	0.97904248	0.96438205	0.98770876	0 0 1
0.73397030	0.98321369	0.95458548	0.85518008	0.95031056	0.96351890	0.97353709	0.99140289	0.94984782	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.96085555	0.97951581	0.96120897	0.99936586	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.99181873	0.96915778	0.97325841	0.96147338	0 0 1
0.56187406	0.99204859	0.75676142	0.97081138	0.98757764	0.99978147	0.96240092	0.97715826	0.95481989	0 0 1
0.60089789	0.99006074	0.76616663	0.86688087	0.95652174	0.96118335	0.97427314	0.99483215	0.94335677	0 1 0
0.65546218	0.98718940	0.82040265	0.86668402	0.95031056	0.96289063	0.97696447	0.98821883	0.96460733	1 0 0
0.45032808	0.99779128	0.71016427	0.99539276	1.00000000	0.98658763	0.97070503	0.97277589	0.96128365	0 0 1
0.61402095	0.98939812	0.75961902	0.86756852	0.95652174	0.95144504	0.97640511	0.99906292	0.95208027	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97291576	0.97726975	0.96963195	0.97989351	0 0 1
0.57177392	0.99149641	0.76923487	0.89078134	0.96273292	0.95147236	0.97873904	0.99633364	0.94379186	0 1 0
0.68216876	0.98586416	0.88330030	0.86485973	0.95031056	0.96635981	0.97334375	0.99355184	0.94948054	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97142701	0.97695524	0.97001989	0.98297753	0 0 1
0.53505238	0.99348426	0.82466114	0.95705574	0.98136646	0.92852655	0.99630461	0.98688499	0.96606209	0 1 0
0.61402095	0.98939812	0.75961902	0.86756852	0.95652174	0.95144504	0.97640511	0.99906292	0.95208027	0 1 0
0.63255439	0.98840420	0.89648688	0.86670020	0.95652174	0.95357572	0.97508776	0.99662607	0.94557153	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97704054	0.97347362	0.97086765	0.98012338	0 0 1
0.54702429	0.99282165	0.78992836	0.97305231	0.98757764	0.99207823	0.96682004	0.97803408	0.96041528	0 0 1
0.59272476	0.99050248	0.92566434	0.95653933	0.98136646	0.94710173	0.98707600	0.97838089	0.96271062	1 0 0
0.64245424	0.98785202	0.92235247	0.91523193	0.96894410	0.95271525	0.98215004	0.98444167	0.95243063	1 0 0
0.66605272	0.98663722	0.84829597	0.86952629	0.95652174	0.96544471	0.97411639	0.99342136	0.94602103	1 0 0
0.60285484	0.98995030	0.81966668	0.92546305	0.96894410	0.95644395	0.98129976	0.98649257	0.95183542	1 0 0
0.64210890	0.98796245	0.86660897	0.87612231	0.95652174	0.95163625	0.98038356	0.98754853	0.96957481	0 1 0

0.59836537	0.99017118	0.75236589	0.86219273	0.95031056	0.95391718	0.97480414	1.00000000	0.94389726	0 1 0
0.50316565	0.99503037	0.87592536	0.99179138	0.99378882	0.94199355	0.97076116	0.97363330	0.96567059	1 0 0
0.46437205	0.99712866	0.76083464	0.99579187	1.00000000	0.94091455	0.99251234	0.97576965	0.97520214	0 1 0
0.63508691	0.98829376	0.76837708	0.86143497	0.95031056	0.95487325	0.97714238	0.99464764	0.95874325	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97291576	0.97726975	0.96963195	0.97989351	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.96085555	0.97951581	0.96120897	0.99936586	0 0 1
0.60642339	0.98972943	0.87500412	0.90572217	0.96894410	0.94953289	0.98057016	0.98992320	0.94738130	0 1 0
0.57960170	0.99116510	0.78697686	0.95486470	0.98136646	0.99752786	0.96374413	0.97717820	0.95345154	0 0 1
0.63589271	0.98818332	0.93796014	0.92615744	0.97515528	0.95069384	0.98549948	0.97645767	0.98354245	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.99125874	0.96942361	0.97343251	0.96154055	0 0 1
0.57591804	0.99127554	0.78043432	0.86974472	0.95652174	0.94798951	0.97614625	0.98988699	0.93715996	0 1 0
0.61137332	0.98950856	0.84836196	0.92774307	0.97515528	0.95699028	0.98372388	0.97979896	0.96223318	1 0 0
0.64602279	0.98774158	0.90525763	0.90457340	0.96273292	0.95373962	0.98169030	0.98459675	0.97293120	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.99114948	0.96947089	0.97303954	0.96626076	0 0 1
0.60665362	0.98972943	0.88831759	0.91984456	0.96894410	0.94979240	0.98389259	0.98260566	0.97799580	0 1 0
0.55001727	0.99271121	0.77374460	0.91949400	0.96894410	0.94898656	0.98393700	0.98844281	0.95089630	0 1 0
0.68262922	0.98586416	0.86052325	0.87890257	0.95652174	0.96022727	0.98055806	0.98613728	0.95308334	1 0 0
0.55853574	0.99226946	0.76341562	0.90253742	0.96273292	0.94790756	0.98159422	0.99335754	0.94599909	0 1 0
0.64579256	0.98774158	0.86979903	0.87858976	0.95652174	0.95836976	0.97627495	0.98928695	0.94487935	0 1 0
0.53182917	0.99359470	0.76039052	0.94162007	0.97515528	0.94864510	0.98302931	0.98792436	0.97486431	0 1 0
0.59272476	0.99050248	0.81537774	0.95550111	0.98136646	0.99945367	0.96283282	0.97698673	0.95472236	0 0 1
0.40727524	1.00000000	0.69914247	0.99171182	0.99378882	0.93606589	0.99497122	0.96926783	0.99155774	0 1 0
0.67583746	0.98619547	0.85965277	0.85803179	0.95031056	0.95505081	0.97681903	0.99148063	0.95940019	0 1 0
0.60170370	0.98995030	0.80925142	0.91535868	0.96894410	0.95768684	0.98074666	0.98665699	0.97138127	1 0 0
0.71601243	0.98409718	0.96762741	0.86020934	0.95031056	0.96015898	0.97541971	0.99230482	0.94638594	1 0 0
0.51110855	0.99469906	0.75640612	0.98065958	0.99378882	0.99326650	0.96709620	0.97528998	0.96366241	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97142701	0.97695524	0.97001989	0.98297753	0 0 1
0.99735237	0.96974047	0.93192010	0.69450246	0.90062112	0.90353201	0.99617423	0.96709844	0.99765465	0 1 0
0.44549327	0.99801215	0.71397864	0.99841032	1.00000000	0.97478693	0.96969073	0.96761041	0.98652293	0 0 1
0.63589271	0.98818332	0.93796014	0.92615744	0.97515528	0.95069384	0.98549948	0.97645767	0.98354245	1 0 0
0.44480258	0.99812258	0.71448620	0.99856942	1.00000000	0.97805125	0.97015559	0.96873635	0.98311202	0 0 1
0.60607805	0.98972943	0.87596596	0.90808579	0.96894410	0.95018848	0.98036639	0.98961743	0.94745017	0 1 0
0.60112812	0.99006074	0.97359893	0.96903024	0.98757764	0.94452032	0.98976592	0.97351596	0.98711330	1 0 0
0.63508691	0.98829376	0.81574065	0.88731075	0.96273292	0.96297257	0.97644877	0.99102269	0.94643459	1 0 0
0.56291010	0.99204859	0.82785119	0.96240456	0.98136646	0.95156796	0.98567447	0.98100418	0.98229212	1 0 0
0.58777484	0.99072336	0.91234072	0.95332356	0.98136646	0.94075066	0.98993339	0.98094412	0.96232183	0 1 0
0.46368136	0.99712866	0.71553179	0.99021922	0.99378882	0.99356698	0.96653372	0.97633272	0.95411457	0 0 1
1.00000000	0.96951960	0.95147918	0.67187481	0.89440994	0.89666193	1.00000000	0.96854701	1.00000000	0 1 0

IESC-USP SERVIÇO DE BIBLIOTECA INFORMAÇÃO

0.56003223	0.99215903	0.76636458	0.97147341	0.98757764	1.00000000	0.96256722	0.97763081	0.95249700	0 0 1
0.50523771	0.99491993	0.81414689	0.99690424	1.00000000	0.94816707	0.98921081	0.97577677	0.97340704	1 0 0
0.48612870	0.99591386	0.79769666	1.00000000	1.00000000	0.94726563	0.96901535	0.97555750	0.96437155	1 0 0
0.58098308	0.99105467	0.82402160	0.96648729	0.98757764	0.99995903	0.96287618	0.97632368	0.95699635	0 0 1
0.44986762	0.99779128	0.70860858	0.99478467	0.99378882	0.97260162	0.97715859	0.96837459	0.97507295	0 0 1
0.64233913	0.98785202	0.89476369	0.87246565	0.95652174	0.95132212	0.97646681	0.99609134	0.95265767	0 1 0
0.58363071	0.99094423	0.87203232	0.93550945	0.97515528	0.94950557	0.98198773	0.98544077	0.97388809	0 1 0
0.52480718	0.99392601	0.74760746	0.97491570	0.98757764	0.99894832	0.96352036	0.97688359	0.95240441	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97142701	0.97695524	0.97001989	0.98297753	0 0 1
0.69966617	0.98498067	0.86307377	0.85726055	0.95031056	0.96910511	0.97308339	0.99301293	0.94823711	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.98418378	0.97123881	0.97306162	0.97242397	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97739565	0.97539474	0.97089304	0.97259060	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.99527426	0.95140307	0.97203260	0.95520076	0 0 1
0.56786002	0.99171728	0.75304603	0.90572217	0.96273292	0.94996995	0.97853302	0.99406704	0.96360802	0 1 0
0.51030275	0.99469906	0.75085842	0.98152116	0.99378882	0.99251530	0.96738969	0.97540397	0.96398599	0 0 1
0.53862093	0.99326339	0.88119390	0.98112745	0.99378882	0.94381010	0.98849249	0.97675817	0.96714901	0 1 0
0.60999194	0.98961900	0.85321683	0.90247809	0.96273292	0.95171820	0.97751083	0.99046125	0.96370880	0 1 0
0.56786002	0.99171728	0.75304603	0.90572217	0.96273292	0.94996995	0.97853302	0.99406704	0.96360802	0 1 0
0.46057327	0.99723909	0.71485419	0.99185475	0.99378882	0.99541084	0.96576764	0.97592158	0.95403547	0 0 1
0.60331530	0.98995030	0.76942521	0.87922077	0.95652174	0.95156796	0.98100196	0.98919129	0.97097710	0 1 0
0.59606308	0.99028161	0.81307085	0.95265479	0.98136646	0.99812882	0.96332002	0.97756855	0.95461815	0 0 1
0.57119834	0.99160685	0.87981585	0.95106916	0.98136646	0.94635052	0.98473910	0.98340633	0.95559574	0 1 0
0.46437205	0.99712866	0.76083464	0.99579187	1.00000000	0.94091455	0.99251234	0.97576965	0.97520214	0 1 0
0.50316565	0.99503037	0.87592536	0.99179138	0.99378882	0.94445203	0.99085733	0.97190694	0.98421123	1 0 0
0.42477265	0.99911651	0.70102808	0.99991775	1.00000000	0.94207550	0.99138873	0.97628996	0.99073237	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.96085555	0.97951581	0.96120897	0.99936586	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97806490	0.97461764	0.96978435	0.97893643	0 0 1
0.53424658	0.99348426	0.91139411	0.98740526	0.99378882	0.94426082	0.99000654	0.97378306	0.98914881	1 0 0
0.57672384	0.99127554	0.86942343	0.97408917	0.98757764	0.94950557	0.98831766	0.97641259	0.97269652	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.98298186	0.94928642	0.97721689	0.94724766	0 0 1
0.65200875	0.98741027	0.95255268	0.91896815	0.96894410	0.95177284	0.98531787	0.97852403	0.96290096	1 0 0
0.56267987	0.99204859	0.80128008	0.93835037	0.97515528	0.96610031	0.97798327	0.98237880	0.97127140	0 1 0
0.50316565	0.99503037	0.87592536	0.99179138	0.99378882	0.94284036	0.99091229	0.97230759	0.98285987	1 0 0
0.56371590	0.99193816	0.84912838	0.97884877	0.98757764	0.94878169	0.98859735	0.97515599	0.98616163	1 0 0
0.50915161	0.99480950	0.91455371	0.99179407	0.99378882	0.93930288	0.99204468	0.97316795	0.97697557	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.98657397	0.94054199	0.98377965	0.93204252	0 0 1
0.51053298	0.99469906	0.82111325	0.99619232	1.00000000	0.94977874	0.98820818	0.97593604	0.98676339	1 0 0
0.46690457	0.99690779	0.71839700	0.99056035	0.99378882	0.98964707	0.96948389	0.97470814	0.96627533	0 0 1

0.50915161	0.99480950	0.91455371	0.99179407	0.99378882	0.94207550	0.99149537	0.97198010	0.98564127	0 1 0
0.58742949	0.99072336	0.82082139	0.96328636	0.98136646	0.99810151	0.96361011	0.97711754	0.95698063	0 0 1
0.52918154	0.99370514	0.90622454	0.98829516	0.99378882	0.94311353	0.99077534	0.97319437	0.97506545	1 0 0
0.95637159	0.97183876	0.93024767	0.70120230	0.90062112	0.89828726	0.99916043	0.96827561	0.99994215	0 1 0
0.58224934	0.99094423	0.92601964	0.96446076	0.98757764	0.94359156	0.98923877	0.97694248	0.98749044	0 1 0
0.61022217	0.98950856	0.97859084	0.96364367	0.98757764	0.94676027	0.98877299	0.97388455	0.97363288	1 0 0
0.54368597	0.99293208	0.76473275	0.97559795	0.98757764	0.98885490	0.96584546	0.97950015	0.95930553	0 0 1
0.54230459	0.99304252	0.79779818	0.97569773	0.98757764	0.99617570	0.96504559	0.97625429	0.96043914	0 0 1
0.47553816	0.99646604	0.76556262	0.98805920	0.99378882	0.94259451	0.98988003	0.97926984	0.96747239	0 1 0
0.52791528	0.99381557	0.81549195	0.97258039	0.98757764	0.94267646	0.99049151	0.97934680	0.98868111	0 1 0
0.64475653	0.98774158	0.82085185	0.87960100	0.95652174	0.96523984	0.97581959	0.99011950	0.96050637	1 0 0
0.75284908	0.98221977	1.00000000	0.83980104	0.94409938	0.96428376	0.97405534	0.99169861	0.94852914	1 0 0
0.58386094	0.99094423	0.76251723	0.95864002	0.98136646	0.99086265	0.96497153	0.97876133	0.95748519	0 0 1
0.73120755	0.98332413	0.97152044	0.84978137	0.95031056	0.96050044	0.97560153	0.99042375	0.95694652	1 0 0
0.69517670	0.98520155	0.87917378	0.86325252	0.95031056	0.96704272	0.97379895	0.99357079	0.95075790	1 0 0
0.63589271	0.98818332	0.93796014	0.92615744	0.97515528	0.95069384	0.98549948	0.97645767	0.98354245	1 0 0
0.62357546	0.98884594	0.87018224	0.88228013	0.96273292	0.95162260	0.97822978	0.99299376	0.94501314	0 1 0
0.62748935	0.98862507	0.85914521	0.88221676	0.96273292	0.95782343	0.97413966	0.99331671	0.94625516	0 1 0
0.43570853	0.99856433	0.70375625	0.99650244	1.00000000	0.98168433	0.94058325	0.98599872	0.93012380	0 0 1
0.57822033	0.99116510	0.78531458	0.95543370	0.98136646	0.99792395	0.96362249	0.97670977	0.95358876	0 0 1
0.58904110	0.99061292	0.77260511	0.86672986	0.95652174	0.95290647	0.97622101	0.99908263	0.94414984	0 1 0
0.42477265	0.99911651	0.70102808	0.99991775	1.00000000	0.93613418	0.99457334	0.97469584	0.97923021	0 1 0
0.64061241	0.98796245	0.85698297	0.86555816	0.95652174	0.95722247	0.97462726	0.99354214	0.94318131	0 1 0
0.56786002	0.99171728	0.75304603	0.90572217	0.96273292	0.94996995	0.97853302	0.99406704	0.96360802	0 1 0
0.43352135	0.99867477	0.70380954	0.99983820	1.00000000	0.98399257	0.95143472	0.97298502	0.94815534	0 0 1
0.43536319	0.99856433	0.70385268	0.99666154	1.00000000	0.98957878	0.96931247	0.97577644	0.95746050	0 0 1
0.61977668	0.98906681	0.80529747	0.89469554	0.96273292	0.95883413	0.98095741	0.98374833	0.97388734	1 0 0
0.48612870	0.99591386	0.79769666	1.00000000	1.00000000	0.94813975	0.98999529	0.97468972	0.98234812	1 0 0
0.59399102	0.99039205	0.82893738	0.93675530	0.97515528	0.95511910	0.98439926	0.97813162	0.98220665	1 0 0
0.62691378	0.98873551	0.87090552	0.88127158	0.95652174	0.95263330	0.97883772	0.99135277	0.94505507	0 1 0
0.49441694	0.99558255	0.82799585	0.99179138	0.99378882	0.94311353	0.99139794	0.97225258	0.98483951	0 1 0
0.52964199	0.99370514	0.87927530	0.98751852	0.99378882	0.93154502	0.99602063	0.97642714	0.97734634	0 1 0
0.42477265	0.99911651	0.70102808	0.99991775	1.00000000	0.94468422	0.99112243	0.97426661	0.98466535	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.99063046	0.97001772	0.97268770	0.96502049	0 0 1
0.49441694	0.99558255	0.82799585	0.99179138	0.99378882	0.94311353	0.99139794	0.97225258	0.98483951	0 1 0
0.50316565	0.99503037	0.87592536	0.99179138	0.99378882	0.94445203	0.99085733	0.97190694	0.98421123	1 0 0
0.52653390	0.99392601	0.75538338	0.96664640	0.98757764	0.99474159	0.96535691	0.97611288	0.95398845	0 0 1
0.52964199	0.99370514	0.87927530	0.98751852	0.99378882	0.93154502	0.99602063	0.97642714	0.97734634	0 1 0

0.52883619	0.99370514	0.87670955	0.98804437	0.99378882	0.94221208	0.99100840	0.97375568	0.97444568	0 1 0
0.48198457	0.99613473	0.71946036	0.98185285	0.99378882	0.99236506	0.96723895	0.97391030	0.95686322	0 0 1
0.61402095	0.98939812	0.75961902	0.86756852	0.95652174	0.95144504	0.97640511	0.99906292	0.95208027	0 1 0
0.52918154	0.99370514	0.90622454	0.98829516	0.99378882	0.94311353	0.99077534	0.97319437	0.97506545	1 0 0
0.68481639	0.98575373	0.91922078	0.87865582	0.95652174	0.95826049	0.97768391	0.98942444	0.94704461	1 0 0
0.68999655	0.98542242	0.84502471	0.84953058	0.95031056	0.96816270	0.97403842	0.99220389	0.94624994	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97142701	0.97695524	0.97001989	0.98297753	0 0 1
0.52100840	0.99414688	0.76920188	0.94855318	0.98136646	0.94199355	0.98745700	0.98658931	0.95551051	0 1 0
0.59744446	0.99017118	0.92515424	0.95332626	0.98136646	0.94773000	0.98653847	0.97921605	0.98337689	1 0 0
0.64326004	0.98785202	0.81890533	0.88340059	0.95652174	0.96306818	0.97926337	0.98542507	0.95302800	1 0 0
0.49902153	0.99525124	0.74456460	0.98722324	0.99378882	0.98209408	0.96789873	0.97455818	0.97190783	0 0 1
0.64337516	0.98785202	0.86160690	0.87228228	0.95652174	0.95836976	0.97393594	0.99282447	0.95173935	0 1 0
0.46621388	0.99701822	0.72089423	0.99233071	0.99378882	0.99140898	0.96869686	0.97418437	0.96584231	0 0 1
0.70668815	0.98453893	0.92960813	0.85696526	0.95031056	0.95924388	0.97777621	0.98639045	0.96564164	1 0 0
0.52998734	0.99370514	0.75916221	0.94192884	0.97515528	0.94936899	0.98266055	0.98752474	0.97492125	0 1 0
0.49821572	0.99536168	0.74397582	0.98738099	0.99378882	0.99005682	0.96663440	0.97771047	0.96220039	0 0 1
0.74456084	0.98266151	0.99515528	0.84801371	0.95031056	0.96436571	0.97410891	0.99123162	0.95162622	1 0 0
0.48612870	0.99591386	0.79769666	1.00000000	1.00000000	0.94813975	0.98999529	0.97468972	0.98234812	1 0 0
0.52480718	0.99392601	0.75564731	0.96932687	0.98757764	0.99501475	0.96524599	0.97581288	0.95412076	0 0 1
0.48612870	0.99591386	0.79769666	1.00000000	1.00000000	0.94923241	0.98869239	0.97526615	0.97989471	1 0 0
0.54702429	0.99282165	0.76542558	0.97604425	0.98757764	0.99706348	0.96353065	0.97714281	0.95770495	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.98974268	0.97042253	0.97295004	0.96514577	0 0 1
0.45539312	0.99757040	0.71439230	0.99356982	0.99378882	0.98507157	0.97131015	0.97329347	0.96109778	0 0 1
0.49107862	0.99569299	0.71740725	0.97265994	0.98757764	0.98967439	0.96833604	0.97493027	0.95665235	0 0 1

SNNS pattern definition file V3.2
generated at Sat Apr 20 06:33:38 2002

No. of patterns: 118
No. of input units: 9
No. of output units: 3

0.60999194	0.98961900	0.85321683	0.90247809	0.96273292	0.95171820	0.97751083	0.99046125	0.96370880	0 1 0
0.56786002	0.99171728	0.75304603	0.90572217	0.96273292	0.94996995	0.97853302	0.99406704	0.96360802	0 1 0
0.46057327	0.99723909	0.71485419	0.99185475	0.99378882	0.99541084	0.96576764	0.97592158	0.95403547	0 0 1
0.60331530	0.98995030	0.76942521	0.87922077	0.95652174	0.95156796	0.98100196	0.98919129	0.97097710	0 1 0
0.59606308	0.99028161	0.81307085	0.95265479	0.98136646	0.99812882	0.96332002	0.97756855	0.95461815	0 0 1
0.57119834	0.99160685	0.87981585	0.95106916	0.98136646	0.94635052	0.98473910	0.98340633	0.95559574	0 1 0
0.46437205	0.99712866	0.76083464	0.99579187	1.00000000	0.94091455	0.99251234	0.97576965	0.97520214	0 1 0
0.50316565	0.99503037	0.87592536	0.99179138	0.99378882	0.94445203	0.99085733	0.97190694	0.98421123	1 0 0
0.42477265	0.99911651	0.70102808	0.99991775	1.00000000	0.94207550	0.99138873	0.97628996	0.99073237	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.96085555	0.97951581	0.96120897	0.99936586	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97806490	0.97461764	0.96978435	0.97893643	0 0 1
0.53424658	0.99348426	0.91139411	0.98740526	0.99378882	0.94426082	0.99000654	0.97378306	0.98914881	1 0 0
0.57672384	0.99127554	0.86942343	0.97408917	0.98757764	0.94950557	0.98831766	0.97641259	0.97269652	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.98298186	0.94928642	0.97721689	0.94724766	0 0 1
0.65200875	0.98741027	0.95255268	0.91896815	0.96894410	0.95177284	0.98531787	0.97852403	0.96290096	1 0 0
0.56267987	0.99204859	0.80128008	0.93835037	0.97515528	0.96610031	0.97798327	0.98237880	0.97127140	0 1 0
0.50316565	0.99503037	0.87592536	0.99179138	0.99378882	0.94284036	0.99091229	0.97230759	0.98285987	1 0 0
0.56371590	0.99193816	0.84912838	0.97884877	0.98757764	0.94878169	0.98859735	0.97515599	0.98616163	1 0 0
0.50915161	0.99480950	0.91455371	0.99179407	0.99378882	0.93930288	0.99204468	0.97316795	0.97697557	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.98657397	0.94054199	0.98377965	0.93204252	0 0 1
0.51053298	0.99469906	0.82111325	0.99619232	1.00000000	0.94977874	0.98820818	0.97593604	0.98676339	1 0 0
0.46690457	0.99690779	0.71839700	0.99056035	0.99378882	0.98964707	0.96948389	0.97470814	0.96627533	0 0 1
0.50915161	0.99480950	0.91455371	0.99179407	0.99378882	0.94207550	0.99149537	0.97198010	0.98564127	0 1 0
0.58742949	0.99072336	0.82082139	0.96328636	0.98136646	0.99810151	0.96361011	0.97711754	0.95698063	0 0 1
0.52918154	0.99370514	0.90622454	0.98829516	0.99378882	0.94311353	0.99077534	0.97319437	0.97506545	1 0 0
0.95637159	0.97183876	0.93024767	0.70120230	0.90062112	0.89828726	0.99916043	0.96827561	0.99994215	0 1 0
0.58224934	0.99094423	0.92601964	0.96446076	0.98757764	0.94359156	0.98923877	0.97694248	0.98749044	0 1 0
0.61022217	0.98950856	0.97859084	0.96364367	0.98757764	0.94676027	0.98877299	0.97388455	0.97363288	1 0 0
0.54368597	0.99293208	0.76473275	0.97559795	0.98757764	0.98885490	0.96584546	0.97950015	0.95930553	0 0 1

0.54230459	0.99304252	0.79779818	0.97569773	0.98757764	0.99617570	0.96504559	0.97625429	0.96043914	0 0 1
0.47553816	0.99646604	0.76556262	0.98805920	0.99378882	0.94259451	0.98988003	0.97926984	0.96747239	0 1 0
0.52791528	0.99381557	0.81549195	0.97258039	0.98757764	0.94267646	0.99049151	0.97934680	0.98868111	0 1 0
0.64475653	0.98774158	0.82085185	0.87960100	0.95652174	0.96523984	0.97581959	0.99011950	0.96050637	1 0 0
0.75284908	0.98221977	1.00000000	0.83980104	0.94409938	0.96428376	0.97405534	0.99169861	0.94852914	1 0 0
0.55761483	0.99226946	0.82823440	0.96521582	0.98757764	0.95106261	0.98611059	0.98078307	0.96182486	1 0 0
0.55830551	0.99226946	0.75805318	0.96401311	0.98136646	0.99666740	0.96380946	0.97905582	0.95236867	0 0 1
0.68976632	0.98542242	0.92531158	0.86720177	0.95652174	0.96019996	0.97700660	0.98873446	0.96154992	1 0 0
0.52169909	0.99414688	0.74461282	0.97905237	0.98757764	0.99908490	0.96347520	0.97674173	0.95245115	0 0 1
0.66812478	0.98652678	0.81424587	0.86055586	0.95031056	0.96880463	0.97524127	0.98850119	0.94826770	1 0 0
0.49441694	0.99558255	0.82799585	0.99179138	0.99378882	0.94236233	0.99034411	0.97456184	0.98908314	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.98926464	0.97033955	0.97379561	0.96647412	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.96604567	0.97904248	0.96438205	0.98770876	0 0 1
0.73397030	0.98321369	0.95458548	0.85518008	0.95031056	0.96351890	0.97353709	0.99140289	0.94984782	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.96085555	0.97951581	0.96120897	0.99936586	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.99181873	0.96915778	0.97325841	0.96147338	0 0 1
0.56187406	0.99204859	0.75676142	0.97081138	0.98757764	0.99978147	0.96240092	0.97715826	0.95481989	0 0 1
0.60089789	0.99006074	0.76616663	0.86688087	0.95652174	0.96118335	0.97427314	0.99483215	0.94335677	0 1 0
0.65546218	0.98718940	0.82040265	0.86668402	0.95031056	0.96289063	0.97696447	0.98821883	0.96460733	1 0 0
0.45032808	0.99779128	0.71016427	0.99539276	1.00000000	0.98658763	0.97070503	0.97277589	0.96128365	0 0 1
0.61402095	0.98939812	0.75961902	0.86756852	0.95652174	0.95144504	0.97640511	0.99906292	0.95208027	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97291576	0.97726975	0.96963195	0.97989351	0 0 1
0.57177392	0.99149641	0.76923487	0.89078134	0.96273292	0.95147236	0.97873904	0.99633364	0.94379186	0 1 0
0.68216876	0.98586416	0.88330030	0.86485973	0.95031056	0.96635981	0.97334375	0.99355184	0.94948054	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97142701	0.97695524	0.97001989	0.98297753	0 0 1
0.53505238	0.99348426	0.82466114	0.95705574	0.98136646	0.92852655	0.99630461	0.98688499	0.96606209	0 1 0
0.61402095	0.98939812	0.75961902	0.86756852	0.95652174	0.95144504	0.97640511	0.99906292	0.95208027	0 1 0
0.63255439	0.98840420	0.89648688	0.86670020	0.95652174	0.95357572	0.97508776	0.99662607	0.94557153	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97704054	0.97347362	0.97086765	0.98012338	0 0 1
0.54702429	0.99282165	0.78992836	0.97305231	0.98757764	0.99207823	0.96682004	0.97803408	0.96041528	0 0 1
0.59272476	0.99050248	0.92566434	0.95653933	0.98136646	0.94710173	0.98707600	0.97838089	0.96271062	1 0 0
0.64245424	0.98785202	0.92235247	0.91523193	0.96894410	0.95271525	0.98215004	0.98444167	0.95243063	1 0 0
0.66605272	0.98663722	0.84829597	0.86952629	0.95652174	0.96544471	0.97411639	0.99342136	0.94602103	1 0 0
0.60285484	0.98995030	0.81966668	0.92546305	0.96894410	0.95644395	0.98129976	0.98649257	0.95183542	1 0 0
0.64210890	0.98796245	0.86660897	0.87612231	0.95652174	0.95163625	0.98038356	0.98754853	0.96957481	0 1 0
0.59836537	0.99017118	0.75236589	0.86219273	0.95031056	0.95391718	0.97480414	1.00000000	0.94389726	0 1 0

0.50316565	0.99503037	0.87592536	0.99179138	0.99378882	0.94199355	0.97076116	0.97363330	0.96567059	1 0 0
0.46437205	0.99712866	0.76083464	0.99579187	1.00000000	0.94091455	0.99251234	0.97576965	0.97520214	0 1 0
0.63508691	0.98829376	0.76837708	0.86143497	0.95031056	0.95487325	0.97714238	0.99464764	0.95874325	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97291576	0.97726975	0.96963195	0.97989351	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.96085555	0.97951581	0.96120897	0.99936586	0 0 1
0.60642339	0.98972943	0.87500412	0.90572217	0.96894410	0.94953289	0.98057016	0.98992320	0.94738130	0 1 0
0.57960170	0.99116510	0.78697686	0.95486470	0.98136646	0.99752786	0.96374413	0.97717820	0.95345154	0 0 1
0.63589271	0.98818332	0.93796014	0.92615744	0.97515528	0.95069384	0.98549948	0.97645767	0.98354245	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.99125874	0.96942361	0.97343251	0.96154055	0 0 1
0.57591804	0.99127554	0.78043432	0.86974472	0.95652174	0.94798951	0.97614625	0.98988699	0.93715996	0 1 0
0.61137332	0.98950856	0.84836196	0.92774307	0.97515528	0.95699028	0.98372388	0.97979896	0.96223318	1 0 0
0.64602279	0.98774158	0.90525763	0.90457340	0.96273292	0.95373962	0.98169030	0.98459675	0.97293120	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.99114948	0.96947089	0.97303954	0.96626076	0 0 1
0.60665362	0.98972943	0.88831759	0.91984456	0.96894410	0.94979240	0.98389259	0.98260566	0.97799580	0 1 0
0.55001727	0.99271121	0.77374460	0.91949400	0.96894410	0.94898656	0.98393700	0.98844281	0.95089630	0 1 0
0.68262922	0.98586416	0.86052325	0.87890257	0.95652174	0.96022727	0.98055806	0.98613728	0.95308334	1 0 0
0.55853574	0.99226946	0.76341562	0.90253742	0.96273292	0.94790756	0.98159422	0.99335754	0.94599909	0 1 0
0.64579256	0.98774158	0.86979903	0.87858976	0.95652174	0.95836976	0.97627495	0.98928695	0.94487935	0 1 0
0.53182917	0.99359470	0.76039052	0.94162007	0.97515528	0.94864510	0.98302931	0.98792436	0.97486431	0 1 0
0.59272476	0.99050248	0.81537774	0.95550111	0.98136646	0.99945367	0.96283282	0.97698673	0.95472236	0 0 1
0.40727524	1.00000000	0.69914247	0.99171182	0.99378882	0.93606589	0.99497122	0.96926783	0.99155774	0 1 0
0.67583746	0.98619547	0.85965277	0.85803179	0.95031056	0.95505081	0.97681903	0.99148063	0.95940019	0 1 0
0.60170370	0.98995030	0.80925142	0.91535868	0.96894410	0.95768684	0.98074666	0.98665699	0.97138127	1 0 0
0.71601243	0.98409718	0.96762741	0.86020934	0.95031056	0.96015898	0.97541971	0.99230482	0.94638594	1 0 0
0.51110855	0.99469906	0.75640612	0.98065958	0.99378882	0.99326650	0.96709620	0.97528998	0.96366241	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97142701	0.97695524	0.97001989	0.98297753	0 0 1
0.99735237	0.96974047	0.93192010	0.69450246	0.90062112	0.90353201	0.99617423	0.96709844	0.99765465	0 1 0
0.44549327	0.99801215	0.71397864	0.99841032	1.00000000	0.97478693	0.96969073	0.96761041	0.98652293	0 0 1
0.63589271	0.98818332	0.93796014	0.92615744	0.97515528	0.95069384	0.98549948	0.97645767	0.98354245	1 0 0
0.44480258	0.99812258	0.71448620	0.99856942	1.00000000	0.97805125	0.97015559	0.96873635	0.98311202	0 0 1
0.60607805	0.98972943	0.87596596	0.90808579	0.96894410	0.95018848	0.98036639	0.98961743	0.94745017	0 1 0
0.60112812	0.99006074	0.97359893	0.96903024	0.98757764	0.94452032	0.98976592	0.97351596	0.98711330	1 0 0
0.63508691	0.98829376	0.81574065	0.88731075	0.96273292	0.96297257	0.97644877	0.99102269	0.94643459	1 0 0
0.56291010	0.99204859	0.82785119	0.96240456	0.98136646	0.95156796	0.98567447	0.98100418	0.98229212	1 0 0
0.58777484	0.99072336	0.91234072	0.95332356	0.98136646	0.94075066	0.98993339	0.98094412	0.96232183	0 1 0
0.46368136	0.99712866	0.71553179	0.99021922	0.99378882	0.99356698	0.96653372	0.97633272	0.95411457	0 0 1

EESC-USP SERVIÇO DE BIBLIOTECA INFORMAÇÃO

1.00000000	0.96951960	0.95147918	0.67187481	0.89440994	0.89666193	1.00000000	0.96854701	1.00000000	0 1 0
0.56003223	0.99215903	0.76636458	0.97147341	0.98757764	1.00000000	0.96256722	0.97763081	0.95249700	0 0 1
0.50523771	0.99491993	0.81414689	0.99690424	1.00000000	0.94816707	0.98921081	0.97577677	0.97340704	1 0 0
0.48612870	0.99591386	0.79769666	1.00000000	1.00000000	0.94726563	0.96901535	0.97555750	0.96437155	1 0 0
0.58098308	0.99105467	0.82402160	0.96648729	0.98757764	0.99995903	0.96287618	0.97632368	0.95699635	0 0 1
0.44986762	0.99779128	0.70860858	0.99478467	0.99378882	0.97260162	0.97715859	0.96837459	0.97507295	0 0 1
0.64233913	0.98785202	0.89476369	0.87246565	0.95652174	0.95132212	0.97646681	0.99609134	0.95265767	0 1 0
0.58363071	0.99094423	0.87203232	0.93550945	0.97515528	0.94950557	0.98198773	0.98544077	0.97388809	0 1 0
0.52480718	0.99392601	0.74760746	0.97491570	0.98757764	0.99894832	0.96352036	0.97688359	0.95240441	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97142701	0.97695524	0.97001989	0.98297753	0 0 1
0.69966617	0.98498067	0.86307377	0.85726055	0.95031056	0.96910511	0.97308339	0.99301293	0.94823711	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.98418378	0.97123881	0.97306162	0.97242397	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97739565	0.97539474	0.97089304	0.97259060	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.99527426	0.95140307	0.97203260	0.95520076	0 0 1
0.56786002	0.99171728	0.75304603	0.90572217	0.96273292	0.94996995	0.97853302	0.99406704	0.96360802	0 1 0
0.51030275	0.99469906	0.75085842	0.98152116	0.99378882	0.99251530	0.96738969	0.97540397	0.96398599	0 0 1
0.53862093	0.99326339	0.88119390	0.98112745	0.99378882	0.94381010	0.98849249	0.97675817	0.96714901	0 1 0

SNNS pattern definition file V3.2
generated at Sat Apr 20 06:33:38 2002

No. of patterns: 84
No. of input units: 9
No. of output units: 3

0.55761483	0.99226946	0.82823440	0.96521582	0.98757764	0.95106261	0.98611059	0.98078307	0.96182486	1 0 0
0.55830551	0.99226946	0.75805318	0.96401311	0.98136646	0.99666740	0.96380946	0.97905582	0.95236867	0 0 1
0.68976632	0.98542242	0.92531158	0.86720177	0.95652174	0.96019996	0.97700660	0.98873446	0.96154992	1 0 0
0.52169909	0.99414688	0.74461282	0.97905237	0.98757764	0.99908490	0.96347520	0.97674173	0.95245115	0 0 1
0.66812478	0.98652678	0.81424587	0.86055586	0.95031056	0.96880463	0.97524127	0.98850119	0.94826770	1 0 0
0.49441694	0.99558255	0.82799585	0.99179138	0.99378882	0.94236233	0.99034411	0.97456184	0.98908314	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.98926464	0.97033955	0.97379561	0.96647412	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.96604567	0.97904248	0.96438205	0.98770876	0 0 1
0.73397030	0.98321369	0.95458548	0.85518008	0.95031056	0.96351890	0.97353709	0.99140289	0.94984782	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.96085555	0.97951581	0.96120897	0.99936586	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.99181873	0.96915778	0.97325841	0.96147338	0 0 1
0.56187406	0.99204859	0.75676142	0.97081138	0.98757764	0.99978147	0.96240092	0.97715826	0.95481989	0 0 1
0.60089789	0.99006074	0.76616663	0.86688087	0.95652174	0.96118335	0.97427314	0.99483215	0.94335677	0 1 0
0.65546218	0.98718940	0.82040265	0.86668402	0.95031056	0.96289063	0.97696447	0.98821883	0.96460733	1 0 0
0.45032808	0.99779128	0.71016427	0.99539276	1.00000000	0.98658763	0.97070503	0.97277589	0.96128365	0 0 1
0.61402095	0.98939812	0.75961902	0.86756852	0.95652174	0.95144504	0.97640511	0.99906292	0.95208027	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97291576	0.97726975	0.96963195	0.97989351	0 0 1
0.57177392	0.99149641	0.76923487	0.89078134	0.96273292	0.95147236	0.97873904	0.99633364	0.94379186	0 1 0
0.68216876	0.98586416	0.88330030	0.86485973	0.95031056	0.96635981	0.97334375	0.99355184	0.94948054	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97142701	0.97695524	0.97001989	0.98297753	0 0 1
0.53505238	0.99348426	0.82466114	0.95705574	0.98136646	0.92852655	0.99630461	0.98688499	0.96606209	0 1 0
0.61402095	0.98939812	0.75961902	0.86756852	0.95652174	0.95144504	0.97640511	0.99906292	0.95208027	0 1 0
0.63255439	0.98840420	0.89648688	0.86670020	0.95652174	0.95357572	0.97508776	0.99662607	0.94557153	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97704054	0.97347362	0.97086765	0.98012338	0 0 1
0.54702429	0.99282165	0.78992836	0.97305231	0.98757764	0.99207823	0.96682004	0.97803408	0.96041528	0 0 1
0.59272476	0.99050248	0.92566434	0.95653933	0.98136646	0.94710173	0.98707600	0.97838089	0.96271062	1 0 0
0.64245424	0.98785202	0.92235247	0.91523193	0.96894410	0.95271525	0.98215004	0.98444167	0.95243063	1 0 0
0.66605272	0.98663722	0.84829597	0.86952629	0.95652174	0.96544471	0.97411639	0.99342136	0.94602103	1 0 0
0.60285484	0.98995030	0.81966668	0.92546305	0.96894410	0.95644395	0.98129976	0.98649257	0.95183542	1 0 0
0.64210890	0.98796245	0.86660897	0.87612231	0.95652174	0.95163625	0.98038356	0.98754853	0.96957481	0 1 0

0.59836537	0.99017118	0.75236589	0.86219273	0.95031056	0.95391718	0.97480414	1.00000000	0.94389726	0 1 0
0.50316565	0.99503037	0.87592536	0.99179138	0.99378882	0.94199355	0.97076116	0.97363330	0.96567059	1 0 0
0.46437205	0.99712866	0.76083464	0.99579187	1.00000000	0.94091455	0.99251234	0.97576965	0.97520214	0 1 0
0.63508691	0.98829376	0.76837708	0.86143497	0.95031056	0.95487325	0.97714238	0.99464764	0.95874325	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97291576	0.97726975	0.96963195	0.97989351	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.96085555	0.97951581	0.96120897	0.99936586	0 0 1
0.60642339	0.98972943	0.87500412	0.90572217	0.96894410	0.94953289	0.98057016	0.98992320	0.94738130	0 1 0
0.57960170	0.99116510	0.78697686	0.95486470	0.98136646	0.99752786	0.96374413	0.97717820	0.95345154	0 0 1
0.63589271	0.98818332	0.93796014	0.92615744	0.97515528	0.95069384	0.98549948	0.97645767	0.98354245	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.99125874	0.96942361	0.97343251	0.96154055	0 0 1
0.57591804	0.99127554	0.78043432	0.86974472	0.95652174	0.94798951	0.97614625	0.98988699	0.93715996	0 1 0
0.61137332	0.98950856	0.84836196	0.92774307	0.97515528	0.95699028	0.98372388	0.97979896	0.96223318	1 0 0
0.64602279	0.98774158	0.90525763	0.90457340	0.96273292	0.95373962	0.98169030	0.98459675	0.97293120	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.99114948	0.96947089	0.97303954	0.96626076	0 0 1
0.60665362	0.98972943	0.88831759	0.91984456	0.96894410	0.94979240	0.98389259	0.98260566	0.97799580	0 1 0
0.55001727	0.99271121	0.77374460	0.91949400	0.96894410	0.94898656	0.98393700	0.98844281	0.95089630	0 1 0
0.68262922	0.98586416	0.86052325	0.87890257	0.95652174	0.96022727	0.98055806	0.98613728	0.95308334	1 0 0
0.55853574	0.99226946	0.76341562	0.90253742	0.96273292	0.94790756	0.98159422	0.99335754	0.94599909	0 1 0
0.64579256	0.98774158	0.86979903	0.87858976	0.95652174	0.95836976	0.97627495	0.98928695	0.94487935	0 1 0
0.53182917	0.99359470	0.76039052	0.94162007	0.97515528	0.94864510	0.98302931	0.98792436	0.97486431	0 1 0
0.59272476	0.99050248	0.81537774	0.95550111	0.98136646	0.99945367	0.96283282	0.97698673	0.95472236	0 0 1
0.40727524	1.00000000	0.69914247	0.99171182	0.99378882	0.93606589	0.99497122	0.96926783	0.99155774	0 1 0
0.67583746	0.98619547	0.85965277	0.85803179	0.95031056	0.95505081	0.97681903	0.99148063	0.95940019	0 1 0
0.60170370	0.98995030	0.80925142	0.91535868	0.96894410	0.95768684	0.98074666	0.98665699	0.97138127	1 0 0
0.71601243	0.98409718	0.96762741	0.86020934	0.95031056	0.96015898	0.97541971	0.99230482	0.94638594	1 0 0
0.51110855	0.99469906	0.75640612	0.98065958	0.99378882	0.99326650	0.96709620	0.97528998	0.96366241	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97142701	0.97695524	0.97001989	0.98297753	0 0 1
0.99735237	0.96974047	0.93192010	0.69450246	0.90062112	0.90353201	0.99617423	0.96709844	0.99765465	0 1 0
0.44549327	0.99801215	0.71397864	0.99841032	1.00000000	0.97478693	0.96969073	0.96761041	0.98652293	0 0 1
0.63589271	0.98818332	0.93796014	0.92615744	0.97515528	0.95069384	0.98549948	0.97645767	0.98354245	1 0 0
0.44480258	0.99812258	0.71448620	0.99856942	1.00000000	0.97805125	0.97015559	0.96873635	0.98311202	0 0 1
0.60607805	0.98972943	0.87596596	0.90808579	0.96894410	0.95018848	0.98036639	0.98961743	0.94745017	0 1 0
0.60112812	0.99006074	0.97359893	0.96903024	0.98757764	0.94452032	0.98976592	0.97351596	0.98711330	1 0 0
0.63508691	0.98829376	0.81574065	0.88731075	0.96273292	0.96297257	0.97644877	0.99102269	0.94643459	1 0 0
0.56291010	0.99204859	0.82785119	0.96240456	0.98136646	0.95156796	0.98567447	0.98100418	0.98229212	1 0 0
0.58777484	0.99072336	0.91234072	0.95332356	0.98136646	0.94075066	0.98993339	0.98094412	0.96232183	0 1 0
0.46368136	0.99712866	0.71553179	0.99021922	0.99378882	0.99356698	0.96653372	0.97633272	0.95411457	0 0 1
1.00000000	0.96951960	0.95147918	0.67187481	0.89440994	0.89666193	1.00000000	0.96854701	1.00000000	0 1 0

0.56003223	0.99215903	0.76636458	0.97147341	0.98757764	1.00000000	0.96256722	0.97763081	0.95249700	0 0 1
0.50523771	0.99491993	0.81414689	0.99690424	1.00000000	0.94816707	0.98921081	0.97577677	0.97340704	1 0 0
0.48612870	0.99591386	0.79769666	1.00000000	1.00000000	0.94726563	0.96901535	0.97555750	0.96437155	1 0 0
0.58098308	0.99105467	0.82402160	0.96648729	0.98757764	0.99995903	0.96287618	0.97632368	0.95699635	0 0 1
0.44986762	0.99779128	0.70860858	0.99478467	0.99378882	0.97260162	0.97715859	0.96837459	0.97507295	0 0 1
0.64233913	0.98785202	0.89476369	0.87246565	0.95652174	0.95132212	0.97646681	0.99609134	0.95265767	0 1 0
0.58363071	0.99094423	0.87203232	0.93550945	0.97515528	0.94950557	0.98198773	0.98544077	0.97388809	0 1 0
0.52480718	0.99392601	0.74760746	0.97491570	0.98757764	0.99894832	0.96352036	0.97688359	0.95240441	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97142701	0.97695524	0.97001989	0.98297753	0 0 1
0.69966617	0.98498067	0.86307377	0.85726055	0.95031056	0.96910511	0.97308339	0.99301293	0.94823711	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.98418378	0.97123881	0.97306162	0.97242397	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97739565	0.97539474	0.97089304	0.97259060	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.99527426	0.95140307	0.97203260	0.95520076	0 0 1
0.56786002	0.99171728	0.75304603	0.90572217	0.96273292	0.94996995	0.97853302	0.99406704	0.96360802	0 1 0
0.51030275	0.99469906	0.75085842	0.98152116	0.99378882	0.99251530	0.96738969	0.97540397	0.96398599	0 0 1
0.53862093	0.99326339	0.88119390	0.98112745	0.99378882	0.94381010	0.98849249	0.97675817	0.96714901	0 1 0

SNNS pattern definition file V3.2
generated at Sat Apr 20 06:33:38 2002

No. of patterns: 42
No. of input units: 9
No. of output units: 3

0.60999194	0.98961900	0.85321683	0.90247809	0.96273292	0.95171820	0.97751083	0.99046125	0.96370880	0 1 0
0.56786002	0.99171728	0.75304603	0.90572217	0.96273292	0.94996995	0.97853302	0.99406704	0.96360802	0 1 0
0.46057327	0.99723909	0.71485419	0.99185475	0.99378882	0.99541084	0.96576764	0.97592158	0.95403547	0 0 1
0.60331530	0.98995030	0.76942521	0.87922077	0.95652174	0.95156796	0.98100196	0.98919129	0.97097710	0 1 0
0.59606308	0.99028161	0.81307085	0.95265479	0.98136646	0.99812882	0.96332002	0.97756855	0.95461815	0 0 1
0.57119834	0.99160685	0.87981585	0.95106916	0.98136646	0.94635052	0.98473910	0.98340633	0.95559574	0 1 0
0.46437205	0.99712866	0.76083464	0.99579187	1.00000000	0.94091455	0.99251234	0.97576965	0.97520214	0 1 0
0.50316565	0.99503037	0.87592536	0.99179138	0.99378882	0.94445203	0.99085733	0.97190694	0.98421123	1 0 0
0.42477265	0.99911651	0.70102808	0.99991775	1.00000000	0.94207550	0.99138873	0.97628996	0.99073237	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.96085555	0.97951581	0.96120897	0.99936586	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97806490	0.97461764	0.96978435	0.97893643	0 0 1
0.53424658	0.99348426	0.91139411	0.98740526	0.99378882	0.94426082	0.99000654	0.97378306	0.98914881	1 0 0
0.57672384	0.99127554	0.86942343	0.97408917	0.98757764	0.94950557	0.98831766	0.97641259	0.97269652	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.98298186	0.94928642	0.97721689	0.94724766	0 0 1
0.65200875	0.98741027	0.95255268	0.91896815	0.96894410	0.95177284	0.98531787	0.97852403	0.96290096	1 0 0
0.56267987	0.99204859	0.80128008	0.93835037	0.97515528	0.96610031	0.97798327	0.98237880	0.97127140	0 1 0
0.50316565	0.99503037	0.87592536	0.99179138	0.99378882	0.94284036	0.99091229	0.97230759	0.98285987	1 0 0
0.56371590	0.99193816	0.84912838	0.97884877	0.98757764	0.94878169	0.98859735	0.97515599	0.98616163	1 0 0
0.50915161	0.99480950	0.91455371	0.99179407	0.99378882	0.93930288	0.99204468	0.97316795	0.97697557	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.98657397	0.94054199	0.98377965	0.93204252	0 0 1
0.51053298	0.99469906	0.82111325	0.99619232	1.00000000	0.94977874	0.98820818	0.97593604	0.98676339	1 0 0
0.46690457	0.99690779	0.71839700	0.99056035	0.99378882	0.98964707	0.96948389	0.97470814	0.96627533	0 0 1
0.50915161	0.99480950	0.91455371	0.99179407	0.99378882	0.94207550	0.99149537	0.97198010	0.98564127	0 1 0
0.58742949	0.99072336	0.82082139	0.96328636	0.98136646	0.99810151	0.96361011	0.97711754	0.95698063	0 0 1
0.52918154	0.99370514	0.90622454	0.98829516	0.99378882	0.94311353	0.99077534	0.97319437	0.97506545	1 0 0
0.95637159	0.97183876	0.93024767	0.70120230	0.90062112	0.89828726	0.99916043	0.96827561	0.99994215	0 1 0
0.58224934	0.99094423	0.92601964	0.96446076	0.98757764	0.94359156	0.98923877	0.97694248	0.98749044	0 1 0
0.61022217	0.98950856	0.97859084	0.96364367	0.98757764	0.94676027	0.98877299	0.97388455	0.97363288	1 0 0
0.54368597	0.99293208	0.76473275	0.97559795	0.98757764	0.98885490	0.96584546	0.97950015	0.95930553	0 0 1
0.54230459	0.99304252	0.79779818	0.97569773	0.98757764	0.99617570	0.96504559	0.97625429	0.96043914	0 0 1
0.47553816	0.99646604	0.76556262	0.98805920	0.99378882	0.94259451	0.98988003	0.97926984	0.96747239	0 1 0

0.52791528	0.99381557	0.81549195	0.97258039	0.98757764	0.94267646	0.99049151	0.97934680	0.98868111	0 1 0
0.64475653	0.98774158	0.82085185	0.87960100	0.95652174	0.96523984	0.97581959	0.99011950	0.96050637	1 0 0
0.75284908	0.98221977	1.00000000	0.83980104	0.94409938	0.96428376	0.97405534	0.99169861	0.94852914	1 0 0
0.58386094	0.99094423	0.76251723	0.95864002	0.98136646	0.99086265	0.96497153	0.97876133	0.95748519	0 0 1
0.73120755	0.98332413	0.97152044	0.84978137	0.95031056	0.96050044	0.97560153	0.99042375	0.95694652	1 0 0
0.69517670	0.98520155	0.87917378	0.86325252	0.95031056	0.96704272	0.97379895	0.99357079	0.95075790	1 0 0
0.63589271	0.98818332	0.93796014	0.92615744	0.97515528	0.95069384	0.98549948	0.97645767	0.98354245	1 0 0
0.62357546	0.98884594	0.87018224	0.88228013	0.96273292	0.95162260	0.97822978	0.99299376	0.94501314	0 1 0
0.62748935	0.98862507	0.85914521	0.88221676	0.96273292	0.95782343	0.97413966	0.99331671	0.94625516	0 1 0
0.43570853	0.99856433	0.70375625	0.99650244	1.00000000	0.98168433	0.94058325	0.98599872	0.93012380	0 0 1
0.57822033	0.99116510	0.78531458	0.95543370	0.98136646	0.99792395	0.96362249	0.97670977	0.95358876	0 0 1

SNNS pattern definition file V3.2
generated at Satur 20 Apr 06:33:38 2002

No. of patterns: 42
No. of input units: 9
No. of output units: 3

0.58904110	0.99061292	0.77260511	0.86672986	0.95652174	0.95290647	0.97622101	0.99908263	0.94414984	0 1 0
0.70668815	0.98453893	0.92960813	0.85696526	0.95031056	0.95924388	0.97777621	0.98639045	0.96564164	1 0 0
0.45539312	0.99757040	0.71439230	0.99356982	0.99378882	0.98507157	0.97131015	0.97329347	0.96109778	0 0 1
0.52964199	0.99370514	0.87927530	0.98751852	0.99378882	0.93154502	0.99602063	0.97642714	0.97734634	0 1 0
0.62691378	0.98873551	0.87090552	0.88127158	0.95652174	0.95263330	0.97883772	0.99135277	0.94505507	0 1 0
0.68481639	0.98575373	0.91922078	0.87865582	0.95652174	0.95826049	0.97768391	0.98942444	0.94704461	1 0 0
0.52883619	0.99370514	0.87670955	0.98804437	0.99378882	0.94221208	0.99100840	0.97375568	0.97444568	0 1 0
0.59399102	0.99039205	0.82893738	0.93675530	0.97515528	0.95511910	0.98439926	0.97813162	0.98220665	1 0 0
0.52480718	0.99392601	0.75564731	0.96932687	0.98757764	0.99501475	0.96524599	0.97581288	0.95412076	0 0 1
0.49441694	0.99558255	0.82799585	0.99179138	0.99378882	0.94311353	0.99139794	0.97225258	0.98483951	0 1 0
0.48612870	0.99591386	0.79769666	1.00000000	1.00000000	0.94923241	0.98869239	0.97526615	0.97989471	1 0 0
0.61977668	0.98906681	0.80529747	0.89469554	0.96273292	0.95883413	0.98095741	0.98374833	0.97388734	1 0 0
0.42477265	0.99911651	0.70102808	0.99991775	1.00000000	0.94468422	0.99112243	0.97426661	0.98466535	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.99063046	0.97001772	0.97268770	0.96502049	0 0 1
0.49441694	0.99558255	0.82799585	0.99179138	0.99378882	0.94311353	0.99139794	0.97225258	0.98483951	0 1 0
0.48198457	0.99613473	0.71946036	0.98185285	0.99378882	0.99236506	0.96723895	0.97391030	0.95686322	0 0 1
0.50316565	0.99503037	0.87592536	0.99179138	0.99378882	0.94445203	0.99085733	0.97190694	0.98421123	1 0 0
0.52653390	0.99392601	0.75538338	0.96664640	0.98757764	0.99474159	0.96535691	0.97611288	0.95398845	0 0 1
0.52918154	0.99370514	0.90622454	0.98829516	0.99378882	0.94311353	0.99077534	0.97319437	0.97506545	1 0 0
0.52964199	0.99370514	0.87927530	0.98751852	0.99378882	0.93154502	0.99602063	0.97642714	0.97734634	0 1 0
0.68999655	0.98542242	0.84502471	0.84953058	0.95031056	0.96816270	0.97403842	0.99220389	0.94624994	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97142701	0.97695524	0.97001989	0.98297753	0 0 1
0.42477265	0.99911651	0.70102808	0.99991775	1.00000000	0.93613418	0.99457334	0.97469584	0.97923021	0 1 0
0.64326004	0.98785202	0.81890533	0.88340059	0.95652174	0.96306818	0.97926337	0.98542507	0.95302800	1 0 0
0.49902153	0.99525124	0.74456460	0.98722324	0.99378882	0.98209408	0.96789873	0.97455818	0.97190783	0 0 1
0.64337516	0.98785202	0.86160690	0.87228228	0.95652174	0.95836976	0.97393594	0.99282447	0.95173935	0 1 0
0.46621388	0.99701822	0.72089423	0.99233071	0.99378882	0.99140898	0.96869686	0.97418437	0.96584231	0 0 1
0.52998734	0.99370514	0.75916221	0.94192884	0.97515528	0.94936899	0.98266055	0.98752474	0.97492125	0 1 0
0.49821572	0.99536168	0.74397582	0.98738099	0.99378882	0.99005682	0.96663440	0.97771047	0.96220039	0 0 1
0.74456084	0.98266151	0.99515528	0.84801371	0.95031056	0.96436571	0.97410891	0.99123162	0.95162622	1 0 0

0.59744446	0.99017118	0.92515424	0.95332626	0.98136646	0.94773000	0.98653847	0.97921605	0.98337689	1 0 0
0.64061241	0.98796245	0.85698297	0.86555816	0.95652174	0.95722247	0.97462726	0.99354214	0.94318131	0 1 0
0.43536319	0.99856433	0.70385268	0.99666154	1.00000000	0.98957878	0.96931247	0.97577644	0.95746050	0 0 1
0.61402095	0.98939812	0.75961902	0.86756852	0.95652174	0.95144504	0.97640511	0.99906292	0.95208027	0 1 0
0.49107862	0.99569299	0.71740725	0.97265994	0.98757764	0.98967439	0.96833604	0.97493027	0.95665235	0 0 1
0.48612870	0.99591386	0.79769666	1.00000000	1.00000000	0.94813975	0.98999529	0.97468972	0.98234812	1 0 0
0.52100840	0.99414688	0.76920188	0.94855318	0.98136646	0.94199355	0.98745700	0.98658931	0.95551051	0 1 0
0.48612870	0.99591386	0.79769666	1.00000000	1.00000000	0.94813975	0.98999529	0.97468972	0.98234812	1 0 0
0.43352135	0.99867477	0.70380954	0.99983820	1.00000000	0.98399257	0.95143472	0.97298502	0.94815534	0 0 1
0.54702429	0.99282165	0.76542558	0.97604425	0.98757764	0.99706348	0.96353065	0.97714281	0.95770495	0 0 1
0.56786002	0.99171728	0.75304603	0.90572217	0.96273292	0.94996995	0.97853302	0.99406704	0.96360802	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.98974268	0.97042253	0.97295004	0.96514577	0 0 1

CONJUNTO DE TESTE 1

TESTES DE VERIFICAÇÃO DE APRENDIZADO

Padrão carregado de:
/root/tb2.pat

1 - Função de Aprendizagem: Std_Backpropagation

Aprendizagem para todos padrões:
ciclos : 60000
parametro: 0.40000
un. saídas : 3
padrões: 118 (total: 118)

ciclos/trein.:	SSE	MSE	SSE/o-units
60000:	85.20742	0.72210	28.40247
54000:	14.46873	0.12262	4.82291
48000:	8.87226	0.07519	2.95742
42000:	8.41004	0.07127	2.80335
36000:	8.98454	0.07614	2.99485
30000:	7.39743	0.06269	2.46581
24000:	4.48228	0.03799	1.49409
18000:	3.60810	0.03058	1.20270
12000:	3.66549	0.03106	1.22183
6000:	3.54942	0.03008	1.18314
1	2.93114	0.02484	0.97705

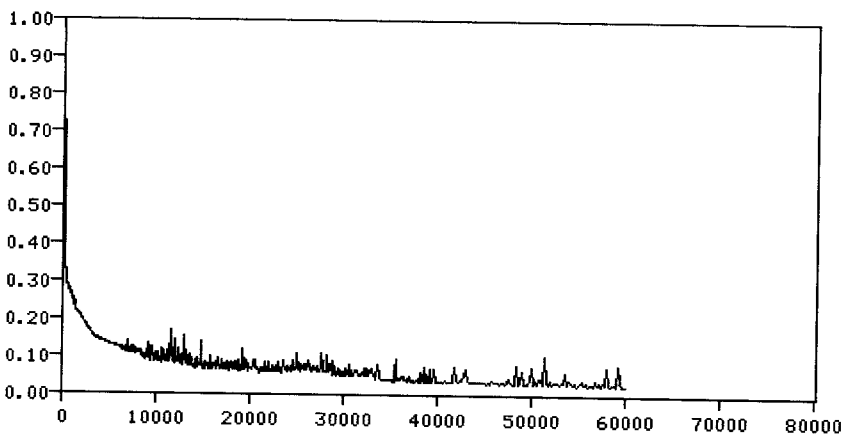


Figura - Gráfico do MSE (Erro Médio Quadrático)

A seguir realizou-se na sequencia, mais 60.000 ciclos, totalizando 120.000 ciclos

2 - Learning all patterns:
epochs : 60000
parameter: 0.40000
#o-units : 3

#patterns: 118 (total: 118)

epoch:	SSE	MSE	SSE/o-units
Train60000:	2.71302	0.02299	0.90434
Train54000:	5.58836	0.04736	1.86279
Train48000:	2.56849	0.02177	0.85616
Train42000:	2.38523	0.02021	0.79508
Train36000:	6.61468	0.05606	2.20489
Train30000:	1.53609	0.01302	0.51203
Train24000:	4.95784	0.04202	1.65261
Train18000:	1.11438	0.00944	0.37146
Train12000:	2.00844	0.01702	0.66948
Train 6000:	0.00000	0.00000	0.00000
Train 1:	0.00000	0.00000	0.00000

Verificou-se o aprendizado de 100%, após 114.000 ciclos.

3 – Realizando com 120.000 ciclos direto

Learning all patterns:
epochs : 120000
parameter: 0.40000
#o-units : 3
#patterns: 118 (total: 118)

epoch:	SSE	MSE	SSE/o-units
Train120000:	90.79363	0.76944	30.26454
Train108000:	12.67863	0.10745	4.22621
Train96000:	8.68298	0.07358	2.89433
Train84000:	7.32394	0.06207	2.44131
Train72000:	3.99167	0.03383	1.33056
Train60000:	2.50426	0.02122	0.83475
Train48000:	8.07071	0.06840	2.69024
Train36000:	4.40780	0.03735	1.46927
Train24000:	3.77065	0.03195	1.25688
Train12000:	3.39797	0.02880	1.13266
Train 1:	2.94547	0.02496	0.98182

4 - exp.120.000 ciclos direto reduzindo os passos e modificando a função de saída

Learning all patterns:
epochs : 120000
parameter: 0.40000
#o-units : 3
#patterns: 118 (total: 118)

epoch:	SSE	MSE	SSE/o-units
Train120000:	83.16278	0.70477	27.72093
Train108000:	11.77132	0.09976	3.92377
Train96000:	8.93833	0.07575	2.97944
Train84000:	6.77592	0.05742	2.25864

Train72000:	4.58270	0.03884	1.52757
Train60000:	2.17965	0.01847	0.72655
Train48000:	1.93352	0.01639	0.64451
Train36000:	2.20855	0.01872	0.73618
Train24000:	1.11554	0.00945	0.37185
Train12000:	0.00000	0.00000	0.00000
Train 1:	0.00000	0.00000	0.00000

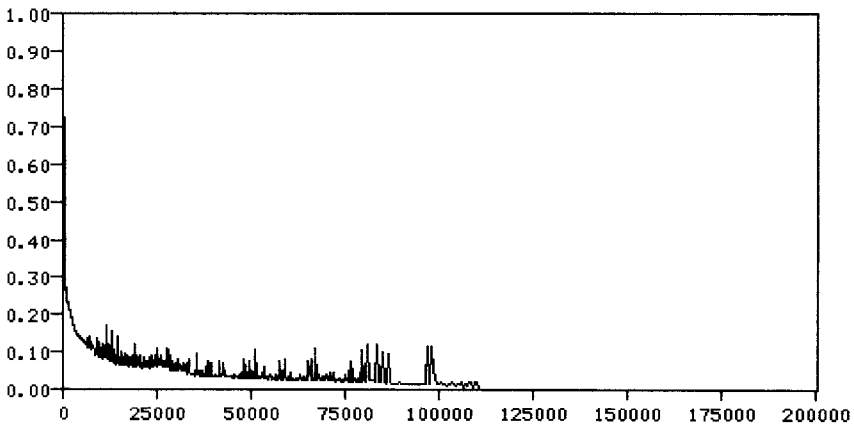


Figura - Gráfico do MSE

Learning func: Std_Backpropagation
Init. func: Randomize_Weights
Update func: Topological_Order
Learning func: Std_Backpropagation
Update func: Topological_Order
Init. func: Randomize_Weights
Remap. func: Binary

Verificou-se o aprendizado de 100% após 108.000 ciclos

5 - Learning all patterns:
epochs : 120000
parameter: 0.40000
#o-units : 3
#patterns: 118 (total: 118)

epoch:	SSE	MSE	SSE/o-units
Train120000:	6.13399	0.05198	2.04466
Train108000:	0.00000	0.00000	0.00000
Train96000:	0.00000	0.00000	0.00000
Train84000:	0.00000	0.00000	0.00000
Train72000:	0.00000	0.00000	0.00000
Train60669:	0.00000	0.00000	0.00000

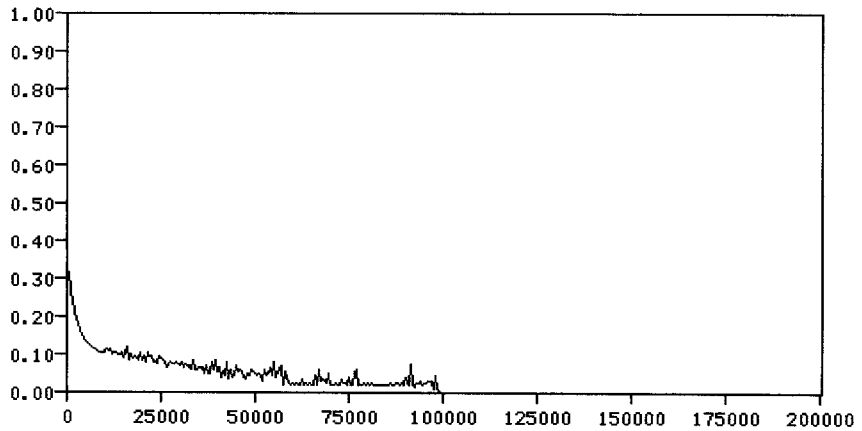


Figura - Gráfico do MSE

6 - Learning all patterns:
epochs : 60000
parameter: 0.40000
#o-units : 3
#patterns: 118 (total: 118)

epoch:	SSE	MSE	SSE/o-units
Train60000:	82.31165	0.69756	27.43722
Train54000:	13.28389	0.11258	4.42796
Train48000:	11.23285	0.09519	3.74428
Train42000:	7.40495	0.06275	2.46832
Train36000:	7.80947	0.06618	2.60316
Train30000:	5.29303	0.04486	1.76434
Train24000:	8.06282	0.06833	2.68761
Train18000:	5.04496	0.04275	1.68165
Train12000:	2.76498	0.02343	0.92166
Train 6000:	2.29734	0.01947	0.76578
Train 1:	3.27841	0.02778	1.09280

APÊNDICE C

DETECÇÃO DA IMAGEM

DETECÇÃO DA IMAGEM
(Complemento)

1 – CÂMERA

Devido às características dos sensores CCD, assim como, o seu baixo custo, baixa potência de consumo de energia, boa resolução, dimensão reduzida, durabilidade e de fácil operação, tem se tornado como uma tecnologia padrão para o uso em câmeras.

Os dois tipos de fotosensores são: fotodiodos e fotogate. Existem também dois tipos principais de mecanismos de transferência que são, a transferência chaveada digitalizada de tensão ou pacotes de carga e a transferência de pacotes analógicos de fotocarga. Estes fotosensores combinam para formar os quatro tipos primários de arquitetura de leitores de saída: CID, CCD, CCPD, e SSPD, conforme pode ser visto na fig. 1 abaixo.

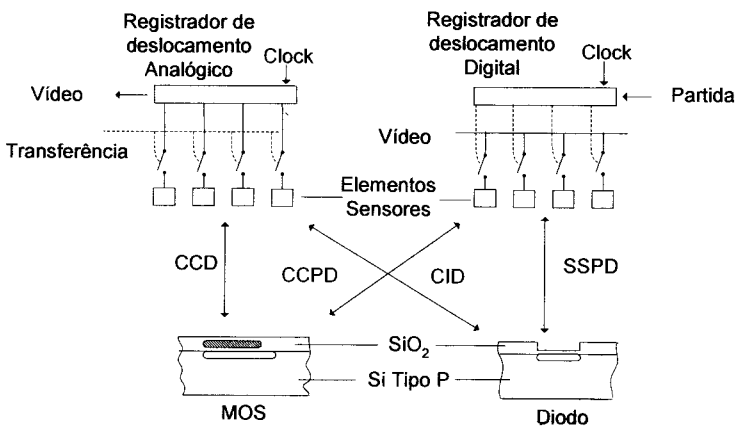


Fig. 1 - Os tipos primários de sensores de imagem, definidos pela combinação de fotosensores.

No CCD, a carga do sensor fotogate é transferida de forma serial da célula CCD para a célula CCD até alcançar o estágio final de saída. A arquitetura CCPD (charge-coupled priming device) utiliza uma especial estrutura principal de carga de multi-passagem para transferir a carga de um fotodiodo em um sistema CCD. Finalizando, temos a arquitetura SSPD (serially-switched fotodiodo device), um sistema multiplex digital ou chaves de leitura de saída para varrer a tensão ou cargas presentes na matriz de fotodiodos.

Estas configurações de sensores e processos de leitura de saída definem as principais arquiteturas de sensores em uso, atualmente, nas câmeras de vídeo de estado-sólido.

Na arquitetura CCD encontramos as quatro principais funções de sensor de imagem que são: fóton conversão, acumulação/coleção/armazenamento de carga, transferência/leitura

IFSC-USP SERVIÇO DE BIBLIOTECA INFORMAÇÃO

para exibição da carga, e medida do sinal. O CCD consiste em um mecanismo baseado em semicondutor no qual um finito pacote de cargas isolado é transportado de uma posição do semicondutor para uma posição adjacente numa seqüência de ciclos para uma matriz de portas.

Desde a sua invenção no início do ano de 1970 por Boyle e Smith do Laboratório Bell, os CCDs têm sido encontrados em várias aplicações na área de varredura e sensores de imagem. Os CCDs são basicamente registradores de deslocamento. A capacidade de não só transferir cargas, mas também de armazená-las, possibilitou o uso do led para o desenvolvimento dos primeiros sensores de imagem CCD, utilizando os fotodiodos como elementos sensor e CCDs como elementos para armazenar e transferir. Em pesquisas realizadas verificou-se que as células CCDs de silício podem ser transdutores de luz, assim também que as portas MOS existentes na estrutura CCDs podem ser usadas como fotosensor. O primeiro chip CCD comercial para uso em imagem foi introduzido em 1973, fabricado pela Faichild Weston.

Um CCD é, essencialmente, uma seqüência de portas capacitivas MOS, em que cada uma armazena uma carga no poço de potencial embaixo do eletrodo MOS. O poço de potencial é um volume localizado no silício que é atrativo para os elétrons (isto é, região mais positiva). O chip CCD para imagem é fabricado com vários milhares de poços de potencial colocados na área de silício, cada qual servindo como um fotolocal para detectar e acumular a fotocarga (fig.2).

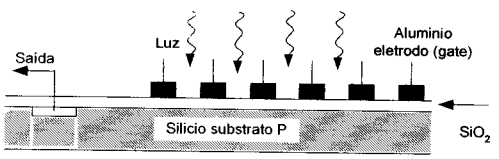


Fig. 2 - Vista de uma estrutura primária de um sensor CCD.

Nas câmeras de varredura de linha e câmeras de área, temos, respectivamente, os sensores de varredura de linha e o em forma de matriz, os quais são descritos abaixo:

. **Sensores de varredura de linha** – Os sensores de varredura de linha diferem dos de arranjo matricial ou de área principalmente no método de aquisição da imagem. O arranjo bidimensional ou sensor matricial adquire a imagem de um objeto em uma simples exposição, fazendo uma simples aquisição bidimensional. O sensor de varredura de linha ou arranjo unidimensional pode adquirir a mesma imagem, mas é preciso fazer uma série de varreduras unidimensional linha a linha da imagem. A construção básica de um sensor de varredura de linha é uma única linha de fotolocais. Colidindo um fóton da imagem muda-se a carga deste

local proporcionalmente à intensidade de iluminação da localização. As cargas serão lidas na saída como uma série de pulsos de amplitude modulada. Os sensores de varredura de linha podem ser utilizados em CCD, SSPD ou CCPD. Nas câmeras de varredura de linha a imagem bidimensional é obtida movimentando-se a câmera ou a imagem. Os sensores das câmeras de varredura de linha são encontrados facilmente com resoluções na faixa de 256 a 4.096 pixels, podendo ser fabricados com resoluções de 10.000 pixels ou acima, sob encomenda. Com uma taxa de varredura de 20.000 pixels por segundo, uma câmera de varredura que apresenta 2.048 pixels por linha obtém uma saída de aproximadamente 9.000 linhas por segundo. A Fig. 3 mostra um diagrama em bloco de uma típica câmera de varredura de linha.

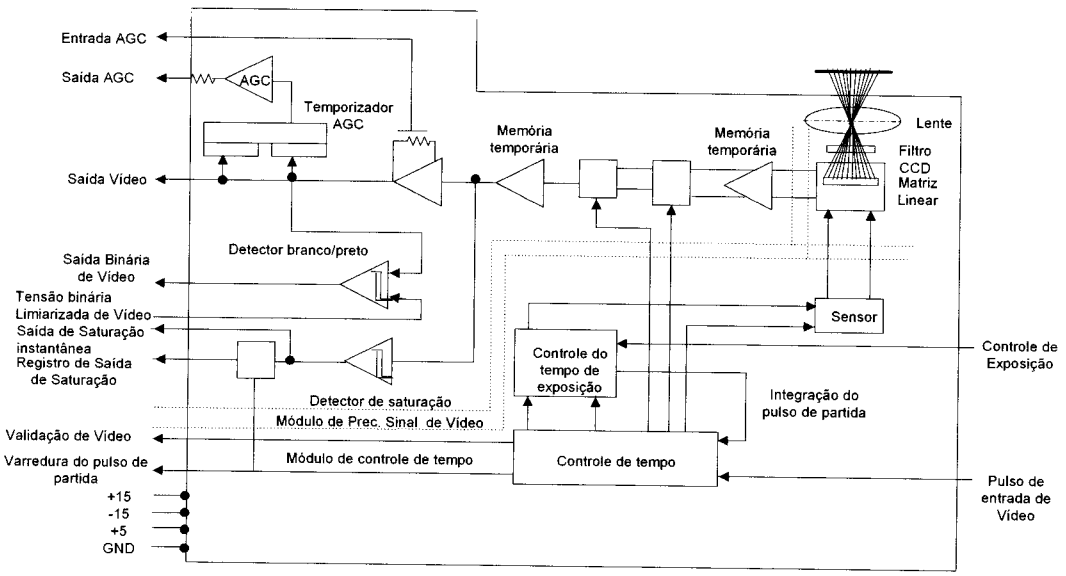


Fig. 3 - Típica câmera de varredura de linha (Fairchild).

As principais vantagens de uma câmera de varredura de linha são:

- alta densidade de pixel (acima de 10.000) útil para medições de precisão.
- Alta taxa de transferência de pixel, acima de 30 MHz (MHz e milhões de pixel/segundo são comumente permutável).
- Baixo custo.
- Simplicidade e confiabilidade.
- Resolução flexível, horizontalmente e verticalmente.
- Requer simples sistema de iluminação.
- Não tem distorção na perspectiva e poucos problemas de DOV.

- Bom controle de exposição e orientação.
- Saída de fluxo de dados contínuo em vez de uma imagem discreta.
- Apresenta interfaceamento flexível.

Na figura 4, encontramos os componentes de um típico sensor CCD para varredura de linha de 2.592 x 1, composto de 2.592 fotolocais de imagem, separados por um canal difusor, envolvido por uma superfície de SiO_2, e na fig. 5 apresenta o seu diagrama em bloco.

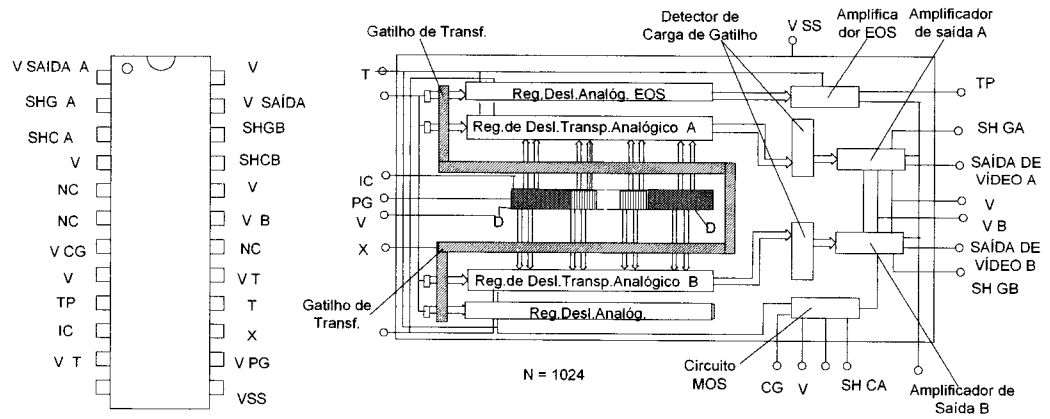


Fig. 4 - Componentes de um sensor CCD de varredura de linha.

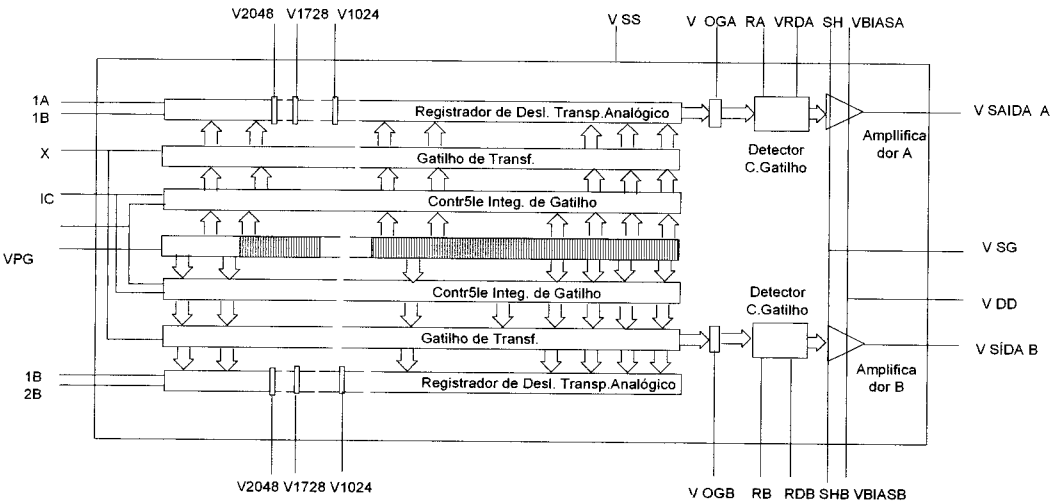


Fig. 5 - Digrama em bloco do sensor CCD de varredura de linha.

. **Sensores em forma de matriz (de área) -** Os sensores de arranjo de matriz são similares aos de varredura de linha, com exceção que os fotolocais são dispostos em formato de uma matriz e possui um porta adicional para transporte dos processos dos fotolocais.

A resolução dos sensores de arranjo de matriz se encontra entre 32 x 32 elementos para baixa resolução, 256 x 256 pixels para um sensor de média resolução e 480 x 380 para alta resolução, podendo ser utilizado sensores de 2.048 x 2.048 ou superior para altíssimas resoluções.

Os sensores de matriz que usam a arquitetura CCD para leitura e exposição podem ser classificados em quatro tipos: sensor FF (full-frame), sensor FT (frame-transfer), sensor ILT (interline-transfer) e sensor híbrido FIT (frame-interline-transfer).

Sensor FF: o sensor CCD de matriz FF (full-frame) tem uma arquitetura simples matriz completa bidimensional, requerendo apenas registrador H CCD (horizontal CCD) para carga, armazenar e transportar. Uma arquitetura básica e sua forma de operação podem ser vistas na fig. 6.

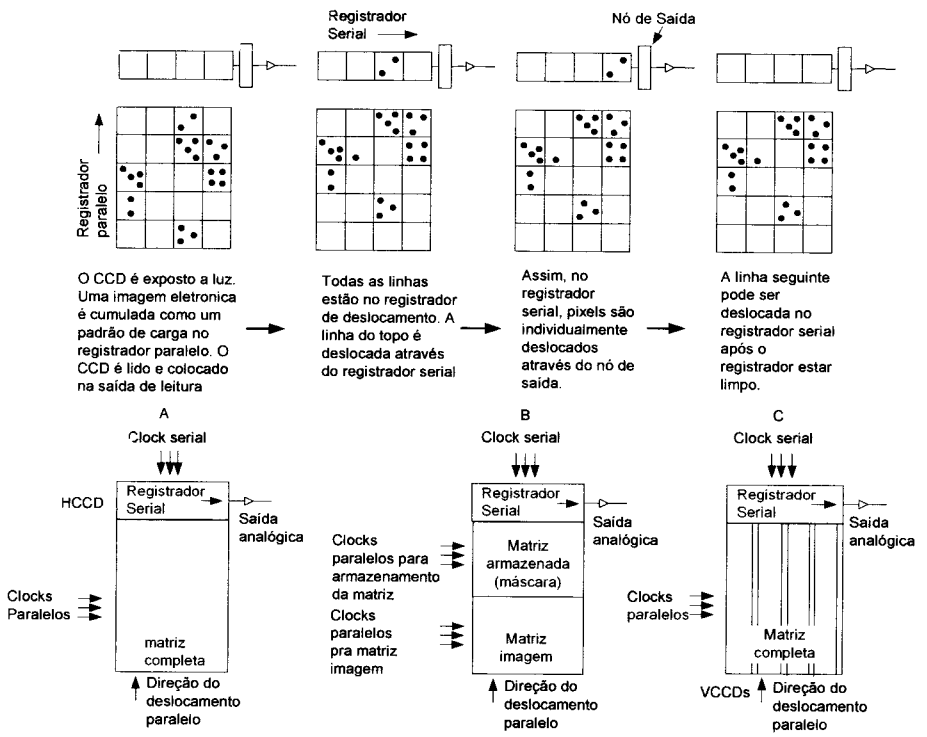


Fig. 6 – Organização e operação de um sensor FF (full-frame)

Uma câmera de matriz para vídeo executando o padrão NTSC, reseta a 1/60 segundo um campo, e a 1/30 segundo um frame, transferindo a imagem para o amplificador de vídeo a uma velocidade constante.

Sensor FT (frame-transfer): neste dispositivo, uma imagem inteira ou frame da informação é movida para um campo de armazenagem, sendo mais veloz que um dispositivo FF(full-frame). O dispositivo FT consiste de um imageador de área, que armazena a carga da área no mesmo circuito integrado (fig. 7).

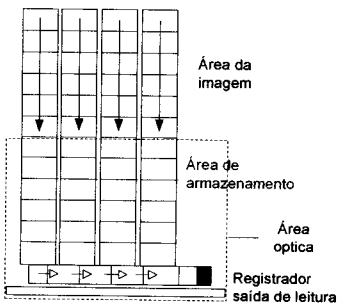


Fig. 7 - Arquitetura de um dispositivo FT (frame-transfer).

O sensor FT (frame-transfer) é composto por dois arranjos CCD, adaptados um próximo ao outro. Eles são essencialmente idênticos, exceto para os controles de linha e pelo fato de a região de armazenamento estar protegida da luz por uma camada de filme de alumínio.

O sensor FT (frame-transfer), contraria o seu nome, pois seria mais apropriado o termo "dispositivo de transferência de campo", quando usado em entrelaçamento nas câmeras de vídeo.

Empresas européias, como a EEV, Philips, Thonson-CSF e outras como Texas Instruments, Sierra Scientific, Sanyo e Cohu, preferem o FT CCD por sua alta resposta e baixo ruído, ale m de apresentar alta resolução e excelente sensibilidade.

Sensor ILT (interline-transfer): foram desenvolvidos para controlar a exposição e reduzir o borramento devido ao movimento, sem o que cada pixel depois de exposto é enviado diretamente para o campo de armazenagem de área, com um mínimo de atraso entre a exposição e armazenagem. Na arquitetura ILT (interline-transfer), cada coluna de fotolocais tem um registrador de deslocamento de campo de luz VCCD adjacente a ele. Um sensor ILT consiste de uma coluna vertical de fotolocais alternando com uma coluna de área de armazenagem (fig. 8). Estas áreas de armazenamento são registradores lineares de deslocamento CCD (VCCDs) protegidos da luz por uma camada constituída de uma máscara de alumínio. Estas colunas de registradores de armazenagem estão todas sincronizadas com um registrador HCCD (horizontal shift register), que está ligado ao amplificador de saída.

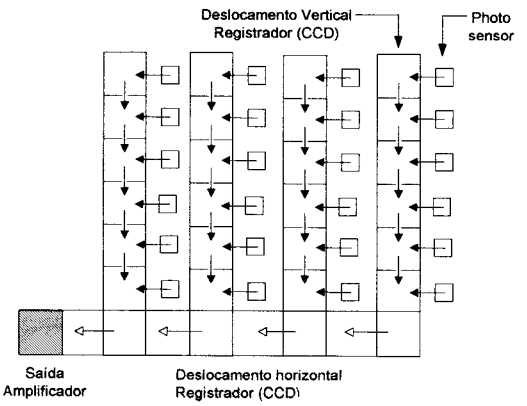


Fig. 8 - Arquitetura de um sensor ILT (interline-transfer).

As principais vantagens do sensor ILT estão no baixo ruído e o bom controle de exposição, sendo fabricado principalmente pela Sony, Nec, Elmo e Toshiba.

Sensor FIT (frame-interline transfer): o projeto do sensor FT e do sensor ILT, apresenta vantagens e desvantagens. Por exemplo, o arranjo ILT apresenta distorção vertical, e uma pequena área de luz, quando comparado com a grande porcentagem de área destinada ao controle, armazenagem e estrutura de transporte. A área total destinada aos sensores de luz é reduzida, limitando a sensibilidade. A introdução do espaçamento entre os pixels reduz o MFT (resolução espacial) do sensor e pode produzir um pixel não quadrado. O sensor de arranjo FT utiliza 100% da área para conversão da luz e apresenta pixels quadrados. Mas isto pode ser prejudicial por produzirem manchas em situações em que envolvem movimentos.

A arquitetura ILT é relativamente livre de borramento devido ao movimento e apresenta também flexibilidade no controle de exposição.

Um circuito integrado imageador híbrido CCD combina as características de projeto do FT e do ILT, formando o FIT (frame-interline transfer) (fig. 9).

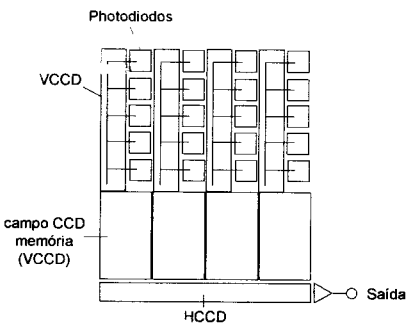


Fig. 9 - Arquitetura do sensor FIT (frame-interline transfer).

Este tipo de sensor é a combinação básica das arquiteturas ILT e FT. O formato do FIT é essencialmente o arranjo FT ligado a uma área de campo de armazenagem igual ao da arquitetura FT. O sensor FIT combina as melhores características de ambas arquiteturas ILT e FT. O uso de área separada, por exemplo, memória de frame como do projeto do ILT, reduz a dispersão de luz e distorção vertical.

O uso dos registradores de armazenagem no projeto do FIT dá um bom controle sobre fluorescência e exposição, minimizando o efeito de borramento devido ao movimento.

Os principais fabricantes de câmeras que contêm a arquitetura FIT são Nec e Toshiba.

Considerações que justificam o uso de câmeras CCD

A seguintes considerações contribuem para justificar a predominância do uso das câmeras CCD para aplicações em sistemas de visão:

- opera livre de manutenção, não susceptível à degradação com o tempo.
- Não precisa de aquecimento.
- Não apresenta deslocamento de imagem.
- Resistência ao campo magnético estático.
- Leve e de tamanho reduzido.
- Não apresenta distorção geométrica de imagem.
- Alta resistência a impacto e vibração.
- Não apresenta envelhecimento quanto à imagem de entrada.
- Mínimo atraso na imagem.
- Baixa potência de consumo.

Características de desempenho das câmeras CCD.

Na escolha de uma câmera CCD, durante o desenvolvimento de um projeto de sistema de visão de máquina, é extremamente importante o conhecimento das características e propriedades abaixo:

- Resolução espacial/MTF.
- Sensibilidade/resposta.
- Detectividade (mínima resposta).
- Linearidade.
- Atraso/inércia.
- Alcance (amplitude) dinâmico.

- Sensibilidade espectral.
- Fluorescência.
- Ruído.
- Distorção geométrica.
- Longa estabilidade.
- Tempo de varredura/taxa de exibição.
- Dados mecânicos e elétricos.

A seguir, temos um breve comentário sobre os parâmetros acima mencionados.

- **Sensibilidade/resposta** – Estes parâmetros são indicações da quantidade de luz necessária para se ter um nível de uma dada saída. A sensibilidade de um foto sensor é a saída elétrica (corrente em ampères) e o fluxo de entrada (em watts). A saída é, às vezes, dada como uma tensão equivalente. Note que é uma simples expressão de ganho. Seu real sentido, sensibilidade/resposta, indica o ganho do sensor, mas também note que é importante especificar a unidade de entrada e de saída.

A sensibilidade é característica de classificação do fotosensor. Ela pode ser uma resposta por unidade de área. A reposta pode ser a voltagem gerada na saída da câmera por unidade de energia de fluxo de luz. A resposta é um sinal de saída (a fotocorrente em ampères) por unidade de luz irradiada (em w/cm^2). Como ampères/segundo = Coulomb, e watt x Seg = Joule, a resposta pode ser também ser expressa como a soma de cargas q (em coulomb) acumulada no fotodiodo por unidade de exposição (J/cm^2). A unidade de resposta Γ é assim dada em unidade de $ampères.cm^2/w$ ou em $coulomb.cm^2/J$. A soma das cargas pode ser convertida para uma voltagem pelo uso de dispositivos de sensibilidade de difusão e deslocamento que resulta a característica Sc, que por padrão é $1\mu V$ por elétron. Assim, uma forma comum de expressar a medida de resposta, pode ser dada em $\mu V.cm^2/J$. Note que as cargas $q = C.V$, e que $A = C\ (dv/dt)$. Assim, utilizamos dois pontos inclinados na determinação de Γ empiricamente.

A expressão formal para a resposta Γ de um sensor (em unidade de $V.cm^2/\mu J$) é dada por:

$$\Gamma = 1 \times 10^{-6}.Ac\,.\,\eta\,.S.\lambda\,.\beta\,/1{,}24.\,E$$

onde Ac é a área do detector em cm^2. η é o QE, S é a sensibilidade de difusão de deslocamento, λ é o comprimento de onda em micrômetro, β é o ganho de saída do amplificador e **e** é a carga do elétron (em coulombs).

A resposta é também, algumas vezes, dada em ampères ou em carga de sinal (pico Coulombs). Por exemplo, a resposta de um fotodiodo é tipicamente 1μ A.cm^2/mW. A resposta de um fotosensor é dado em V.cm^2/μJ ou em V.cm^2/nJ. Em termos de exposição de luz, nJ/cm^2 pode ser encontrado pela multiplicação da luz irradiada (mw/cm^2) pelo tempo de exposição (ms), por que watts é equivalente a J/s.

É importante não confundir resposta com detectividade. Esta última determina a soma de fluxo incidente mínimo aceitável para o sensor fornecer uma resposta de saída.

A resposta de exposição é altamente dependente do tamanho da área de um pixel. A tabela 1 mostra o tamanho do pixel para diversas câmeras.

Câmeras	Pixel dimensões (μ^2)	Matriz resolução
RCA TC 2800/2900	16 X 20	403 X 512
Hitachi KP-120U	27 X 27	320 X 244
Hitachi VK- M98E	23 X 12	384 X 577
GE TN2505	24 X 28	377 X 242
GE TN2506	23 X 22	400 X 288
GE TN2507	23 X 14	377 X 484
Pulnix TM34K/36K	23 X 13	382 X 491
VSP SC500	10 X 16	604 X 576
EEV P4310	22 X 22	385 X 576
Fairchild CCD3000	30 X 18	380 X 488
Fairchild CCD4000	36 X 36	256 X 256

Tabela 1 - Tamanho do pixel em função do tipo de câmera.

A resposta de exposição varia com o comprimento de onda.

- **Detectividade e ruído** – Resposta de exposição que corresponde a uma medida que possibilita ao detector converter o fluxo óptico em um sinal elétrico. Um alto valor de resposta de exposição é bom, em geral, porque ele produz uma alta saída de sinal para uma dada radiação de fluxo. Entretanto, isto é acompanhado de um custo um tanto alto de nível de ruído, o qual limita o detector a sinais muito baixo. "Resposta de exposição não dá a indicação de como e de que modo um pequeno fluxo óptico e ou ausência de sinal pode ser detectado (Pinson, 1986)". Em outras palavras, ele não dá informação sobre o ruído a terra da qualidade do sensor.

Uma maior capacidade sensitiva de imageamento de uma matriz CCD é realizada somente quando vários níveis de sinais baixos podem ser processados, o que quer dizer que o nível de ruído pode ser exatamente pequeno. É desejável que um sensor de imagem deva operar a um nível baixo de iluminação.

A mínima resposta de exposição é o nível mais baixo possível de luz ao qual você pode adquirir a ausência e uma aceitável imagem (Lake, 1992a).

Detectividade de um sensor é a sua mínima resposta à exposição ou o mínimo sinal óptico detectado.

O ruído pode ser definido como dados que são irrelevantes ou de menor sentido, e que não estão relacionados com a imagem.

O nível e o ruído em uma imagem serão sempre considerações importantes, isto é, de importância particular quando o nível do sinal está baixo. O ruído apresenta-se como uma variação ou flutuação estatística de um nível de sinal nominal. O ruído de terra é usualmente definido quantitativamente pelo número de elétrons acumulado na ausência de sinal.

Os fabricantes referem-se, muitas vezes, ao desempenho de um sensor usando o SNR (signal-to-noise ratio - razão sinal/ruído), a razão ou proporção da amplitude de um sinal pela amplitude do sinal de ruído. Na prática esta é expressada em decibel, e, para uma imagem de qualidade satisfatória, o SNR deve ser igual ou superior a 45 dB.

O SNR refere-se ao alcance dinâmico do dispositivo, a razão ou a relação de saturação do sinal para o sinal de ruído.

A detectividade de um sensor é definida como o nível de entrada de energia para SNR igual a um (1). Para câmeras de radiodifusão e TV, em termos de fotometria ela é especificado como o mais baixo nível de luz possível que dá uma imagem útil, tipicamente um (1) lux (0.1fc) com uma fonte f/1.4. Para um dia ensolarado, o nível de luz é de mais ou menos 10.000 lux, enquanto que para uma noite sem nuvem e enluarado, 0.3 lux, para um local de trabalho adequado, 10 lux.

A maioria das câmeras de vídeo do estado-sólido tem uma sensibilidade padrão de 1.0 lux, e varia podendo ir abaixo de 0.1 lux, que é já completamente apropriada para trabalhar no escuro.

- **Blooming** – Os dois problemas que estão associados com a saturação do sensores são: o alto brilho e a distorção horizontal.

O blooming aparece como um alto brilho ou borrão, reduzindo o contraste na imagem, que ocorre devido a uma iluminação intensa, causando uma fotocarga em um ou mais fotolocais um potencial além do seu limite e estendendo e espalhando aos locais adjacentes. Enquanto que a distorção horizontal (streaking) ocorre quando as cargas ultrapassam a capacidade dos fotolocais no registro de armazenamento.

O blooming é bem menor (causa um menor problema) em matrizes de fotolocais que são isolados um do outro.

Os CCDs são particularmente suceptíveis ao blooming, porque eles são fisicamente adjacentes.

Além das considerações acima, para decidir o tipo de câmera a ser utilizada, devemos fazer a opção de escolha do tipo e formato de sinal de vídeo a ser trabalhado, o qual definirá a forma de aquisição da imagem que pode ser analógica ou digital.

2 - AQUISIÇÃO DE IMAGEM ANALÓGICA

Formatos de vídeo analógico em níveis de cinza e de cor são muito amplamente usados para formatos de vídeo no mundo de hoje. Por exemplo, vídeo colorido (comumente chamado de vídeo de cor composta – color-composite vídeo) é usado por todas estações de TV como um padrão de formato de transmissão, por todas as câmaras/gravadores, e por vários fabricantes de câmeras.

Os formatos padronizados de vídeo composto são EIA RS-170 e CCIR para vídeo moncromático e NTSC, PAL e SECAM para vídeo colorido.

. **vídeo monocromático** – EIA RS-170 e CCIR.

- O EIA RS-170 - usado na América do Norte e Japão, este sinal composto monocromático tem uma resolução espacial de 640 pixels por 480 linhas. O RS-170 opera a 60 Hz, o qual é equivalente a 30 FPS (Frames for seconds – quadros ou imagem completa por segundo).
- CCIR - usado primeiramente no Norte da Europa, este é um sinal composto chamado de CCIR (Comité Consulatif International Radio), nome de uma organização de padrão internacional. Este é um sinal monocromático que tem uma resolução espacial de 768 pixels por 576 linhas. O CCIR opera a 50 Hz, que equivale a 25 FPS.

. **vídeo colorido** – NTSC, PAL e SECAM

- NTSC/RS-330 - Usado na América do Norte e Japão, este sinal composto é equivalente ao RS-170 e une as mesmas especificações, exceto que a informação de cor é superposta sobre o sinal monocromático. Este tipo de sinal é padronizado pelo Comitê do Sistema Nacional de Televisão, sendo oficialmente adotado nos Estados Unidos da América, em 1953, após aprovação pela FCC (Federal Communication Commission – Comissão federal de comunicações). Utilizado na maioria dos países que tem tensão da rede elétrica a 60 Hz.
- PAL - o sistema PAL (Phase Altermnation Line), foi desenvolvido pelo Dr. Walter Bruch, engenheiro da Telefunken na Alemanha, usado no Norte da Europa, este sinal composto é equivalente ao CCIR e une as mesmas especificações, exceto que a informação de cor é superposta sobre o sinal monocromático. O

sistema PAL é bem superior aos sistemas NTSC e SECAM, em condições de péssimas de recepção. É utilizado na maioria dos países que tem tensão da rede elétrica a 50 Hz, e sendo adaptado para o uso em alguns países onde a rede elétrica é baseada a 60 Hz.

- SECAM - sistema francês, usado na Europa, Rússia e suas Repúblicas, este sinal composto é equivalente ao CCIR e une as mesmas especificações, exceto que a informação de cor é superposta sobre o sinal monocromático. O SECAM, criado pelo engenheiro Henri de France, é o padrão para Seqüencial Couleur Avec Mémoire (Seqüencial Color with Memory).

Resumindo, temos que os dois principais formatos para cor são: NTSC (National Television Systems Committee) na América do Norte e Japão, e PAL (Phase Alternate Line) na Europa. O mais comum padrão monocromático na América do Norte e Japão é o vídeo RS-170 da EIA (Eletronic Industries Association). O mais comum padrão monocromático na Europa é o CCIR. Temos ainda o sistema PAL-M, usado no Brasil desde 1972, o qual é união do sistema colorido alemão(PAL) compatibilizado com o padrão preto e branco dos USA, dando origem ao melhor sistema de cores do mundo com características de freqüência de canais, FI de vídeo e freqüência de varredura de padrão M, semelhante ao usado nos Estados Unidos da América. A denominação é PAL-M por que se trata de um sistema de TV "PAL" com um padrão "M" de transmissão.

Vários fabricantes de câmeras de vídeo fornecem, freqüentemente, câmeras com saída de vídeo composto colorido em formatos NTSC ou PAL associando os componentes de luminância (brilho) e crominância (cor) em um sinal analógico simples. Pode-se utilizar uma câmera colorida para adquirir imagens em níveis de cinza de um sinal de vídeo colorido (NTSC ou PAL). No circuito interno da câmera, um filtro anticrominância é aplicado ao sinal para remover a informação de cor. Selecionar o filtro é como transformar a cor em níveis de cinda de 8 bits (256 níveis).

Temos ainda os sinais de vídeos não padronizados ou despadronizados, que variam de resolução espacial, sinal de sincronização, ou outras características de sinais.

VÍDEO ANALÓGICO

Em um sinal de vídeo analógico, o pulso de sincronismo vertical (VSYNC) identifica o começo ou início do campo de vídeo e (HSYNC) também identifica o começo ou início do pulso horizontal de cada linha horizontal no campo de vídeo. Um campo é adquirido e completado no monitor à razão de 60 Hz (RS-170). Um quadro de uma imagem inteira (dois campos) de dados é completado a razão de 30 Hz (RS-170). O efeito do entrelaçamento dos

campos a 60 Hz resulta em uma rápida e perceptível imagem completa aos olhos humanos. Entretanto é importante lembrar que só depois de completados os dois campos em um quadro de vídeo entrelaçado, a imagem inteira é completada em somente 30 Hz.

VÍDEO COMPOSTO NO SINAL ANALÓGICO

No sinal analógico de vídeo composto colorido, a informação de cor (croma) é modulada em fase e amplitude através de uma onda subportadora na freqüência de 3,58 MHz (NTSC) ou 4,43 MHz (PAL). Ao sinal modulado é adicionada a informação de luminância e o sinal completo inclui os pulsos de sincronização e sendo transmitidos através de um simples sinal de linha.

AQUISIÇÃO DE IMAGEM DE COR ESTÁTICA (STILLCOLOR)

Cor estática (stillcolor) é uma nova técnica para aquisição de imagens coloridas a 24 bits, a partir de câmera composta usando somente 8 bits (256 níveis) (IMAQ 1408 série). Para adquirir imagem à cor, o algoritmo stiillcolor adquire múltiplos quadros (frames) de imagem de cor composta usando 8 bits (256 níveis) de acordo com o alcance dinâmico da placa. Estes números de quadros são processados para criar a imagem à cor.

A técnica stillcolor é ideal para o uso em aplicações que requerem alta qualidade de captura instantânea de objetos estacionários ou em movimento lento (câmaralenta). Os objetos serão estáveis para 70 ms (NTSC) e 120 ms (PAL). O stillcolor é compatível com vários diferentes formatos de saída em aplicações científicas e industriais. Dentro deste contexto, estão os bitmaps RGB, em plano simples de luminância, matiz e saturação, em formato de 8 e 16 bits.

CAMERAS COM PADRÃO DE VÍDEO ANALÓGICO

Os sinais de vídeo podem vir de uma grande variedade de fontes de vídeo, incluindo câmaras de vídeo, gravadores, câmara e gravador de vídeo, radiodifusão em TV, equipamentos de raio-X, microscópio eletrônico SEM (Scanning Eléctron Microscopes), scanner CT, e muitos outros. Estas fontes fornecem um sinal de vídeo composto (um sinal que contem os dados de vídeo e o tempo da informação) ou um despadronizado sinal de vídeo em que o vídeo e o tempo de informação podem variar em formato, assim como em um simples ou múltiplos sinais.

Para várias aplicações, é fácil configurar e adquirir imagens de uma câmera que usa vídeo padrão. Muitas câmeras usam o padrão popular de vídeo que descreve a sincronização da informação VSYNC (sincronismo vertical) e HSYNC (sincronismo horizontal), assim como também o tamanho de uma imagem.

CÂMERAS RGB(Reed-Green-Blue)

RGB é um padrão popular de cor que usa 8 bits para cada matiz de cor vermelha, verde e azul. O R, G e B digitaliza os valores combinados e cria a imagem à cor usando a técnica de cor estacionária (stillcolor).

CÂMERA DE VARREDURA PROGRESSIVA

As câmeras de varredura progressiva são projetadas para serem usadas em aplicações industriais nas quais os objetos a serem imageados (capturados) estão em movimento.

Em vez de adquirir um campo da imagem e entrelaçar os campos para o monitor ou ser mostrado, a matriz CCD em uma câmera de varredura progressiva adquire a cena de uma vez. Por exemplo, se usar a saída RS-170 da câmera em uma aplicação em movimento, há um pequeno (slight) atraso entre a exposição e a aquisição de cada um dos dois campos em um único quadro (imagem completa). Se há movimento na cena, este atraso na exposição e aquisição causará borramento na imagem adquirida. Cada campo é adquirido com uma pequena diferença de tempo. As câmeras com varredura progressiva eliminam este problema pela aquisição do quadro inteiro de uma só vez, isto é, como um campo não entrelaçado. Podem-se utilizar recursos de solftware para o caso em que se tenha uma cena de imagem em movimento e somente uma câmera RS-170 com vídeo entrelaçado, para varrer somente um campo a cada quadro. Os borramentos serão eliminados na imagem adquirida, entretanto, terá somente uma metade da resolução espacial na imagem adquirida.

CÂMERAS INFRAVERMELHA E TÉRMICA

As câmeras com sensores inframervelho e câmeras térmicas são disponíveis para aplicações em máquinas de visão, aplicações científicas e industriais. Câmeras infravermelha e térmica estão agora com um custo efetivo e, não por muito tempo, requerem um ventilador (coolant) para minimizar os ruídos térmicos do sensor CCD.

CÂMERAS ANALÓGICAS DESPADRONIZADAS

Câmeras fora do padrão são fabricadas a fim de serem utilizadas para captura de imagens de grande tamanho, para aplicações específicas. Podemos configurar a placa de captura de alguns fabricantes para adquirir vídeo de câmeras fora do padrão que tenham grandes imagens, transmitindo as imagens sem entrelaçamento (câmeras progressivas) assim como câmeras que transmitem imagens com variadas taxas de quadros (frames) (varredura variável).

3 - AQUISIÇAO DIGITAL DE IMAGEM

As câmeras digitais apresentam várias vantagens sobre as câmeras analógicas. Por digitalizar a câmera CCD antes da placa de aquisição, a razão sinal-ruído (SNR) é tipicamente alta, a qual resulta em melhor exatidão. Por sua vez, o vídeo analógico é mais susceptível ao ruído durante a transmissão do que o vídeo digital. As câmeras digitais vêm agora com 10 a 16 bits de resolução de níveis de cinza como característica padrão. Esta alta resolução é oferecida e requerida em aplicações médicas, máquinas de visão, astronomia e imagens térmicas. Existem placas para aquisição digital de imagens projetadas para trabalhar com câmeras digitais de vários fabricantes. Estas placas são ideais para rápidas aquisições de imagens digitais, grandes imagens e alta resolução.

INTERFACEAMENTO PARA CÂMERAS DIGITAIS

As câmeras digitais típicas têm um ou mais canais de 8, 10, 12, 14, 16 ou 32 bits (chamados taps). Estes canais são os fluxos de dados digitalizados da matriz de sensor CCD e podem ser lidos na saída em paralelo da câmera para uma aquisição rápida da imagem digital. Não há padrão de como os dados digitais são organizados e transferidos de uma câmera digital. No entanto, é importante ter um equipamento de aquisição de imagem digital e o software que seja fácil de configurar. Em adição, várias câmeras digitais são controladas por outras interfaces seriais ou linhas digitais de entrada/saída. Assim, quando selecionar uma placa, assegure que esta tenha tal capacidade. As placas de aquisição de imagem digital podem vir, por exemplo, com 4 canais de 8 bits de dados; 2 canais de 10, 12, 14 ou 16 bits de dados ou 1 canal de 24 ou 32 bits de dados.

CÂMERAS DIGITAIS E ALTA VELOCIDADE DE TRANSFERÊNCIA

Câmeras digitais proporcionam alta velocidade de saída de imagem. Algumas câmeras digitais podem ter saída de dados com taxas acima de 100 Mbytes/s. A placa PCI-1424 da NI(National Instruments) tem 50 MHz de pixel de clock e pode adquirir dados à razão de 200 Mbytes/s. Em adição, possui um buffer de memória na placa digital. Com uma memória buffer na placa, pode-se adquirir taxas extremamente altas e sustentar uma alta velocidade de transferência além de um grande desempenho do sistema.

CÂMERAS DIGITAIS DE VARREDURA DE LINHA

Câmeras de varredura de linha são, freqüentemente, usadas em aplicações industriais para fazer medida de altura e largura. Diferente de uma câmera de varredura de área, uma câmera de varredura de linha adquire uma imagem que é somente um ou poucos pixels distantes. A câmera de varredura de linha usa uma única linha de sensores CCD em vez de uma matriz retangular. A câmera é focalizada em uma linha próxima da parte móvel que passa, ela adquire linhas a uma razão muito rápida. A câmera de varredura de linha é especialmente útil para inspecionar partes arredondadas ou cilíndricas, porque como as partes giram, suas superfícies podem desenrolar em uma longa fita de imagem que é facilitado para o processo. Completando, técnicas de iluminação não são complexas para as câmeras de varredura de linha, porque precisa da luz somente para varrer a linha da parte e não toda á parte. As câmeras de varredura de linha são preferidas para aplicações em que a imagem está normalmente em movimento, assim como linha de sistemas de produção.

4 - RESOLUÇAO ESPACIAL

Resolução espacial define o número de elementos em uma imagem e é expressa como uma matriz de linhas por colunas. As linhas definem o comprimento ou extensão da imagem representada em linhas e as colunas definem a largura da imagem representada em pixels. Para o padrão de imagem RS-170/NTSC, a resolução espacial é de 640 pixels por 480 linhas (640x480), para o padrão de imagem CCIR/PAL, a resolução espacial é de 768 pixels por 576 linhas (768x576).

Dependendo da fonte de vídeo ou câmera usada, a resolução espacial pode alcançar de 256x256, elevando a 4096x4096 ou superior. Assim a resolução espacial afeta diretamente o tamanho da imagem; em várias aplicações, use somente a resolução requerida. Para transferências rápidas de imagens e manipulações, as quais são de grande importância em rápidas inspeções em aplicações industriais, as imagens usualmente têm a resolução espacial na faixa de 512x512. Para outras aplicações que necessitam de alta resolução espacial, assim como alta exatidão como aferição e aplicações de medida, a resolução de 1024x1024 e superior são freqüentemente usadas.

5 - ASPECTO DE RELAÇÃO/PROPORÇÃO

O aspecto de relação refere-se ao pixel individualmente, ou seja, a relação entre seu comprimento e sua largura.

Normalmente, é necessária a relação de 1:1, a qual é a média que o pixel tem de sua extensão e sua altura ou quadrado. Algumas entradas de fontes, câmeras e placa de captura de vídeo não criam ou convertem dados de vídeo em pixel quadrado. Estas freqüentemente criam o efeito de um círculo de aparência ovular (amassado no topo e em baixo), ou assemelhante a um quadro e retângulo.

O aspecto relacional é importante em certas operações e processos, como por exemplo, quando se deseja determinar a área de uma região pela contagem de pixels, ou operações de grupo como convoluções onde a imagem está sendo processada. Se o aspecto relacional não é 1:1, pode-se compensar durante o processamento da imagem ou durante a calibração do software de imagem.

6 - RESOLUÇÃO DE BRILHO

Quando os dados de vídeo são criados e convertido este deverá ser conforme certa resolução de brilho (às vezes, chamada resolução de profundidade de bit). Resolução de brilho define o número de variação da graduação de níveis de cinza ou gradiente disponível existente em uma imagem. Este gradiente é tipicamente referido ao nível de cinza (para imagens monocromáticas) ou número de cores (para imagens coloridas). Para imagens no padrão RS-170, a resolução de brilho é 8 bits ou 256 níveis de cinza (a nomenclatura é comumente mostrada para uma imagem RS-170 é 640x480x8). A resolução comumente usada é 8 bits (256 níveis de cinza), 10 bits (1024 níveis de cinza), 16 bits (65.536 níveis de cinza) e superiores. O aumento de dados contribui para o aumento da imagem assim como também quando a resolução de brilho aumenta. Por exemplo, uma imagem padrão RS-170 é aproximadamente de 308 KB (640x480), a 16 bits a mesma imagem terá resolução de aproximadamente 614 KB, e a 24 bits a imagem colorida (RGB) será de aproximadamente 922 KB.

7 - FORMATO ENTRELAÇADO E NÃO ENTRELAÇADO

Um sinal de vídeo contém uma série de linhas de pixels. Um pulso de sincronismo chamado de sincronismo horizontal são linhas separadas uma da outra. Todas as fontes de vídeo composto, incluindo RS-170/NTSC, CCIR/PAL e algumas fontes não padronizadas, transmitem uma linha de dados no formato entrelaçado. Entrelaçamento é o processo de envio de dados de vídeo em duas separadas porções chamados campos. O número de linhas de vídeo ímpar são transferido primeiro; em seguida, um número igual de linhas pares. A imagem completa que

IFSC-USP SERVIÇO DE BIBLIOT... INFORMAÇ...

contém as linhas ímpares e iguais pares de campo (linhas pares), é chamada de quadro (frame). Cada campo é mostrado seqüencialmente, dando a percepção de que o quadro é completado duas vezes assim como ele realmente é. Um pulso de sincronismo chamado de sincronismo vertical determina quando termina e quando deve iniciar o próximo campo.

O entrelaçamento evita a cintilação quando é mostrado algum tipo de imagem, como gráfico e suas linhas. Algumas fontes de vídeo não entrelaçadas transferem dados de vídeo no formato não entrelaçado, este processo é comumente chamado de varredura progressiva. Em um formato não entrelaçado somente um campo é transferido. O único campo contém as linhas (ímpares e pares) da informação de vídeo.

Tenha em mente que quando o objeto está ou será visualizado em movimento, onde a entrada de uma fonte usa o formato não entrelaçado/varredura progressiva é freqüentemente mais vantajosa do que aquelas que usam o formato entrelaçado. O formato entrelaçado tipicamente causa borrões ou excesso de brilho (aliasing) quando visualizamos objetos em movimento porque os dois campos estão normalmente desalinhados um com o outro.

8 - TAXA DE QUADRO

Freqüentemente referenciado como quadros por segundo (FPS), a taxa de quadros é a taxa (razão) a qual um novo quadro é transferido e/ou mostrado. Para dar a aparência de um fluxo livre da imagem, no RS-170/NTSC a imagem é mostrada a 30 FPS e para imagem CCIR/PAL, a imagem é normalmente 25 FPS. Imagem com taxa abaixo disto pode dar a aparência de tremulação ou o movimento consecutivo será afetado.

9 - CONSIDEREÇOES IMPORTANTES PARA ESCOLHA DE UMA PLACA DE CAPTURA DE VÍDEO

Os equipamentos que produzem imagem ajudam a mostrar e analisar a informação visual representada pelo sinal de vídeo. A mostra eletrônica é executada por dispositivos de captura de vídeo, freqüentemente chamada de placa de captura de imagens ou de vídeo (Frame Grabber), que consiste em uma placa colocada em um microcomputador (PC). A placa de captura de imagem converte uma imagem (ou imagens) fornecidas por uma fonte de vídeo em uma matriz de pontos de dados, a qual pode ser armazenada digitalmente, processada e otimizada, e subseqüentemente, analisada ou projetada em um monitor de vídeo.

Quando, eletronicamente, se mostra um sinal de vídeo, o processo refere-se comumente a uma aquisição de imagem, do qual é primeiramente necessário conhecer o tipo do sinal de vídeo que se irá trabalhar.

A placa de captura de vídeo é, portanto, um dispositivo que captura e converte sinais de vídeo analógico em dados digitais (uma exceção são as placas de captura digital, as quais interfaceia a câmera digital e, entretanto, não converte dados). A seguir, temos especificações importantes, características e capacidades a considerar quando relacionamos uma placa de captura para imagem ou aplicações em máquinas de visão.

PROJETO DE PLACA DE CAPTURA

Uma simples e mais importante tarefa que a placa de captura desempenha é converter dados de vídeo. Dependendo dos fabricantes e projetos fim, uma grande variedade de circuitos e técnicas pode-se ter sido usada no desenvolvimento de uma placa de captura. Dois tipos básicos de placa de captura são disponíveis: aquelas projetadas para aplicações industriais e científicas e aquelas projetadas para aplicações em multimídia.

Placa de captura industriais e científicas converte dados de vídeo como o mais fiel possível, ou seja, sem alterá-los. Placa de captura para multimídia converte dados, portanto, por propósitos estéticos, alterando o dado par ser visto melhor. Devido à vasta diferença de projetos fim, as placas de captura normalmente não podem ser usadas permutavelmente. No entanto, muitas placas de multimídia são divulgadas como sendo universal para todas as aplicações de imagem.

Os circuitos projetados para placa de captura, a fim de serem utilizadas em multimídia podem induzir uma soma de variação de fase (flutuação de fase de alta freqüência) e ruídos. Em alguns casos, elas têm outros circuitos que podem significativamente alterar os dados de vídeo. Estas alterações, que não estão presentes nos dados originais, podem causar erros de medida, quando os dados de imagem forem posteriormente processados e analisados. Quando usado em aplicações que requerem alta exatidão (por exemplo: aferição, microscopia ou inspeção de disco) estas alterações podem ter efeitos danosos.

Quando se avaliar uma placa de captura para aplicações de uso industrial, é importante distinguir aquelas projetadas para uso industrial daquelas projetadas para aplicações em multimídia. As especificações típicas e critérios de projetos a considerar quando se seleciona uma placa de captura para aplicações industriais e científicas são descritas a seguir:

IMPEDÂNCIA DO CANAL DE ENTRADA – O uso de terminais de entrada descasados pode causar um sinal de vídeo a ser "refletido" para baixo do cabo, resultando um "fantasma" ou distorção do vídeo de entrada. Para prevenir a reflexão do sinal, assegure que a impedância de

entrada da placa de captura tenha especificação de 75 ohms. Este valor equilibra a impedância da fonte de vídeo, e resulta em uma imagem não distorcida.

FILTRO DE ENTRADA DE ALTA QUALIDADE – Filtros de crominância, que são usados sobre a placa de captura monocromática, para remover informação de cor do sinal de vídeo, permitem aquisição limpa e maior exatidão de análise. Se os filtros não são propriamente projetados, eles podem remover dados adicionais que são necessários para a exatidão da imagem. Entretanto, assegure somente filtro de alta qualidade do tipo "chanfro" que são usados para que a largura de banda seja preservada e somente dados de cor sejam removidos.

ENTRADA DE BAIXO RUÍDO e CONVERSÃO DE CIRCUITOS – Entrada e conversão de circuitos, se não foram cuidadosamente projetados, podem criar ruídos espúrios que alteram os dados de vídeo. Duas especificações de ruídos são importantes: ruído integral não linear e RMS (Root Mean Square). Ambas unidades de especificações destes são medidas em unidades chamadas LSB (Least Significant Bit), as quais são representadas como uma parte da exatidão da escala de níveis de cinza.

Integral não linear representa o desvio dos valores dos níveis de valores da escala de cinza, qual a placa adquiriu do valor atual da escala de cinza da imagem. Ruído RMS representa os ruídos espúrios adicionados pelo circuito da placa. Em ambos os casos, o baixo valor é a melhor especificação. Os valores da Integral não linear, menores do que 0,5LSB são considerados excelentes. Ruídos RMS de valores menores do que 0,7LSB são considerados muito bons, enquanto que ruídos RMS com valores menores do que 0,5LSB são considerados excelentes.

EXATIDÃO DE DIGITALIZAÇÃO E VELOCIDADE DO CIRCUITO DE RESPOSTA – Quando convertemos os dados de entrada, a placa de captura precisa de sincronização do sinal de vídeo de entrada. Este processo é executado usando um PLL (Phase Lock Loop) ou um DCS (Digital Clock Sync) no circuito localizado na placa de captura. Uma relação de especificação do desempenho de circuito é chamada de variação de fase, que é o erro de medida de sincronização, quantificando o erro induzido pelo circuito de clock A/D. Tipicamente, é medida em nanosegundos, variações de fase levam os dados da linha horizontal a ficarem desequilibrados, resultando em pixels não posicionados corretamente. Isto vem alterar a integridade dos dados de entrada. Quando medido este "erro de posicionamento" induz ao erro no valor de pixel. Um grande valor de variação de fase, um grande erro é induzido na medida dos dados.

Placa de captura para multimídia, freqüentemente, usa circuitos PLL, enquanto que placas de captura para aplicações científicas e industriais, usualmente, utilizam os circuitos de

alto desempenho DCS. Circuitos DCS apresentam baixas variações de fase do que os circuitos PLL devido à sua inerente estabilidade. Em adição, os circuitos DCS podem resincronizar muito mais rapidamente do que os PLLs quando a alteração de fluxo de vídeo ocorre, assim como quando a entrada da fonte é comutada a meio fluxo ou quando usando religação (reset) assíncrono da câmera. Se a placa de captura com PLL é usada com um religamento assíncrono (o qual está na saída somente uma imagem quando disparada, ou reinicia o fluxo médio de vídeo, quando disparado), o circuito não estará apto a sincronizar dados antes dele ter passado, freqüentemente, uma resposta de tempo muito baixa. Como resultado, uns poucos primeiros dados de quadros de imagem serão perdidos enquanto o circuito PLL atender ao sincronismo para a sincronização do sinal de entrada. Em contraste, o circuito DCS, pode sincronizar o sinal dentro do primeiro quadro. Para testes importantes de digitalização, exatidão e flexibilidade com câmeras que possuem religamento assíncrono, assegure que a placa usa um circuito DCS. Para variação de fase especificações, ±6ns máximo (±4 ns típico) é considerado bom, ±5 ns máximo (±3,5 ns típico) é considerado muito bom, e ±4 ns máximo (±2,5 ns típico) ou menor é considerado excelente.

RELAÇÃO DE ASPECTO

A razão da dimensão horizontal de uma imagem para sua razão vertical é o termo de sua relação de aspecto. Sinais RS-170 têm uma relação de aspecto de 4:3 a extensão (comprimento) do sinal é 1:3 do tempo da altura do sinal. O processamento da imagem está igualmente ligado à relação de aspecto de cada pixel ou cada elemento da imagem. Várias placas de captura de 60 Hz têm uma relação de aspecto de 5:4, enquanto várias placas de captura de 50 Hz tem uma relação de aspecto de 3:2. Contudo, com programas é possível trabalhar a relação de aspecto. Contorno ou forma do quadrado não distorcido, pois aparece quadrado, o círculo é visto circular.

Relação de aspecto é importante em certas operações de processamento, assim como a determinação da área de uma região pela contagem de pixels ou operações em grupo como convolução onde a região da imagem está sendo processada. Nestas operações, é preciso compensar os pixels quando a relação de aspecto não é 1:1. A razão de aspecto também é importante quando a imagem precisa ajustar um gráfico padrão. Entretanto, em aplicações que requerem precisão de medida de pixel, assegure que a placa de captura provê de pixel quadrado ou uma relação de aspecto 1:1.

OUTRAS CARACTERÍSTICAS IMPORTANTES DE HADWARE

MÚLTIPLAS ENTRADAS – Para certas aplicações, podemos precisar conectar mais do que uma fonte de vídeo, assim como múltiplas câmeras ou VCRs para uma placa de captura. Muitas

placas de captura são construídas com um circuito multiplexador de entrada que segue conexão acima de quatro fontes de vídeo. Dependendo da requisição da aplicação, pode-se ser preciso adquirir muitas fontes de vídeo simultaneamente (freqüentemente, usado em máquinas de visão para imagear os planos *x, y* e *z* de um objeto). Para este tipo de requerimento, a placa de captura precisa ter outras múltiplas ações de entrada simultânea nos canais de entrada disponíveis ou drivers de software que suporte múltiplas placas de captura em um único computador, que roda em paralelo para prover os canais necessários.

SELEÇÃO DAS FONTES DE SINCRONISMO – A fim de prover grande flexibilidade para aplicações especializadas, várias placas de captura provêem a possibilidade de selecionar a fonte de sincronização em adição à fonte de entrada de vídeo. Por exemplo, a metade da fonte de sincronismo pode ser provida sobre a mesma entrada como fonte de vídeo sobre um não usual canal de entrada ou assim como uma linha externa dedicada. Para operações adequadas e casamento de sincronismo de entrada de placas de captura para suas aplicações são necessárias.

RELAÇÃO ESPACIAL PROGRAMADA E CLOCK DE PIXEL – Quando uma fonte não padronizada de entrada é usada (aquelas que não aderem ao formato do padrão de formatos de resolução espacial e/ou de sincronização do formato de vídeo), é preciso variar a varredura da placa de captura. Este tipo de placa de captura segue com programa para resolução espacial, assim como para razão de clock fazendo a placa ser perfeitamente sincronizada com a fonte de vídeo. Quando selecionar uma placa de captura com varredura variável, assegure que todos os limites de programação para ambas resoluções espaciais e velocidade de pixel de clock satisfaça ou exceda as especificações da fonte de vídeo.

ENTRADA DE SINAL FLEXÍVEL – é o ganho programado e um ajuste da faixa de entrada (obtido pelo ajuste do valor do conversor A/D e valores de referência), que permite digitalizar um sinal despadronizado de vídeo e um sinal de vídeo em baixas condições de iluminação. Por exemplo, se parte do sinal está abaixo da faixa de digitalização da placa de captura, podemos ajustar o valor de referencia (offset) e o valor de ganho para elevar o nível de voltagem para uma faixa que pode ser digitalizada. Do mesmo modo, se parte do sinal está acima da faixa de digitalização da placa de captura, podemos usar o ganho e ajuste de referência para abaixar o nível de voltagem para a faixa que pode ser digitalizado. Este circuito, do mesmo modo, ajusta-se à entrada para casamento exato do sinal que está sendo alimentado, resultando em uma maior exatidão nos dados mostrados.

CORREÇÃO DO PULSO DE SÍNCRONISMO – Em aplicações nas quais fontes de entrada de ruído são usadas, assim como VCRs ou câmeras com cabos de comprimento longo, (picos de

voltagem) picos de ruídos presentes no sinal de vídeo podem causar a placa de captura a uma interpretação incorreta da sincronização do sinal da informação. Este resultado de borramento na aquisição é corrigido abaixando-se ou excedendo-se o pulso de sincronismo. Para aliviar este problema, algumas indústrias e placa de captura provê circuitos especializados que corrigem a informação de sincronização pela inserção e remoção de erros de pulsos de sincronismo, estabilizando, assim, a imagem a ser capturada. Esta característica é ideal se a aplicação sofre de ruído devido a fonte usada ou do ambiente na qual o equipamento está operando.

TABELAS DE CONSULTA LUT (Look-Up Table) – Freqüentemente, placas de captura provêem tabelas de consulta ou LUTs. Dois tipos de LUTs são disponíveis: tabelas de entrada ILUTs e de equilíbrio de cores LUTs. Estas são similares no projeto, porém, estes dois tipos de LUTS desenvolvem várias tarefas diferentes.

ILUTs estão no formato de RAM para faixa de valores de pixel de uma imagem enquanto ela está no domínio digital, antes de ser armazenada e mostrada. ILUTs são comumente usadas para faixas de escala de níveis de cinza ou correção sobre a imagem, assim como no ajuste de contraste, brilho, inversão (vídeo reverso) e também na multiplicação e divisão da imagem. Com isto, adianta-se muito o desempenho da tarefa ao se usar equipamento de captura de imagem (como oposição a sistema de software), pois esta característica é muito útil em aplicações de inspeção de alta velocidade em que a imagem manipulada precisa ser desenvolvida com maior velocidade possível.

A ILUT do equilíbrio de cores é uma RAM formatada que leva em conta a faixa da imagem, durante o tempo em que a imagem está sendo apresentada. Pelo uso da LUT de equilíbrio de cores, podemos manter a faixa da imagem mostrada, enquanto altera a ILUT original. Isto leva em conta para amostra reversa, falsa cor ou diferentes melhora na imagem sem despedaçar a linha fundamental da cor ou a escala de níveis de cinza fixada. Esta característica é útil quando os dados da imagem original precisam estar à esquerda em sua forma original, mas a imagem de dados mostrada precisa ser melhorada para facilitar a visualização.

MOSTRA DE DADOS DA IMAGEM E SOBREPOSIÇÃO DE VÍDEO – Freqüentemente, as aplicações requerem a mostra ao vivo de vídeo (30 FPS), assim bem como sobrepor (gráficos sobre telas, texto, cruzamento, assim por diante, sobre o topo do vídeo), para alto brilho em certos aspectos da imagem. Várias placas de captura provêem circuitos na placa para mostra de execução destas tarefas. Enquanto ela pode ser útil, este tipo de circuito não necessita duplicar o equipamento já existente no computador principal. Em adição, isto aumenta o preço da placa de captura e, freqüentemente, usa técnicas inconvenientes (assim como chaveamento de croma) para exceder uma camada usando circuitos de amostragem e memória na mesma placa. Um

melhor acesso é o uso de Microsoft's Direct Draw interface padrão, o qual permite levar em conta uma seqüência limpa, livre de cintilação (flicker) na mostra do vídeo, como ao vivo, animado, sobreposição não destrutiva. Essa técnica é suportada pelo Windows 95/98 e Windows NT, que foi originalmente desenvolvido na fabricação de vídeo game e disponível para vários flexíveis mostras e a capacidade de sobreposição sem a necessidade de adição de equipamento para mostra. Sobreposições podem ser feitas translucidamente em escala sobre a imagem, com efeito, não destrutivo e que não alteram os dados originais da imagem. Em adição, por não ter equipamento de mostra na placa de captura, não está impedido na limitação do desempenho, o qual freqüentemente não pode executar o desempenho gráfico adaptado já usado no computador principal.

MEDIDA DE IMAGEM E RECORTE – Medidas da imagem e corte podem ser importantes características, especialmente a aplicação que requer imagens para serem separadas e redimensionadas para serem destacadas (clipped) sobre os limites de uma janela e sobre uma imagem mostrada.

Medição é tipicamente desenvolvida usando um dos dois métodos: decimação ou interpolação. Decimação envolve o processo de descartar linhas ou pixels para reduzir o tamanho da imagem, enquanto a interpolação usa a técnica da média entre as linhas e pixels. Entretanto, ambas são úteis, escala interpolada retém mais detalhes da imagem como a imagem é escalada. Se o objeto de aplicação requer o uso de imagens escaladas, determine qual é a melhor técnica que adapta a seus requisitos.

Recorte da imagem permite a Região de interesse ROI (Region of Interest) a ser selecionada na saída de uma imagem total. Dados da porção da imagem da região são transferidos para o computador principal, enquanto os dados ao lado da região são descartados. Freqüentemente, usadas em aplicações de alta velocidade, ROIs segue para transferência rápida de imagem (devido ao pequeno resultado do tamanho da imagem) e reduz o processamento para aplicações em tempo crítico. Se ROIs são requeridos em aplicações, assegure que esta característica seja disponível e que o programa e o tamanho do ROI pode ser facilmente aplicado à aplicação requerida.

CONTROLE DA CÂMERA – Tipicamente, a câmera gera sinal de sincronismo que é embutido no fluxo de dados. A placa de captura tira este sinal de saída e utiliza-o para "broquear" o sinal de clock da câmera. Em algumas aplicações, entretanto, adiciona-se o controle sobre a sincronização da câmera ou a sincronização da placa de captura, se é requerido. Placa de captura que provê em separado sincronismo horizontal (HSYNC), sincronismo vertical (VSYNC), e clock de pulso de entrada e saída de sinal permitindo adicionar à câmera configuração flexível, assim como gen-locking (todo clock da câmera estão amarrado junto e o

sinal que roda a placa de captura) e sincronismo principal (o sincronismo de funcionamento de sincronização da câmera). Sinais de controle, assim como exposição e sinais de religação provêem igual controle de grandes câmeras são essencialmente úteis em aplicações especializadas.

ENTRADA/SAÍDA DIGITAL – Linhas digitais de entrada e saída digital presentes nas placas de captura permitem o controle no monitor de dispositivos externos ou periféricos sem ter que instalar uma placa separada de entrada e saída. Estas também facilita a tarefa de integração, sem adicionar drivers que são requeridos e os comandos usados para operar as entradas/saída digital estão disponíveis no conjunto de comandos da placa de captura. A entrada e saída de sinais destas linhas digitais são tipicamente níveis TTL, os quais são fabricados para compatibilidade universal com diversos dispositivos e periféricos. Quando selecionar uma placa de captura com I/O digital, assegure que todas as linhas I/O estejam providas das necessidades da aplicação.

ARQUITETURA DE TRANSFERÊNCIA DE DADOS - Em aplicações de imagem, transferência de dados em tempo real é essencial. Para acomodar a alta velocidade de transferência requerida para dados de vídeo, o barramento PCI (Peripheral Component Interface) é requerido e a placa de captura precisa ser projetada assim como o barramento principal. Pela operação com um PCI como barramento principal, a placa de captura pode transferir uma grande soma de dados contíguos em um bloco da memória do sistema no computador principal a uma velocidade acima de 132 MB/s. Esta arquitetura elimina a necessidade de memória e circuito de mostra na placa de captura e elimina o atraso de CPU, proporcionando ao computador principal livre desempenho de outras operações de processamento de imagens. Placas de captura com memória na placa não necessariamente duplicam a memória já presente no computador principal e aumentam o preço da placa. Em adição, memória na placa realmente atrasa a transferência da imagem depois de manter temporariamente o dado, transferindo-o somente após o buffer de memória ter sido ocupado.

SUPORTE DE SOFTWARE – Antes de selecionar a placa de captura cheque para ver que o software que está incluído, e determine quais características são benéficas para as necessidades da aplicação.

DESENVOLVIMENTO DE SOFTWARE – Para fazer a implementação de tarefas para uma placa de captura assegure que facilmente as propriedades da ferramenta de software suporte o sistema operacional disponível. A seguir, temos como exemplo uma lista de checagem

altamente comum às áreas de investigação quando integramos o equipamento e placa de captura em uma determinada aplicação.

. Você deseja criar suas aplicações em C? Se sim, assegure que os drivers para objetivo aceita o sistema operacional assim bem como o SDK (Software Developer's Kit) esteja disponível para fazer o trabalho de implementação da placa de captura em uma aplicação facilmente.

. Você deseja criar suas aplicações através de programas Visual Basic ou Visual C++? Se sim, assegure drivers para o sistema operacional, assim como também um Active X Controle, que sejam disponíveis para fazer o trabalho da placa de captura na aplicação facilmente.

. O seu plano de completar ou mudar para outros tipos de placas de captura (por exemplo, monocromática, colorida ou digital) no futuro? Se for usar múltiplos tipos de placas de captura em várias aplicações.

Se você respondeu sim a todas ou a duas destas questões, assegure que os drivers são escritos usando a mesma modular, API independente de hardware. Isto feito usa-se comandos comuns e funções de chamadas para todas as placas e, a seguir, pode-se migrar facilmente de uma placa para outra, sem ter que reescrever o código. Se usar múltiplas placas, assegure que os drivers suportam rodar em múltiplas placas, iguais e diferentes tipos, no mesmo sistema principal.

APÊNDICE D

ALGORITMO

E

CÓDIGOS FONTE

Algoritmo 1 - ALGORITMO PARALELO PARA O NÓ MESTRE

```
1.  Gerar a lista de Imagens
    Total_Escravos = Número_de_Escravos
    Total_Faixas = Total_de_Imagens/10
    i = 1;
    Rank = 1;

2.  WHILE(i < Total_Faixas and Total_Escravos < > ∅)
        SEND (i , 1 , MPI_INT, Rank , Tag , MPI_COMM_WORLD)
        i = i + 1;
        Rank = Rank + 1;
        Atualiza Tabela
        Total_Escravos = Total_Escravos – 1;

3.  WHILE (Total_Escravos < Número_de_Escravos)
        Recv (Vetor[90], 10 , MPI_FLOAT, MPI_ANY_SOURCE, TAG,
                MPI_COMM_WORD, Status)
        Verifica quem enviou Resultados(MPI_Status) → P1=MPI_Source
        Total_Escravos = Total_Escravos + 1;
        Atualiza Matriz;
        IF (i < Total_Faixas)
            SEND( i , 1 , MPI_INT, P1, TAG, MPI_COMM_WORD)
            i = i + 1;
            Atualiza Tabela;
            Total_Escravos = Total_Escravos –1;

4.  MPI_FINALIZE;

        Tratamento da Matriz
```

Algoritmo 2 - ALGORITMO PARALELO PARA O NÓ ESCRAVO

1. RECV (Faixa, 1 , MPI_INT, ∅ , Tag, MPI_COMM_WORD)

   Processamento da Imagem

2. SEND (Vetor, 9 , MPI_FLOAT, ∅ , Tag, MPI_COMM_WORD)

ALGORITMO 3 – PROCURA DO "THRESHOLD" USANDO A SELEÇÃO ITERATIVA

1. Início
2. Zerar o "threshold" do objeto (to), o "threshold" do fundo (tb), e o "threshold" entre fundo e objeto (t2)
3. Execute Pare = FALSO
4. Enquanto Pare = FALSO execute
   Calcular média do objeto (to) para os "pixels" ≥ tt
   Calcular média do fundo (tb) para os "pixels" < tt
   Execute t2 = (to + tb)/2
   Se t2 = tt, execute Pare = VERDADEIRO
   Senão continue

Fim.

ALGORITMO 4 – CÁLCULO DO HISTOGRAMA

1. Início
   Para i=0 até o (número_de_tons_de_cinza – 1) Faça p(i) = 0
2. Ler arquivo e colocar no buffer[tamanho_quadro_X * tamanho_quadro_Y)
3. Para i=0 até (tamanho_quadro_X * tamanho_quadro_Y)
   Execute incremente p[buffer(i)]
   Fim i
4. Para i=0 até (número_de_tons_de_cinza – 1)
   Execute p(i) = p(i)/(tamanho_quadro_X * tamanho_quadro_Y)
   Fim i

Fim

ALGORITMO 5 – CÁLCULO DA MÉDIA

```
1. Início
2.     Execute Soma=0
3.     Execute o histograma (p(i)) do quadro
4.   Para i=0 até (número_de_tons_de_cinza – 1)
       Execute Soma = Soma + i*p(i)
     Fim i
       Execute     Média_quadro = Soma/(tamanho_quadro_X * tamanho_quadro_Y)
 Fim.
```

ALGORITMO 6 – CÁLCULO DO DESVIO PADRÃO

```
1. Início
2. Execute histograma (p(i)) do quadro
3. Cálcule a média do quadro
4.   Para i=0 até (número_de_tons_de_cinza – 1)
       Execute   Soma = Soma +Sqr(b – média(i))
       Execute DesvP = Sqr (soma)/Max_Tom
     Fim i
 Fim
```

ALGORITMO 7 – CÁLCULO DA ENERGIA

```
1. Início
2.     Execute Entropia_quadro = 0
3.     Execute histograma (p(i)) do quadro
4.   Para i=0 até (número_de_tons_de cinza –1)
       Se p(i)≠0
       Execute   Energia_quadro = Sqr (Energia_quadro + p(i))
     Fim i
 Fim
```

ALGORITMO 8 – CÁLCULO DA ENTROPIA

```
5. Início
6.     Execute Entropia_quadro = 0
7.     Execute histograma (p(i)) do quadro
8.   Para i=0 até (número_de_tons_de cinza –1)
       Se p(i)≠0
       Execute   Entropia_quadro = Entropia_quadro + p(i).log(p(i))
     Fim i
 Fim
```

ALGORITMO 9 – CÁLCULO DA CURTOSE

```
1.  Início
2.      Execute CURTOSE_quadro = 0
3.      Execute histograma (p(i)) do quadro
4.   Para i=0 até (número_de_tons_de_cinza – 1)
        Execute Soma = Soma + (i – Média _quadro)****4 p(i)
        Execute CURTOSE_quadro = 1/(DesvP_quadro)***3) *Soma/Soma*(Max_Tom + 1)
     Fim i
  Fim
```

ALGORITMO 10 – CÁLCULO DA ÁREA

```
1.  Início
2.      Execute m00 {momento de ordem zero}
3.      Execute Área = m00
    Fim
```

ALGORITMO 11 – CÁLCULO DO EIXO MAIOR

```
1.  Início
2.    Execute m11,m02,m20 {momentos de segunda ordem}
3.    Execute alfa = sqrt((2*(m20 + m02) + (sqrt((m20 - m02)*(m20 – m02)+4*m11*m11))))/m00
4.    Execute Eixo_Maior = alfa
    Fim
```

ALGORITMO 12 – CÁLCULO DO EIXO MENOR

```
1.  Início
2.    Execute m11,m02,m20 {momentos de segunda ordem}
3.    Execute beta = sqrt((2*(m20 + m02) - (sqrt((m20 - m02)*(m20 – m02)+4*m11*m11))))/m00
4.    Execute Eixo_Menor = beta
    Fim
```

ALGORITMO 13 – CÁLCULO DA ELONGAÇÃO

```
1.  Início
2.    Execute m11,m02,m20 {momentos de segunda ordem}
3.    Execute fi = 0,5 ArcTang(2*m11/m20 – m02))
4.    Execute gama = sqrt((m20*Cos(fi)*Cos(fi) + m02*Sin(fi)*Sin(fi) –
                          m11*Sin(2*fi)*Sin(2*fi))/ (m20*Sin(fi)*Sin(fi))
5.    Execute Elongação  = gama
    Fim
```

```
/* cfopen.c -- open a file and write ints received from each process to it */

#include <stdio.h>
#include "mpi.h"
#include "cio.h"

#define MAX 32


FILE* Cfopen(
      char*    filename  /* in */,
      char*    mode      /* in */,
      MPI_Comm  io_comm   /* in */);

main(int argc, char* argv[]) {
   int      p;
   int      my_rank;
   int      io_rank;
   int      list[MAX];
   MPI_Comm  io_comm;
   int      i;
   FILE*    fp;

   MPI_Init(&argc, &argv);
   MPI_Comm_size(MPI_COMM_WORLD, &p);
   MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

   MPI_Comm_dup(MPI_COMM_WORLD, &io_comm);
   Cache_io_rank(MPI_COMM_WORLD, io_comm);

   fp = Cfopen("testfile", "w", io_comm);

   Get_io_rank(io_comm, &io_rank);
   MPI_Gather(&my_rank, 1, MPI_INT, list, 1, MPI_INT,
      io_rank, io_comm);
   if (my_rank == io_rank) {
      for (i = 0; i < p; i++)
         fprintf(fp,"%d ", list[i]);
      fprintf(fp,"\n");
      fclose(fp);
   }

   MPI_Finalize();
} /* main */
```

```
/* send_col_to_row.c -- send column 1 of a matrix on process 0 to row 1 */

#include <stdio.h>
#include "mpi.h"

main(int argc, char* argv[]) {
   int p;
   int my_rank;
   float A[10][10];
   MPI_Status status;
   MPI_Datatype column_mpi_t;
   int i, j;

   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

   MPI_Type_vector(10, 1, 10, MPI_FLOAT, &column_mpi_t);
   MPI_Type_commit(&column_mpi_t);

   if (my_rank == 0) {
      for (i = 0; i < 10; i++)
         for (j = 0; j < 10; j++)
            A[i][j] = (float) i;
      MPI_Send(&(A[0][0]), 1, column_mpi_t, 1, 0,
            MPI_COMM_WORLD);
   } else { /* my_rank = 1 */
      for (i = 0; i < 10; i++)
         for (j = 0; j < 10; j++)
            A[i][j] = 0.0;
      MPI_Recv(&(A[0][0]), 10, MPI_FLOAT, 0, 0,
            MPI_COMM_WORLD, &status);
      for (j = 0; j < 10; j++)
         printf("%3.1f ", A[0][j]);
      printf("\n");
   }

   MPI_Finalize();
} /* main */
```

```
/* Sobel edge detector - no templates */

#define MAX
#include "lib.h"
#include <math.h>

void thresh (IMAGE z)
{
        int histo[256];
        int i,j,t;

/* Compute a grey level histogram */
        for (i=0; i<256; i++) histo[i] = 0;
        for (i=1; i<z->info->nr-1; i++)
          for (j=1; j<z->info->nc-1; j++)
          {
           histo[z->data[i][j]]++;
          }

/* Threshold at the middle of the occupied levels */
        i = 255;
        while (histo[i] == 0) i--;
        j = 0;
        while (histo[j] == 0) j++;
        t = (i+j)/2;

/* Apply the threshold */
        for (i=1; i<z->info->nr-1; i++)
          for (j=1; j<z->info->nc-1; j++)
            if (z->data[i][j] >= t) z->data[i][j] = 0;
            else z->data[i][j] = 255;
}

/*      Apply a Sobel edge mask to the image X          */

void sobel (struct image *x)
{
        int i,j,n,m,k;
        IMAGE z;

        z = 0;
        copy (&z, x);

        for (i=0; i<x->info->nr; i++)
          for (j=0; j<x->info->nc; j++)
            z->data[i][j] = 255;

/* Now compute the convolution, scaling */
        for (i=1; i<x->info->nr-1; i++)
          for (j=1; j<x->info->nc-1; j++)
          {
            n = (x->data[i-1][j+1]+2*x->data[i][j+1]+x->data[i+1][j+1]) -
                (x->data[i-1][j-1]+2*x->data[i][j-1]+x->data[i+1][j-1]);
            m = (x->data[i+1][j-1]+2*x->data[i+1][j]+x->data[i+1][j+1])-
                (x->data[i-1][j-1]+2*x->data[i-1][j]+x->data[i-1][j+1]);
```

```
            k = (int)( sqrt( (double)(n*n + m*m) )/4.0 );
            z->data[i][j] = k;
          }

        thresh (z);

        for (i=0; i<x->info->nr; i++)
          for (j=0; j<x->info->nc; j++)
            x->data[i][j] = z->data[i][j];
}

main (int argc, char *argv[])
{
        IMAGE x=0;

        if (argc < 2)
        {
          printf ("Usage: %s <image> \n", argv[0]);
          exit (1);
        }

        x = Input_BMP (argv[1]);
        if (x == 0)
        {
          printf ("No input image ('%s')\n", argv[1]);
          exit (2);
        }

        sobel (x);

        Output_BMP (x, "sobel.bmp");
        printf ("Output is in file 'sobel.bmp'\n");
}



/*- Função para abrir imagem e aplicar a transformação dos níveis de cinza */

void ABRIR_IMAGEM()

{
 register int i, j;
 int m;
 char nome[40];
 register int i, j;
 int *pont;
 long int buffer1[256], maximo, minimo;

 is_set_active_region(xlinha,ycoluna,xaltura,ylargura);
/* procura dos valores máximo e mínimo da imagem  */
 is_histogram(0,buffer1);
 maximo=0L;
 for(j=0;j<256;j++) {
```

```
        if(buffer1[j]!=0 && maximo==0) minimo = (long int)j:
        if(buffer1[j]!=0) maximo = (long int)j;
  }
/* realiza a transformação linear nos níveis de cinza */
  for(i=xlinha;i<xlinha+xaltura;i++) {
      pont = (int*) malloc( (size_t) (coluna *sizeof(int)) );
      is_get_pixel(0,i,ycoluna,ylargura,&pont[0]);
      for(j=0;j<ylargura;j++)
          pont[j] = (int) ((64L*(pont[j]-minimo))/(maximo-minimo))*4;
      is_put_pixel(0,i,ylargura,coluna,&pont[0]);
      free(pont);
  }
}


/*- Função para binarização da imagem */

void BORDA()
{
  int i, j, x, y, t2i, tti, cont, v, tti2;
  long int n, no, nb;
  float to, tt, tb, t2, mediag;

  is_set_active_region(102,100,126,244);
  is_get_region(0,regiao);
  mediag = 0;
  for(i=0;i<126*224;i++) mediag += (float) regiao[i];
  mediag = mediag/(126.*224.);

  corte = 10;
  quadrox = ;
  quadroy = ;
  for(i=xlinha;i<xaltura+xlinha-5;i+=quadrox) {
    for(j=ycoluna+corte;j<ylargura+ycoluna-corte;j+=quadroy) {
      is_set_active_region(i, j, quadrox, quadroy);
      is_get_region(0,regiao);
      to = 0.;
      n = quadrox*quadroy;
      for(x=0;x<quadrox;x++) {
            for(y=0;y<quadro;y++) {
                to += (float) regiao[y+x*quadroy];
            }
      }
      tt = to/(float)n;
      tti = (int) tt;

      while(n) {
            no = 0;
            nb = 0;
            tb = 0.;
            to = 0.;
            for(x=0;x<quadrox;x++) {
              for(y=0;y<quadroy;y++) {
                if((float)regiao[y+x*quadroy]>=tt) {
                        if(regiao[y+x*quadroy]<238) to = to + (float) regiao[y+x*quadroy];
```

```
                no++;
            } else {
                if(regiao[y+x*quadroy]<238) tb = tb + (float) regiao[y+x*quadroy];
                nb++;
            }
          }
        }
        if(no==0) no = 1;
        if(nb==0) nb = 1;
        t2 = (tb/(float)nb + to/(float)no)/2.;
        t2i = (int) t2;
        if(t2 == tt) n = 0;
        tt = t2;
        tti = (int) tt;
    }
    for(x=0;x<quadrox*quadroy;x++) {
        if(regiao[x]<=tt && regiao[x]<=((int)mediag)) regiao[x] = 0;
        else regiao[x] = 248;
    }
    is_set_active_region(i, j, quadrox, quadroy);
    is_put_region(0, regiao);
  }
 }
}
```

```
/* Armazena os resultados obtidos em cada imagem em um arquivo de saida */
void GRAVA_RESULTADOS(void)
{
 int x;
 float med_tot, var_tot, ent_tot, idm_tot, con_tot, desv_tot, curt_tot, area_tot, eixma_tot,
eixme_tot, elong_tot;

 med_tot = 0.;
 for( x=0; x<10; x++) med_tot += media.valor[x];
 med_tot = med_tot / 10;

 var_tot = 0.;
 for( x=0; x<10; x++) var_tot += variancia.valor[x];
 var_tot = var_tot / 10;

 ent_tot = 0.;
 for( x=0; x<10; x++) ent_tot += entropia.valor[x];
 ent_tot = ent_tot / 10;

 idm_tot = 0.;
 for( x=0; x<10; x++) idm_tot += IDM.valor[x];
 idm_tot = idm_tot / 10;

 con_tot = 0.;
 for( x=0; x<10; x++) con_tot += conar.valor[x];
```

```
  con_tot = con_tot / 10;

  fprintf(arq_saida,"%f\t%f\t%f\t%f\t%f\t%f\n", med_tot, desv_tot, cur_tot, var_tot, ent_tot, idm_tot,
 con_tot, area_tot, eixma_tot, eixmen_tot,);


 }


 main()
 {
  int i, j, numero, vol;
  long int interacao;
  char nome[30], ch1[3];
  time_t inicio_tempo, fim_tempo;


  /* Criacao do arquivo de saida */
  if(!(arq_saida=fopen("saida.txt","w"))) {
     printf("nao eh possivel criar arquivo de saida para gravacao\n");
     exit(1);
  }


 /* Rotate the image X by the given angle, in degrees */

 void rotate (struct image *x, float angle, int *error_code0
 {
     int i, j, n, m, ci, cj;
     float sina, cosa, oldi, oldj ;
     double sin(), cos ();
     float v, t, xmax, xmin, alpha, beta;
     int NN, MM, ii, jj ;
     struct image *rim, *newimage();

     *erro_code = 0;
     angle = angle * (3.1415926535/180.0);   /*Convert to radians */
     MM x - >info - >nc;      NN = x->info->nr;

 /* Rotation assumes that the origin is the CENTER of X */
     sina = sin (angle);      cosa = cos (angle);
     ci = NN/2;      cj = MM/2;

     rim = newimage (NN, MM, error_code);
     if (*error_code) return;

 /* For each pixel in the rotated image RIM, find the nearest 4
    pixel in X and interpolate to find the grey level in RIM. */
     for (i=0; j <MM; j++) {
            oldi = (i – ci)*cosa – (j – cj)*sina + ci;
            oldj = (i – ci)*sina + (j – cj)*cosa + cj;
            ii = (int) oldi;
            jj = (int) oldj;
            alpha = oldi – (float)ii;
```

```
            beta   = oldj – (float)jj:
            rim ->data [i][j] = (unsigned char) bilinear (x, alpha, beta,
            ii, jj);
        }
    }

/*Copy rotated image into x */
        for (i = 0; i < NN; i++)
                for (j = 0; j , MM; j++) {
                        x - > data [i][j] = rim - > data [i][j];
                }
        freeimage (rim, error_code);
}

/* Perform a linear interpolation to get the value of the pixel
   that  is distance a (rows) and b (columns) from (ii, jj) in X. */

int bilinear (struct image *x, float a, float b, int ii, int jj)
{
        float y;

        it (ii < 0) || (ii >= x -> info nr – 1) return BACKGROUND;
        it (jj < 0) || (jj >= x -> info nc – 1) return BACKGROUND;
        if (is_zero (a)  && is_zero (b)  ) return x -> data [ii][jj];
        if (is_zero (a))   {
                y =  (1 – b)*x-> data [ii][jj] + b*x-> data [ii][jj + 1];
                 return (int) (y + 0.5);
        }
        if (is_zero (b))  {
                y = (1 – a)*x-> data [ii][jj] + a*x->data [ii + 1][jj];
                return (int) (y + 0.5);
        }
        y = (1 – a)*(1 – b)*x->data[ii][jj] + (1 – a)*b*x->data[ii][jj + 1];
        y += a*(1 – b)*x->data [ii + 1][jj] + a*b*x->data [ii + 1][jj + 1];
        return (int) (y + 0,5);
}
```

PROGRAMA EM "C" DE UMA REDE NEURAL TREINADA

```
/*******************************************************
 NET13
 -------------------------------------------------------
 generated at Tue May 21 16:26:41 2002
 by snns2c ( Bernward Kett 1995 )
*******************************************************/

#include <math.h>

#define Act_Logistic(sum, bias)  ( (sum+bias<10000.0) ? ( 1.0/(1.0 + exp(-sum-bias) ) ) : 0.0 )
#ifndef NULL
#define NULL (void *)0
#endif

typedef struct UT {
        float act;        /* Activation       */
        float Bias;       /* Bias of the Unit */
        int   NoOfSources; /* Number of predecessor units */
  struct UT   **sources; /* predecessor units */
        float *weights; /* weights from predecessor units */
      } UnitType, *pUnit;

 /* Forward Declaration for all unit types */
 static UnitType Units[22];
 /* Sources definition section */
 static pUnit Sources[] =  {
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units + 9,
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units + 9,
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units + 9,
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units + 9,
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units + 9,
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units + 9,
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units + 9,
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units + 9,
Units + 10, Units + 11, Units + 12, Units + 13, Units + 14, Units + 15, Units + 16, Units + 17,
Units + 18,
Units + 10, Units + 11, Units + 12, Units + 13, Units + 14, Units + 15, Units + 16, Units + 17,
Units + 18,
Units + 10, Units + 11, Units + 12, Units + 13, Units + 14, Units + 15, Units + 16, Units + 17,
Units + 18,

 };

 /* Weigths definition section */
 static float Weights[] =  {
-32.225269, 11.943150, -97.134323, -2.557170, -4.517730, 31.086550, 6.769630, 8.069460,
40.466499,
84.138107, -18.773640, -102.680832, 93.491318, 19.594879, 72.999069, -43.506599, -
34.313999, -30.896481,
94.122391, -36.149441, 128.432632, 50.319592, -24.806061, 26.706739, -40.094589, -
50.996208, -68.373154,
-55.279579, -1.155250, 50.110851, 4.560070, -2.043200, -18.012850, 5.215640, -2.940060, -
0.230810,
```

IPEN-USP SERVIÇO DE BIBLIOTECA E INFORMAÇÃO

```
4.202080, 0.321720, 11.816730, -19.147881, -1.676110, -21.959450, 7.353660, 3.422130.
9.638500,
12.713680, -5.165530, 73.109177, -10.350200, -0.056140, -21.331671, 4.278390, 1.044420, -
35.182259,
68.180931, 21.584709, -66.141724, -106.071800, 0.950760, -22.230499, 37.565849,
72.100479, -7.957800,
-14.612970, 16.465349, 11.051380, -29.077209, 13.698970, -105.745560, 47.470501,
51.850651, -13.526960,
-52.086399, 23.302771, 5.652940, -58.678391, -4.262520, -37.539410, 33.275780, 43.900940,
-2.089770,
-13.280120, 18.688881, 21.119270, -11.954580, -21.721720, -4.852780, -14.935070,
13.855540, -13.962870,
10.371070, -13.643490, -16.275221, 10.468550, 14.653240, 2.726320, 12.216710, 52.908539,
7.722190,
8.580010, 4.677080, 2.900600, -0.337640, -2.556140, -2.451730, -0.565540, -15.461680,
1.446290,

 };

 /* unit definition section (see also UnitType) */
 static UnitType Units[22] =
 {
  { 0.0, 0.0, 0, NULL , NULL },
  { /* unit 1 (Old: 1) */
   0.0, 1.000000, 0,
    &Sources[0] ,
    &Weights[0] ,
   },
  { /* unit 2 (Old: 2) */
   0.0, 0.998030, 0,
    &Sources[0] ,
    &Weights[0] ,
   },
  { /* unit 3 (Old: 3) */
   0.0, 0.916740, 0,
    &Sources[0] ,
    &Weights[0] ,
   },
  { /* unit 4 (Old: 4) */
   0.0, 0.646710, 0,
    &Sources[0] ,
    &Weights[0] ,
   },
  { /* unit 5 (Old: 5) */
   0.0, 0.270800, 0,
    &Sources[0] ,
    &Weights[0] ,
   },
  { /* unit 6 (Old: 6) */
   0.0, 0.817340, 0,
    &Sources[0] ,
    &Weights[0] ,
   },
  { /* unit 7 (Old: 7) */
   0.0, 0.815400, 0,
```

```
  &Sources[0] ,
  &Weights[0] ,
  },
 { /* unit 8 (Old: 8) */
  0.0, 0.025570, 0,
  &Sources[0] ,
  &Weights[0] ,
  },
 { /* unit 9 (Old: 9) */
  0.0, -0.053500, 0,
  &Sources[0] ,
  &Weights[0] ,
  },
 { /* unit 10 (Old: 10) */
  0.0, 8.840620, 9,
  &Sources[0] ,
  &Weights[0] ,
  },
 { /* unit 11 (Old: 11) */
  0.0, -13.126020, 9,
  &Sources[9] ,
  &Weights[9] ,
  },
 { /* unit 12 (Old: 12) */
  0.0, -31.796650, 9,
  &Sources[18] ,
  &Weights[18] ,
  },
 { /* unit 13 (Old: 13) */
  0.0, -4.535110, 9,
  &Sources[27] ,
  &Weights[27] ,
  },
 { /* unit 14 (Old: 14) */
  0.0, 1.158620, 9,
  &Sources[36] ,
  &Weights[36] ,
  },
 { /* unit 15 (Old: 15) */
  0.0, -6.204220, 9,
  &Sources[45] ,
  &Weights[45] ,
  },
 { /* unit 16 (Old: 16) */
  0.0, 24.567390, 9,
  &Sources[54] ,
  &Weights[54] ,
  },
 { /* unit 17 (Old: 17) */
  0.0, 15.629430, 9,
  &Sources[63] ,
  &Weights[63] ,
  },
 { /* unit 18 (Old: 18) */
  0.0, 20.441500, 9,
```

```
      &Sources[72] ,
      &Weights[72] ,
      },
    { /* unit 19 (Old: 19) */
      0.0, -4.464750, 9,
      &Sources[81] ,
      &Weights[81] ,
      },
    { /* unit 20 (Old: 20) */
      0.0, -57.996361, 9,
      &Sources[90] ,
      &Weights[90] ,
      },
    { /* unit 21 (Old: 21) */
      0.0, -2.686650, 9,
      &Sources[99] ,
      &Weights[99] ,
      }

  };


int ch13(float *in, float *out, int init)
{
  int member, source;
  float sum;
  enum{OK, Error, Not_Valid};
  pUnit unit;


  /* layer definition section (names & member units) */

  static pUnit Input[9] = {Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units +
7, Units + 8, Units + 9}; /* members */

  static pUnit Hidden1[9] = {Units + 10, Units + 11, Units + 12, Units + 13, Units + 14, Units +
15, Units + 16, Units + 17, Units + 18}; /* members */

  static pUnit Output1[3] = {Units + 19, Units + 20, Units + 21}; /* members */

  static int Output[3] = {19, 20, 21};

  for(member = 0; member < 9; member++) {
    Input[member]->act = in[member];
  }

  for (member = 0; member < 9; member++) {
    unit = Hidden1[member];
    sum = 0.0;
    for (source = 0; source < unit->NoOfSources; source++) {
      sum += unit->sources[source]->act
            * unit->weights[source];
    }
    unit->act = Act_Logistic(sum, unit->Bias);
  };
```

```
  for (member = 0; member < 3; member++) {
    unit = Output1[member];
    sum = 0.0;
    for (source = 0; source < unit->NoOfSources; source++) {
      sum += unit->sources[source]->act
            * unit->weights[source];
    }
    unit->act = Act_Logistic(sum, unit->Bias);
  };

  for(member = 0; member < 3; member++) {
    out[member] = Units[Output[member]].act;
  }

  return(OK);
}
```

TREINAMENTO PROGRAMA BATCHMAN

```
# definition of the version:
version := "v1"

# definition of the topology:
topol := "9-9-3"

# definition of the data sets:
set_tre := "../Data_bck/treina.pat"
set_val := "../Data_bck/valida.pat"
set_tes := "../Data_bck/testa.pat"

# Load the untrained net:
loadNet("RNtrein.net")

# Load data sets used in the training:
loadPattern(set_tre)
loadPattern(set_val)
setPattern(set_tre)

# definition of the process parameters:
val_step := 10
end_cycle := 60000

# definition of the training parameters:
neta := 0.4
mi := 0.2
c := 0.1
dmax := 0.1

# definition of the Learning Function:
setLearnFunc("BackpropMomentum",neta,mi,c,dmax)
```

```
setUpdateFunc("Topological_Order")
setShuffle(TRUE)

# definition the Initializing Function:
setInitFunc("Randomize_Weights",1,-1)

# Shows the initial time, training parameters and functions:

print(" Training version: ", version)
print(" ")
execute("date", dweek, month, dmonth, hour, est, year)
print("  Date: ", month, " ", dmonth," ", year)
print("  Initial training time: ", hour)
print(" ")
print(" Neural Network:")
print("  Topology: 9-", topol,"-2" )
print("  BackpropMomentum")
print("    neta: ",neta, "  mi: ",mi,"  c: ",c," dmax: ",dmax)
print("  Initialization: Randomize_Weights [-1, 1]")
print("  Cycles: ", end_cycle)
print("  Data sets:")
print("    Training:   ",set_tre)
print("    Validation: ",set_val)
print("    Test:       ",set_tes)
print(" Errors during training (MSE) : ")

# Initializes the training and process parameters:
initNet()
saved = FALSE
best_val := 1
best_cycle := 0

# Begins the training:
while CYCLES < end_cycle do
  trainNet()                # Trains the net.
  errorl := MSE             # Computes the error on the training set.
  if CYCLES mod val_step == 0 then  # makes the validation at validation step
          setPattern(set_val)
          testNet()
          errorv:=MSE
          setPattern(set_tre)
          if errorv < best_val then       # If this is the smallest error
               best_cycle := CYCLES            # save the net and cycle
               best_val := errorv
               best_tre := errorl
               saveNet("RNtrein993") # save the net
               saved := TRUE
          endif
  endif

  if CYCLES mod print_step == 0 then  # prints on the print step
          print(" ")
          print("  Cycle = ", CYCLES, "  -  training = ",errorl,"  validation = ",errorv)
  endif
endwhile        #end of training
```

```
# Shows the time of the end of the training:
print(" ")
print("*************************************************************")
execute("date", dweek, month, dmonth, hour, est, year)
print("   Final training time: ", hour)
print("*************************************************************")

# Loads the net and saves the result files:
loadNet("RNtrein993")

# Training
setPattern(set_tre)
testNet()
arq_res_tre := "tre"+version+".res"
saveResult(arq_res_tre,1,PAT,FALSE,TRUE,"create")
delPattern(set_tre)

# validation
setPattern(set_val)
testNet()
arq_res_val := "val"+version+".res"
saveResult(arq_res_val,1,PAT,FALSE,TRUE,"create")
delPattern(set_val)

# test
loadPattern(set_tes)
setPattern(set_tes)
testNet()
best_tes = MSE
arq_res_tes := "tes"+version+".res"
saveResult(arq_res_tes,1,PAT,FALSE,TRUE,"create")
delPattern(set_tes)


# Prints the net performance statistics in each data set:
print("Performance: ")
print("------------")
print(" ")

print(" Training set: ")
command := "analyze -sc -e 402040 -l 0.4 -h 0.6 -i tre"+version+".res | ../analyze.gawk"
execute(command, np, pw, nw, pr, nr, pu, nu, nw1, nr1, nu1, nw2, nr2, nu2, nw3, nr3, nu3)
print("  Number of patterns: ",np)
print("  Wrong:   ",pw," ( ",nw," patterns )")
print("  Right:   ",pr," ( ",nr," patterns )")

print("  Unknown: ",pu," ( ",nu," patterns )")
print(" ")
print("  Class 1: ")
print("   Wrong:   ",nw1," patterns")
print("   Right:   ",nr1," patterns ")
print("   Unknown: ",nu1," patterns ")
print(" ")
print("  Class 2: ")
```

```
print("   Wrong:   ",nw2," patterns")
print("   Right:   ",nr2," patterns ")
print("   Unknown: ",nu2," patterns ")
print(" ")
print("  Class 3: ")
print("   Wrong:   ",nw3," patterns")
print("   Right:   ",nr3," patterns ")
print("   Unknown: ",nu3," patterns ")
print(" ")

print("------------------------------------------------------------------------------ ")
print(" ")

print(" Validation set: ")
command := "analyze -sc -e 402040 -l 0.4 -h 0.6 -i val"+version+".res | ../analyze.gawk"
execute(command, np, pw, nw, pr, nr, pu, nu, nw1, nr1, nu1, nw2, nr2, nu2, nw3, nr3, nu3)
print("  Number of patterns: ",np)
print("  Wrong:   ",pw," ( ",nw," patterns )")
print("  Right:   ",pr," ( ",nr," patterns )")
print("  Unknown: ",pu," ( ",nu," patterns )")
print(" ")
print("  Class 1: ")
print("   Wrong:   ",nw1," patterns")
print("   Right:   ",nr1," patterns ")
print("   Unknown: ",nu1," patterns ")
print(" ")
print("  Class 2: ")
print("   Wrong:   ",nw2," patterns")
print("   Right:   ",nr2," patterns ")
print("   Unknown: ",nu2," patterns ")
print(" ")
print("  Class 3: ")
print("   Wrong:   ",nw3," patterns")
print("   Right:   ",nr3," patterns ")
print("   Unknown: ",nu3," patterns ")
print(" ")

print("------------------------------------------------------------------------------ ")
print(" ")

print(" Test set: ")
command := "analyze -sc -e 402040 -l 0.4 -h 0.6 -i tes"+version+".res | ../analyze.gawk"
print("  Number of patterns: ",np)
print("  Wrong:   ",pw," ( ",nw," patterns )")
print("  Right:   ",pr," ( ",nr," patterns )")
print("  Unknown: ",pu," ( ",nu," patterns )")
print(" ")
print("  Class 1: ")
print("   Wrong:   ",nw1," patterns")
print("   Right:   ",nr1," patterns ")
print("   Unknown: ",nu1," patterns ")
print(" ")
print("  Class 2: ")
print("   Wrong:   ",nw2," patterns")
print("   Right:   ",nr2," patterns ")
```

```
print("   Unknown: ",nu2," patterns ")
print(" ")
print("  Class 3: ")
print("   Wrong:   ",nw3," patterns")
print("   Right:   ",nr3," patterns ")
print("   Unknown: ",nu3," patterns ")
print(" ")


# Deleting result files:
command = "rm *.res"

execute(command)

# Shows the time of end of the process:
print("*************************************************************************
******")
execute("date", one, two, three, hour)
print("Process end time: ", hour)

*************************************************************************
************************
```

ARQUIVO RESULTANTE DO PROGRAMA BATCHMAN ACIMA:

```
 Training version: v1

  Date: Jul 28 1992
  Initial training time: 01:42:08

 Neural Network:
  Topology: 9-9-3
  BackpropMomentum
    neta: 0.4  mi: 0.2  c: 0.1 dmax: 0.1
  Initialization: Randomize_Weights  [-1, 1]
  Cycles: 60000
 Data sets:
  training:   ../Data_bck/trein.pat
  validation: ../Data_bck/valida.pat
  test:       ../Data_bck/testa.pat


Errors during training (MSE) :


  Cycle = 60000  -   training = 0.0424512   validation = 0.184802

*************************************************************************
*
  Final training time: 01:44:07
*************************************************************************
*

Results in the end cycle:
```

```
--------------------------
Cycle = 60000   training = 0.000000   validation = 0.0012547   test = 0.0010572

Performance:
----------------

 Training set:
  Number of patterns: 84
  Wrong:    0 ( 0 patterns )
  Right:    100 ( 84 patterns )
  Unknown.: 0 ( 0 patterns )
  Class 1:
   Wrong:    0 patterns
   Right:    28 patterns
   Unknown.: 0 patterns
  Class 2:
   Wrong:    0 patterns
   Right:    28 patterns
   Unknown.: 0 patterns
 Class 3:
   Wrong:    0 patterns
   Right:    28 patterns
   Unknown.: 0 patterns

-------------------------------------------------------------------------------
 Validation set:
  Number of patterns: 42
  Wrong:    0.36 ( 1 patterns )
  Right:    99.29 ( 41 patterns )
  Unknown.: 0 ( 0 patterns )
  Class 1:
   Wrong:    0 patterns
   Right:    14 patterns
   Unknown.: 0 patterns
  Class 2:
   Wrong:    1 patterns
   Right:    13 patterns
   Unknown.: 0 patterns
 Class 3:
   Wrong:    0 patterns
   Right:    14 patterns
   Unknown.: 0 patterns

-------------------------------------------------------------------------------
 Test set:
  Number of patterns: 42
  Wrong:    0.3 ( 1 patterns )
  Right:    99.29 ( 41 patterns )
  Unknown.: 0 ( 0 patterns )
  Class 1:
   Wrong:    0 patterns
   Right:    13 patterns
   Unknown.: 1 patterns
  Class 2:
   Wrong:    0 patterns
```

Right: 14 patterns
Unknown.: 0 patterns
Class 3:
Wrong: 0 patterns
Right: 14 patterns
Unknown.: 0 patterns

****
*
Process end time: 01:44:31

SNNS pattern definition file V3.2
generated at Sat Apr 20 06:33:38 2002

No. of patterns: 168
No. of input units: 9
No. of output units: 3

0.55761483	0.99226946	0.82823440	0.96521582	0.98757764	0.95106261	0.98611059	0.98078307	0.96182486	1 0 0
0.55830551	0.99226946	0.75805318	0.96401311	0.98136646	0.99666740	0.96380946	0.97905582	0.95236867	0 0 1
0.68976632	0.98542242	0.92531158	0.86720177	0.95652174	0.96019996	0.97700660	0.98873446	0.96154992	1 0 0
0.52169909	0.99414688	0.74461282	0.97905237	0.98757764	0.99908490	0.96347520	0.97674173	0.95245115	0 0 1
0.66812478	0.98652678	0.81424587	0.86055586	0.95031056	0.96880463	0.97524127	0.98850119	0.94826770	1 0 0
0.49441694	0.99558255	0.82799585	0.99179138	0.99378882	0.94236233	0.99034411	0.97456184	0.98908314	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.98926464	0.97033955	0.97379561	0.96647412	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.96604567	0.97904248	0.96438205	0.98770876	0 0 1
0.73397030	0.98321369	0.95458548	0.85518008	0.95031056	0.96351890	0.97353709	0.99140289	0.94984782	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.96085555	0.97951581	0.96120897	0.99936586	0 0 1
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.99181873	0.96915778	0.97325841	0.96147338	0 0 1
0.56187406	0.99204859	0.75676142	0.97081138	0.98757764	0.99978147	0.96240092	0.97715826	0.95481989	0 0 1
0.60089789	0.99006074	0.76616663	0.86688087	0.95652174	0.96118335	0.97427314	0.99483215	0.94335677	0 1 0
0.65546218	0.98718940	0.82040265	0.86668402	0.95031056	0.96289063	0.97696447	0.98821883	0.96460733	1 0 0
0.45032808	0.99779128	0.71016427	0.99539276	1.00000000	0.98658763	0.97070503	0.97277589	0.96128365	0 0 1
0.61402095	0.98939812	0.75961902	0.86756852	0.95652174	0.95144504	0.97640511	0.99906292	0.95208027	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97291576	0.97726975	0.96963195	0.97989351	0 0 1
0.57177392	0.99149641	0.76923487	0.89078134	0.96273292	0.95147236	0.97873904	0.99633364	0.94379186	0 1 0
0.68216876	0.98586416	0.88330030	0.86485973	0.95031056	0.96635981	0.97334375	0.99355184	0.94948054	1 0 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97142701	0.97695524	0.97001989	0.98297753	0 0 1
0.53505238	0.99348426	0.82466114	0.95705574	0.98136646	0.92852655	0.99630461	0.98688499	0.96606209	0 1 0
0.61402095	0.98939812	0.75961902	0.86756852	0.95652174	0.95144504	0.97640511	0.99906292	0.95208027	0 1 0
0.63255439	0.98840420	0.89648688	0.86670020	0.95652174	0.95357572	0.97508776	0.99662607	0.94557153	0 1 0
0.43352135	0.99867477	0.70385268	0.99991775	1.00000000	0.97704054	0.97347362	0.97086765	0.98012338	0 0 1
0.54702429	0.99282165	0.78992836	0.97305231	0.98757764	0.99207823	0.96682004	0.97803408	0.96041528	0 0 1
0.59272476	0.99050248	0.92566434	0.95653933	0.98136646	0.94710173	0.98707600	0.97838089	0.96271062	1 0 0
0.64245424	0.98785202	0.92235247	0.91523193	0.96894410	0.95271525	0.98215004	0.98444167	0.95243063	1 0 0
0.66605272	0.98663722	0.84829597	0.86952629	0.95652174	0.96544471	0.97411639	0.99342136	0.94602103	1 0 0
0.60285484	0.98995030	0.81966668	0.92546305	0.96894410	0.95644395	0.98129976	0.98649257	0.95183542	1 0 0
0.64210890	0.98796245	0.86660897	0.87612231	0.95652174	0.95163625	0.98038356	0.98754853	0.96957481	0 1 0