SIMULAÇÃO PARALELA E DISTRIBUÍDA DE REDES NEURONAIS PARA PERCEPÇÃO VISUAL DE IMAGENS NATURAIS

Rui Chuo Huei Chiou

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO EM ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO

Aprovada por:

Profa. Sueli Bandeira Teixeira Mendes, Ph. D.
(presidente)

Prof. Valmir Carneiro Barbosa, Ph. D.

Prof. Antônio de Oliveira, D. Sc.

Prof. Luis Paulo V. Braga, D. Sc.

RIO DE JANEIRO, RJ – BRASIL
MAIO DE 1991

CHIOU, RUI CHUO HUEI

Simulação Paralela e Distribuída de Redes Neuronais

para Percepção Visual de Imagens Naturais

[Rio de Janeiro] 1991

VI, 102 p., 29.7 cm,

(COPPE/UFRJ,

M. Sc., ENGENHARIA DE SISTEMAS E COMPUTAÇÃO, 1991)

TESE – Universidade Federal do Rio de Janeiro, COPPE

1 – Redes Neuronais

2 – Processamento Paralelo e Distribuído

3 – Processamento de Imagem

4 – Percepção Visual

I. COPPE/UFRJ II. Título(Série).

À Mãe Natureza,
cuja beleza e sabedoria me darão energia e tranqüilidade
para continuar a viver,

A minha avó
Chiou Shuang,
A meus pais
Chiou Jia Hann e Chiou Chen Chu Ing e
A meus irmãos
Helena-J. I. Chiou, Chiou Chen Tu e Chiou Pei Chih

AGRADECIMENTOS

Muitas pessoas contribuiram para a realização deste trabalho, seja pela amizade ou pelas trocas de idéias. Sendo assim, é um prazer expressar por elas meu setimento de agradecimento.

Aos professores Sueli, Valmir e Antônio pela liberdade de expressão, paciência e apoio.

À Claudia pela amizade e revisão do texto.

Ào Maurício, Mariela, Inês, Wamberto, Márcia, Hércules, Mau, Cláudia, Cristina, Elíseo, Ângela, Fernando, Carlos. A sua amizade e seu apoio foram fundamentais para a conlusão dese Trabalho. Obrigado

Às famílias Wu e aos amigos Marco Yenle Yang Wu, Thomas Yenhon Wu, Kuo Ri Pan, Kuo Ton Hau, Kuo Yeh Pwu, Michele Cinn-Yi Chao, Wu Chin Hui e Wu Ih Hui, que me receberam de braços abertos.

Às instituições CNPQ, CAPES, IBM do Brasil, Sociedade Beneficiente Quilherme Quinle, e aos colegas da empresa Globo Computação Gráfica Ltda, em especial ao José Dias pelo apoio e confiança.

À Lucyanna, com muito carinho.

Resumo da Tese apresentada à COPPE como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M. Sc.)

Simulação Paralela e Dsitribuída de Redes Neuronais para
Percepção Visual de Imagens Naturais

Rui Chuo Huei Chiou

Maio de 1991

Orientadores: Sueli B.T. Mendes e Valmir C. Barbosa
Programa: Engenharia de Sistemas e Computação

Na primeira parte da tese, vamos revisar uma rede neuronal de segmentação de borda, chamada filtro CORT-X, proposto por Gail A. Carpenter. O filtro detecta, regulariza e completa bordas de uma imagem ruidosa por meio da interação não linear entre múltiplas escalas espaciais. As células da rede são análogas às células simples, células complexas, células hipercomplexas e células cooperativas orientadas da córtex visual de primatas.

A segunda parte consiste em mostrar as principais idéias da simulação paralela e distribuída do filtro CORT-X, do sistema BCS e do sistema FCS. Os objetivos dessas idéias são: otimizar a memória necessária para computação do filtro, otimizar as comunicações da rede, otimizar a distância de comunicação entre os processadores e a política de escalonamento de processos nos processadores.

A última parte consiste em apresentar os resultados obtidos da implementação do filtro CORT-X em um computador paralelo de memória distribuída formato por uma rede de *Transputers*. O simulador é implementado na linguagem OCCAM para facilitar o processamento paralelo e distribuído, pois a linguagem possui facilidades para comunicação entre os *Transputers*.

Abstract of Thesis presented to COPPE as partial fulfillment of the requirements for the degree of Master of Science (M. Sc.)

Título em inglês

Distributed Parallel Simulation of Neural Networks for

Visual Perception of Natural Images

Rui Chuo Huei Chiou

May, 1991

Thesis Supervisor: Sueli B. T. Mendes e Valmir C. Barbosa

Department: Programa de Engenharia de Sistemas e Computação

In the fist part of this thesis, we are going to review a neural network for boundary segmentation, called the CORT-X filter, proposed by Gail A. Carpenter. The filter detects, regularizes, and sharp completes of a noise image by using nonlinear interactions between multiple spatial scales. The network cells in these filter are analogous to cortical simple cells, complex cells, hypercomplex cells, and unoriented and oriented cooperative cells of primate visual cortex.

The second part explains the main ideas for distributed parallel simulation of CORT-X filter, BCS system and FCS system. The idea objectives are: to optimize the necessary memory for processing, to optimize network communications, to optimize the distance inter-processors and scheduled policy of processes in the processors.

The last part presents the results of CORT-X implementation in the transputers network. The simulator is implementated in OCCAM language to facilite the distributed parallel processing because the language have facilities for transputer comunication.

Índice

Lista de Figuras

Lista de Tabelas

Capítulo I

Introdução

I.1 Aspectos Teóricos da Percepção Visual

Poucas áreas da ciência podem gabar-se da riqueza de fenômenos paradoxais e interessantes de fácil acesso para introspecção, como a percepção visual pode. A figura I.1 mostra um exemplo dessa riqueza. O suporte teórico da simulação é uma teoria de processamento visual em tempo real, que visa a unificação de percepcão tridimencional de forma, de cor e de brilho, incluíndo a percepcão de profundidade, de textura, de superfície e de movimento [13] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30]. A teoria faz parte do programa de pesquisa do *Center For Adaptive Systems of Boston University*

A teoria é baseada em dois tipos de redes neuronais de propósito

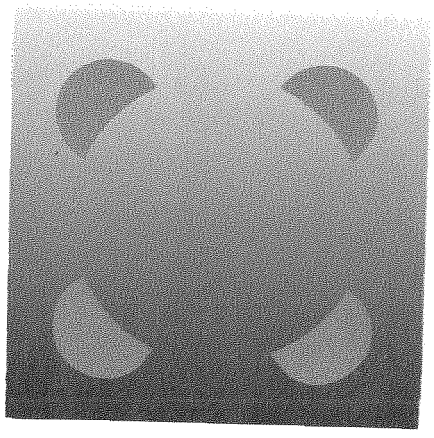


Figura I.1: A figura mostra a percepção ilusória de uma esfera

geral com característica competitiva e cooperativa [12] [11] [14]. Um desses tipos de rede é chamada de rede *FeedForward On-Center-Off-Surround*, por possuir somente caminhos diretos sem realimentação. Apesar de sua aparente simplicidade, as suas células obedecem a lei de ação de massa ou multiplicativa, por isso a rede é capaz resolver alguns problemas de percepcão visual tais como: processamento de refletância, normalização de ativação total, modulação de lei de *Weber*, processamento de nível de adaptação, eliminação de ruído, propriedade de deslocamento, processamento sensível ao raio, invariância à lei de força, sensibilidade à frequência espacial, amplificação energética dos padrões de entradas casadas e eliminação energética dos padrões de entradas descasadas. O outro tipo de rede possui caminhos de realimentação com capacidade de resolver o problema de mudança de contraste, armazenamento de memória de termo curto, multi-estabilidade, histereses, propagação de sensibilidade de refletância e ondas estacionárias sensíveis à freqüência espacial.

Com este mecanismos e suas constelações de propriedades emergentes como tarefas, Stephen Grossberg foi capaz de dedicar-se às variantes da seguinte questão básica: "Como palpites visuais localmente ambíguos são confinados juntos em percepcões sensíveis ao contexto de tal forma que são globalmente não ambíguas ?" A resposta está nas interações sensíveis ao contexto entre as propriedades de profundidade, brilho, cor e forma que são capazes de responder as sequintes questões: Por que a rivalidade binocular e a fusão binocular são dois modos alternados? Por que a fusão pode ocorrer com respeito a uma escala espacial enquanto rivalidade ocarre simultaneamente com respeito a outra escala espacial na mesma região de espaço perceptual? Como a rivalidade inibi a visibilidade das percepcões que seriam visíveis quando vistas monocularmente? Como casamentos binoculares em um número escasso de localizações cénicas pode dar informação não ambígua para as regiões binocularmente ambíguas?

Os mecanismos de percepção visual mencionados consistem de interações monoculares e binoculares entre BCS (Boundary Contour System) e FCS (Feature Contour System), sendo que os sistemas baseam-se em rede *FeedForward On-Center Off-Surround* e *FeedBack On-Center-Off-Surround*, respectivamente. As interações paralelas entre os dois sistemas são capazes de manipular a forma e a cor

de uma imagem. O sistema BCS detecta, afina e completa bordas e o sistema FCS gera os sinais de cor e brilho que ativa a difusão de características dentro destas bordas.

A análise desses sistemas leva a várias conclusões revolucionárias que não seguem os conceitos da geometria euclidiana, tais como: Todas as bordas são invisíveis. Todos os extremos de linha são ilusórios. As bordas são as descontinuidades. Tais conclusões oriundas da análise de percepção visual das seguintes questões: Como nós reconhecemos grupos emergentes sem necessariamente ver contrastes que corresponde a estes grupos? Como bordas podem ser formadas pre-atentivamente, e ainda ser, influenciadas pela atenção e informações aprendidas? Como características locais podem iniciar a organização de uma percepcão, ainda que, sejam controladas demais pelas propriedades da configuração global que determinam a percepcão final? Como estágios primários na formação de borda podem ser sensíveis aos contrastes locais da imagem, mesmo que, a configuração de borda final possua propriedades de estrutura e coerência que pode persistir em descartar mudanças significativas nos constrastes locais da imagem ?

Para entender tais questões, o próprio sistema visual possui exemplos excelentes de como subsistemas neuronais individuais, pela necessidade de serem especializados para tratar parte de uma problema adaptativo, não têm informações sobre o problema como um todo. Mesmo assim as interações entre estes subsistemas são tão habilmente projetados que os sistemas como um todo podem sintetizar uma solução globalmente consistente do problema. Estas interações são oriundas da confrontação dos vários problemas visuais chamado de *Boundary-Feature Trade-Off*. O estudo desta confrontação revela vários princípios básicos de incertezas que limitam as informações, pelas quais, os estágios particulares de processamento visual podem computar. Por isso, este sistemas visuais não resistem a estas incertezas. No entanto, os estágios posteriores são projetados para resolvê-los. Estes princípios de incertezas serão detalhados e usados para justificar a necessidade de processamento paralelo e distribuído no capítulo III.

I.2 Delineamento das Principais Contribuições da Tese

Nesta tese, só vamos rever o modelo CORT-X [6] , uma versão modificada do modelo BCS, no capítulo II. A escolha da CORT-X é motivada pela simplificação das equações diferenciais que regem seus neurônios, com isso, teremos uma apresentação dos resultados de forma mais prática para análise de desempenho do modelo para processamento de imagem natural. Neste capítulo devemos explicar as idéias que regem o compromisso entre a detecção de borda, a eliminação de ruído e o complemento de borda. Para isso, os níveis intermediários serão detalhados no sentido de entender os princípios funcionais de máscara orientada, competição espacial, competição de orientações, interação de múltiplas escalas espaciais e cooperação de orientações.

As contribuições principais da tese são as idéias da simulação paralela e distribuída apresentadas no capítulo III. Este capítulo deverá discutir as principais idéias da simulação paralela e distribuída do filtro CORT-X, BCS e FCS. A primeira idéia apresentada é a própria justificativa da escolha do processamento paralelo e distribuído, só assim podemos explorar as peculiaridades das redes neuronais no sentido de usar as facilidades do ambiente paralelo e distribuído de processamento. As peculiaridades vão permitir a escolha de topologia hipercubo retangular, possibilitando assim, a otimização de comunicação entre os processadores através da compensação do tempos de processamento dos grupos de neurônios situados nas zonas internas com os grupos de neurônios situados nas zonas de fronteira, pois os primeiros grupos não fazem comunicação com os processadores vizinhos. Outra vantagem das peculiaridades é a redução de distância de comunicação entre os processadores para dois, a redução permite processamento altamente paralelo. Devemos ainda, discutir métodos de sincronização e gerenciamento de memória.

No capítulo IV, deverá prover meios de integração as idéias da simulação paralela e distribuída com os recursos disponíveis para implementação. O ambiente de implementação, tais como *hardware*(Harlequin,T800, i860 e NCPI) e sofware (a linguagem OCCAM) serão descritos sucintamente. A integração do am-

biente de implementação com a filosofia da simulação mencionada consiste no mapeamento das idéias da simulação nos recursos disponíveis, ou seja, prover mecanismos pelas quais a comunicação entre os processadores, sincronização, escalonamento de processos, gerenciamento de memória são implementados. Com os contrutores ALT, SEQ E PAR da linguagem OCCAM, implementamos os mecanismos essenciais que permitem os controles dos *buffers*, o processamento paralelo dos processos neurônios e a comunicação entre os processos via canais de comunicações. O uso de buffer tem a dupla funções de evitar *deadlock* e de facilitar tráfegos de mensagens não pertencentes ao processador.

Quanto ao resultados obtidos pelo simulador, apresentaremos no capítulo V dois tipos de resultados, o primeiro tipo tenta mostrar a qualidade do filtro CORT-X quanto ao compromisso de detecção de borda, de eliminação de ruído e de complemento de borda. Para isso, os resultados intermediários de cada nível do filtro CORT-X serão apresentados através de fotos e os ajuste de parâmetros das equações serão feitos para aproveitar melhor as características deste filtro. O segundo tipo de resultado procura avaliar o desempenho do simulador. Esta avaliação deve ressaltar a eficiência de comunicação entre os processadores para averiguar o recurso computacional necessário para o processamento em tempo real. Deverá ser avaliado também, o desempenho do simulador quanto ao número de processadores usados e a influência do processamento de neurônios situados na zona interna. Para finalizar, faremos uma idealização número ilimitado de *Transputers* disponível para uso e daremos alguns aspectos da migração do Simulador para o processador Intel 80860[17] [31].

Capítulo II

O filtro CORT-X

II.1 Introdução

O filtro CORT-X é um pre-processador que detecta, regulariza, e completa bordas de uma imagem ruidosa. Além disso, ele também elimina os ruídos existentes da imagem. O CORT-X é uma versão simplificada do sistema BCS; a modificação do BCS para CORT-X é motivada pela necessidade de se obter um modelo mais versátil em termos computacionais, para a etapa de reconhecimento de padrão pela ART(Adaptive Ressonance Theory) [2] [3] [4] [7]. A principal modificação é a eliminação de ciclo de realimentação não linear, sendo assim o CORT-X tem uma capacidade mais limitada para a percepcão de bordas ilusórias [20]. Mas, por outro lado, ele ganha capacidade de interação de múltiplas escalas espaciais que facilita o complemento de bordas e tempo computacional para possibilitar o processamento em tempo real, facilitando assim a aplicação prática, bem como o reconhecimento de padrão pelas ART-2 e ART-3.

Os multi-níveis hierarquizados do CORT-X possuem funções análogas às células simples, complexas e hipercomplexas do córtex visual do cérebro de primatas. A função de cada um desses níveis será descrita nas próximas seções deste capítulo.

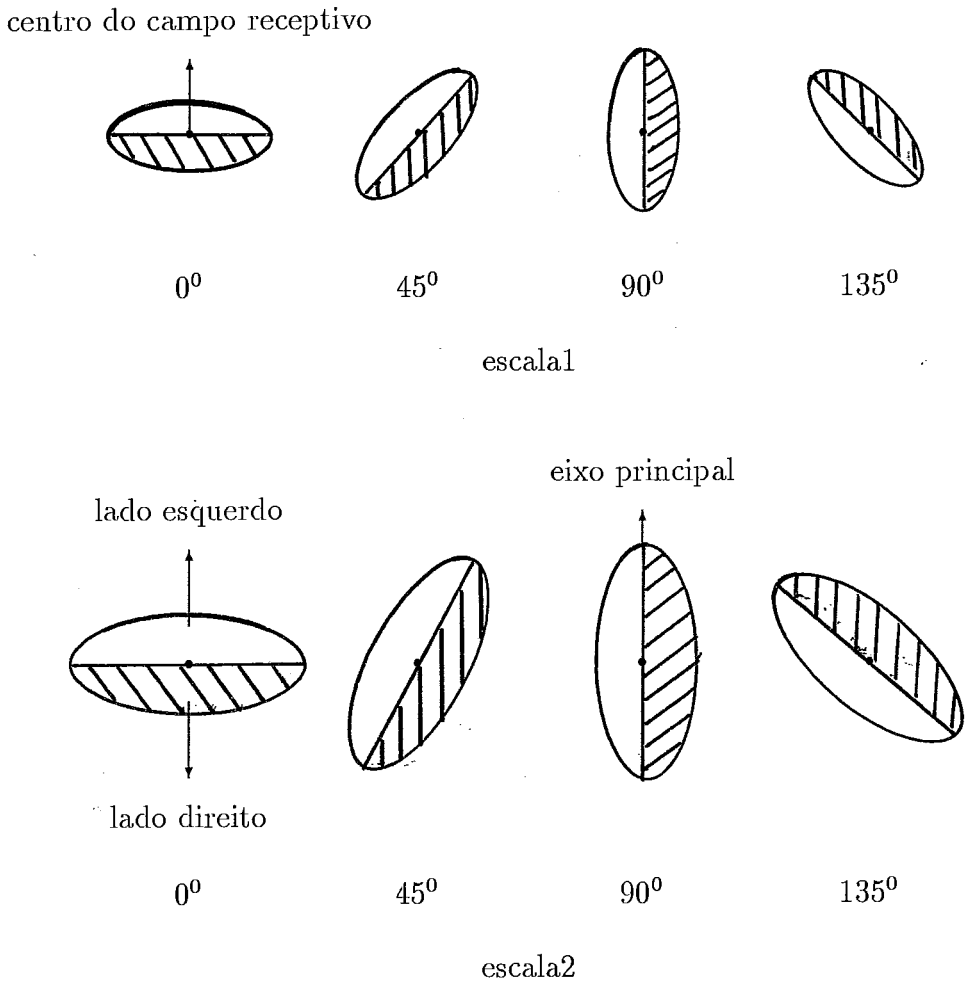


Figura II.1: A figura mostra as possíveis orientações do campo receptivo das células simples com suas respectivas escalas espaciais

II.2 Nível 1: Detector Orientado de Contraste Sensível à Direção de Contraste

Tipo de Célula do Nível: Célula simples da córtex visual

Função Principal: Detector orientado sensível à orientação intensidade, direção e escala espacial

O primeiro nível é um detector de contraste orientado, que é sensível à orientação, intensidade, direção e escala espacial de uma determinada localização da imagem. Este tipo de detector pode ser comparado às células simples do córtex visual.

As características do detector podem ser observadas pela figura II.1, através dos formatos dos campos receptivos das células simples. Percebe-se que a sensibilidade à orientação é obtida pela variação de inclinação do eixo maior do campo receptivo com o eixo horizontal, porque as áreas cobertas pela campo receptivo na imagem é maior nas regiões próximas ao eixo maior. Sendo assim, as bordas da imagem com inclinação próximas ao eixo maior do campo receptivo serão detectadas pela célula simples.

A sensibilidade à intensidade de contraste da imagem é facilmente obtida. Para tanto, basta dividir o campo receptivo em duas partes pelo eixo maior; uma das partes é o lado esquerdo $L_s(x,k)$, enquanto outra parte é o lado direito $R_s(x,k)$. Logo, a diferença entre as duas partes do campo receptivo é a intensidade de contraste da imagem. Para obter a direção de contraste da imagem, basta colocar o peso α_s maior em uma das partes do campo receptivo, por exemplo no lado direito; assim a célula simples é ativada somente se o lado esquerdo do campo receptivo possui maior ativação. A sensibilidade à escala espacial é obtida pelo aumento ou diminuição do campo receptivo. Ainda neste capítulo, se dará uma atenção especial a razão pela qual múltiplas escalas espaciais podem influenciar no desempenho do filtro CORT-X.

A saída da célula simples pode ser modelada pela equação II.1:

$$M_s = Positivo[L_s(x,k) - \alpha_s R_s(x,k) - \beta_s] \tag{II.1}$$

$$L_s(x,k) = \frac{\int_{metadeesquerda} I(y)dy}{\int_{metadeesquerda} dy} \tag{II.2}$$

$$R_s(x,k) = \frac{\int_{metadedireita} I(y)dy}{\int_{metadedireita} dy} \tag{II.3}$$

$$Positivo = \begin{cases} x & \text{se } x > 0 \\ 0 & \text{caso contrário} \end{cases} \tag{II.4}$$

onde,

- $L_s(x,k)$ e $R_s(x,k)$ são as partes esquerda e direita do campo receptivo. Esse campo possui formato de um elipsóide e as ativações das duas partes do campo receptivo obedecem as equações II.2 e II.3;
- x é a posição do centro de campo receptivo;

- y é a posição qualquer do campo receptivo;
- k é a orientação do campo receptivo, na simulação; os valores de k são 0^0, 45^0, 90^0 e 135^0;
- s indica a escala do campo receptivo, podendo variar de 0 a 1 com indicativo de campo receptivo menor e maior, respectivamente;
- α_s é o parâmetro que indica o grau de sensibilidade do campo receptivo, $\alpha > 1$ dá à equação a capacidade de detectar a direção da mudança de contraste, além disso, quando α_s aproxima, o campo receptivo torna mais sensível a pequena mudança de contraste;
- β_x é o parâmetro limiar que indica o mínimo de mudança de contraste aceitável para ativar a célula, tornando-a parte da borda;
- é intensidade da imagem de entrada na posição y.
- $\int_{metadeesquerda} i(y)dy / \int_{metadeesquerda} dy$ é normalizada pela fração, sendo que o numerador é a integral definida na região esquerda do campo receptivo e o denominador eá ntegral unitária definida na região citada.
- $L_s(x,k)$ e $R_s(x,k)$ são as partes esquerda e direita do campo receptivo. Esse campo possui formato de um elipsóide e as ativações das duas partes do campo receptivo obedecem as equações II.2 e II.3;
- $Positivo()$ é uma função que retorna um valor não negativo do parâmetro de entrada.

A diferença entre as partes desse campo receptivo indica a presença ou não de uma borda do centro deste campo.

A célula simples é capaz de responder a vários tipos de contraste, tais como: sensibilidade a passo de iluminação, gradiente de densidade discreta e gradiente de densidade contínua.

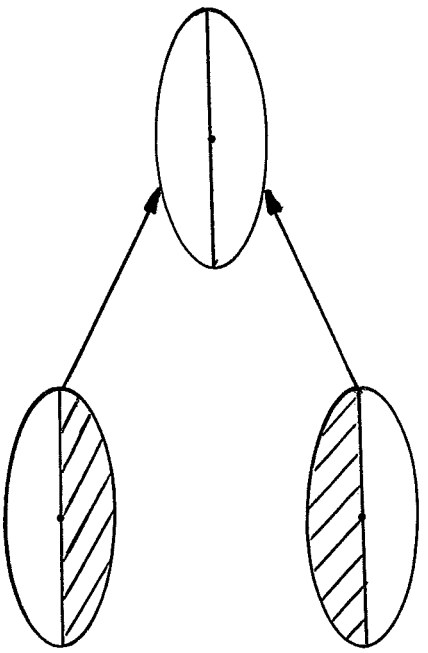


Figura II.2: Célula Complexa : insensível à direção de contraste

II.3 Nível 2: Detector Orientado de Contraste Insensível à Direção de Contraste

Tipo de Célula: Célula complexa da córtex visual

Função Principal: Anular a sensibilidade à direção de contraste da célula simples

A célula complexa é um detector sensível à orientação, à intensidade e à escala espacial de uma dada posição local da imagem. Ao contrário da célula simples, a célula complexa não é sensível à direção de contraste.

A denominação $C_s(x, k)$ à célula complexa é semelhante à célula simples, onde x é localização da célula no plano da imagem, k é a sensibilidade de orientação da célula e s é a sensibilidade à escala da célula.

A característica principal de insensibilidade à direção de contraste

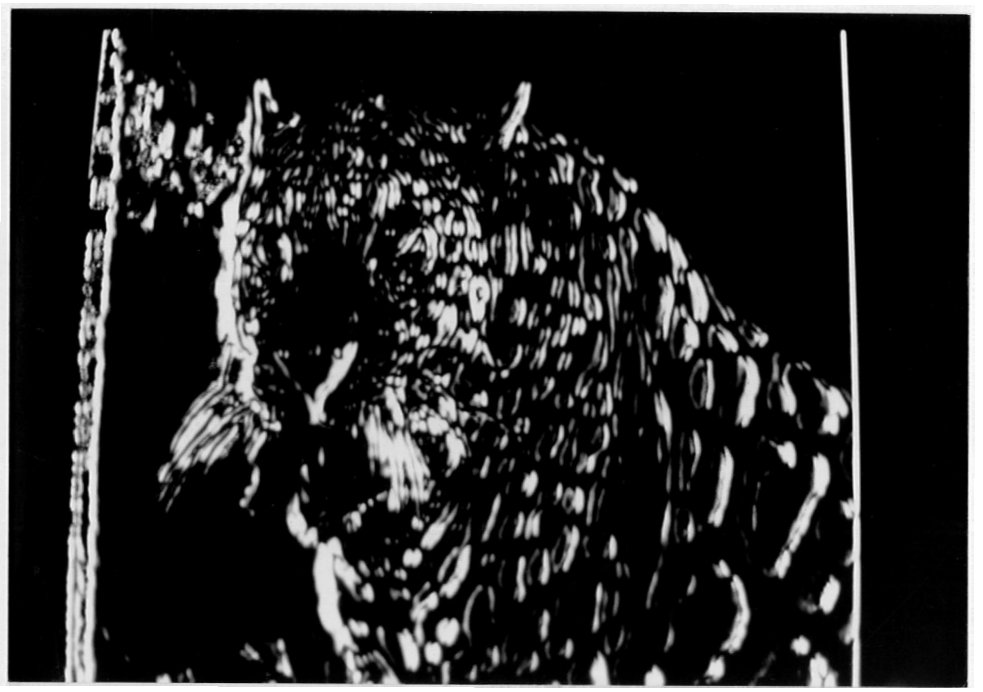


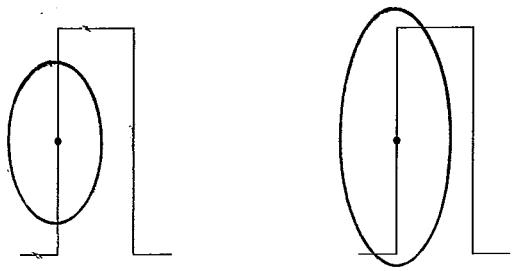
Figura II.3: Respostas das ativações das células complexas.

pode ser facilmente obtida através da soma das duplas células simples com sensibilidade aos sentidos opostos de contrastes. A característica da célula complexa pode ser observada na figura II.2 e é matematicamente descrita pela equação II.5. Percebe-se na equação, se um termo da equação não é nulo implica o outro termo seja nulo ou ambos os termos são nulos. Em termos conceitual, se houver uma diferença de contrastes entre os dois lados do campo receptivo então ela será detectada pela célula complexa independemente da direção das contrastes.

$$C_s(x,k) = Positivo[L_s(x,k) - \alpha_s R_s(x,k) - \beta_s] + Positivo[R_s(x,k) - \alpha_s L_s(x,k) - \beta_s] \quad \text{(II.5)}$$

II.4 Compromisso entre Localização de Borda, Eliminação de Ruído e Complemento de Borda

A idéia principal da CORT-X consiste em explorar as múltiplas escalas espaciais para obter uma boa localização de borda com curvatura acentuada, eliminação de



escala pequena (a)

escala grande (b)

Figura II.4: Influências das múltiplas escalas espaciais

ruído e completar as bordas rompidas pelo ruído. Para isso, precisa-se saber as características do filtro em relação à variação do tamanho do campo de percepção, ou seja, a variação de escala.

Como se pode perceber na figura II.3, a escala pequena pode detectar com mais precisão as curvas acentuadas nas bordas do que a escala grande. Por outro lado, a escala grande pode eliminar ruído com mais eficiência e permite a complemento de borda, pois é mais insensível. Na figura II.7, pode-se observar as tendências das duas escala espaciais. A explicação para boa localização de borda de filtro pequeno é devido ao fato de que ele estima a localização de borda baseadô em contrastes mais locais, é portanto a variação muito acentuada da borda num região pequena pode ser detectada pelo filtro. (veja na figura II.4a). Percebe-se na figura II.4a, que a curva acentuada é detectada pelo filtro pequeno, já o filtro grande não conseque detectar a mesma curva acentuada, porque a curva não conseque ativar o filtro grande pelo fato de que o campo receptivo é maior, tornando insignificantes os contrastes da curva acentuada (veja na figura II.4b).

Quanto à eliminação de ruído, o filtro grande pode fazer melhor essa tarefa do que filtro pequeno, porque o filtro pequeno pode mais facilmente ser ativado pela flutuações locais na distribuição espacial dos *pixels* ruidosos.

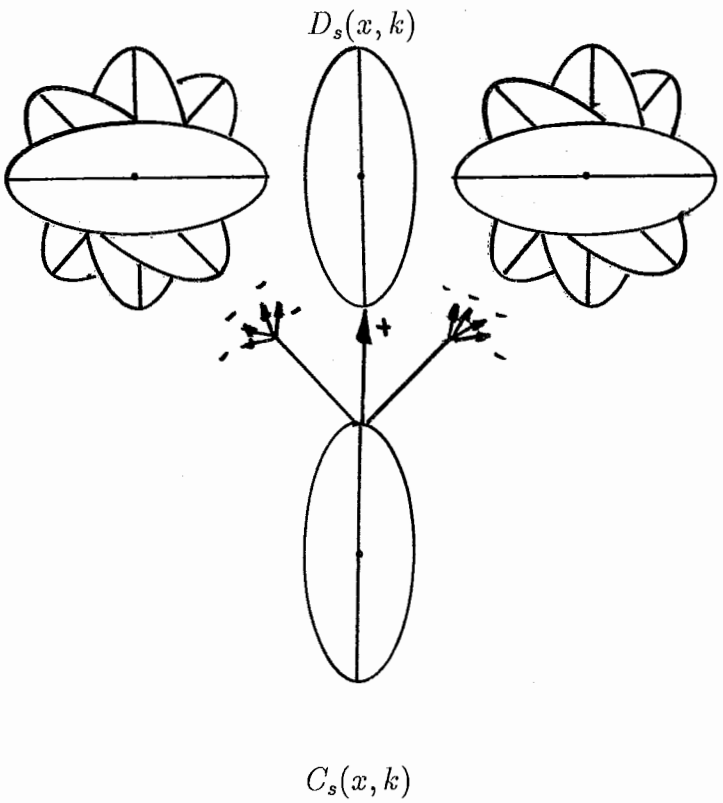


Figura II.5: Rede *Feedforward On-Center-Off-Surround*

II.5 Nível 3: Primeiro Estágio Competitivo

Tipo de Célula do Nível: Célula hipercomplexa da córtex visual

Função Principal: Eliminação de ruídos perto das bordas através da competição espacial

Como foi discutido anteriormente, o filtro de larga escala pode eliminar ruídos afastados da borda da imagem. No entanto, devido a sua grande incerteza posicional, ele não pode eliminar com eficiência ruídos pertos das bordas. Uma maneira de resolver esse problema é utilizar uma competição espacial, ou seja, células complexas de alta atividade nas bordas podem eliminar as células complexas vizinhas com baixas atividades, que podem ser possíveis ruídos. O mecanismo utilizado é uma variante do primeiro estágio competitivo do BCS.

O primeiro estágio competitivo é uma rede *on-center-off-surround*

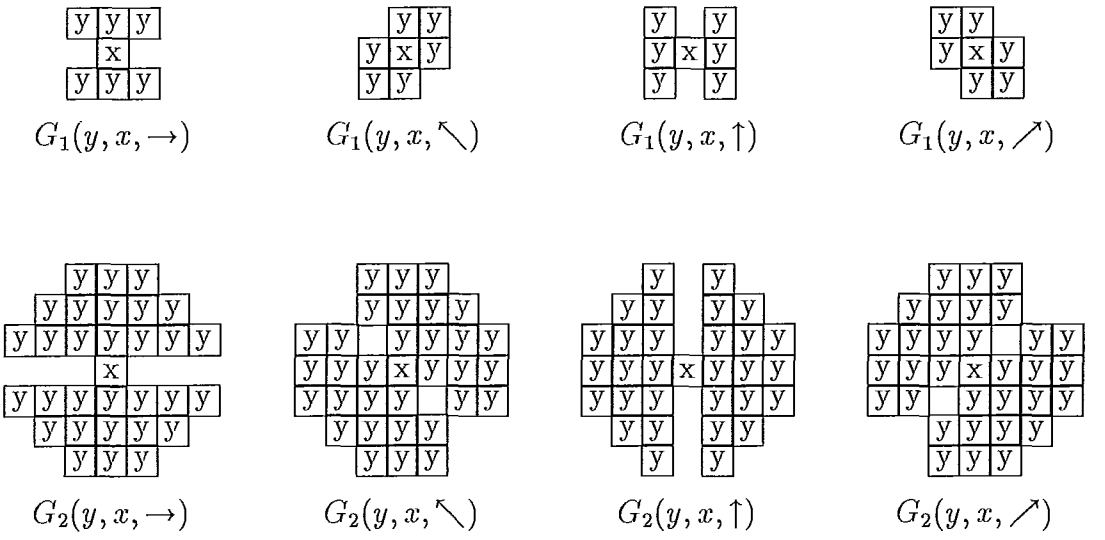


Figura II.6: Núcleo de Competição Espacial

unidirecional(veja na figura II.5). Em função disso, o nível competição espacial recebe somente entradas do nível máscara orientada. Para entender melhor, vamos mostrar como as células dos dois níveis são conectadas entre sí. Uma dada célula complexa $C_s(x,k)$, no nível anterior, excita a célula hipercomplexa $D_s(x,k)$ de mesma posição x, orientação k e escala espacial s no nível competição espacial. A mesma célula no nível máscara orientada inibe as células $D_s(y,m)$, de posições y no nível competitivo espacial, situadas na vizinhança da posição x. E também são inibidas todas as células de orientação m nas posições y (veja a figura II.6).

A saída da célula competitiva na posição x, orientação k, e escala s é definida pela equação II.6,

$$D_s(x,k) = C(x,k)/(1 + \gamma_s \sum_m \sum_y (y,m) G_s(y,x,k)), \qquad \text{(II.6)}$$

onde os núcleos competitivos $G_s(x,y,k)$, definidos pela figura II.6, são funções normalizadas tal que $\sum_y G_s(y,c,k) = 1$. O parâmetro γ_s especifica a força da competição. Na figura II.6, x denota centro do núcleo e y denota os *pixels* que são inibidos. A forma do núcleo é definida pelo ciclo de diâmetro de 3 ou 6 *pixels* ao redor do centro do núcleo. Todos os *pixels* que se interceptam no interior do ciclo e que não estão situados no eixo de orientação do núcleo são inibidos.

II.6 Nível 4: Segundo Estágio Competitivo

Tipo de Célula do Nível: Célula hipercomplexa da córtex visual

Função Principal: Competição orientacional

A competição de orientação utiliza a política do "vencedor leva tudo", em substituição da política "puxa-empurra" (yan-yin) utilizado no modelo BCS. A "política do vencedor leva tudo" escolhe, dentro das orientações possíveis, aquela orientação com máxima ativação como sendo a mais provável orientação da borda da imagem.

O segundo estágio competitivo recebe ativações das células do primeiro estágio. Sendo assim, uma célula nesse estágio, em uma dada escala s e posição x, recebe várias células de orientações diferentes do estágio anterior. Usando a competição orientacional, a célula pode detectar a orientação mais próxima à borda da imagem. A ativação da célula de competição orientacional é descrita pela equação:

$$D_s(x) = D_2(k) = \max_k D_s(x, k), \tag{II.7}$$

onde k denota a orientação vencedora da competição orientacional.

Neste estágio há uma redução considerável de células, um vez que as saídas das células não são mais sensíveis a várias orientações como acontece nos primeiros estágios.

II.7 Nível 5: Interação de Múltiplas Escalas Espaciais

Função Principal: Localização de borda e eliminação de ruído

Nesse nível, vamos explorar as características das múltiplas escalas espaciais. Para isso, precisa-se saber quais são as propriedades dos campos receptivos de tamanhos diferentes. Só assim, pode-se selecionar as propriedades desejáveis e eliminar as propriedades indesejáveis.

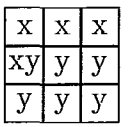
	escala pequena	escala grande	interação das escalas
localização de borda	sim	não em borda de curvatura acentuada	sim
eliminação de ruído	não	sim	sim
completude de borda	não em segmentos eliminados pelos ruídos	sim	sim

Figura II.7: Descrição das propriedades funcionais das escalas funcionais e suas ações combinadas

As propriedades estão esquematizadas na figura II.7. De acordo com a figura, o filtro pequeno desempenha melhor a tarefa de localização de borda e o filtro grande faz uma boa tarefa de eliminação de ruído e complemento de borda. Para obter as qualidades dos dois tipos diferentes de filtros, é necessário uma unificação das qualidades dos dois filtros, ou seja, uma integração em múltiplas escalas para tornar o filtro mais eficiente. Mas, como os dois filtros podem interagir entre si?

Uma solução seria o produto $D_1(x)D_2(x)$, já que D_1 localiza precisamente os segmentos de bordas e elimina ruído perto da borda. Sob outro ponto de vista, $D_1(x)$ geralmente é positivo na borda e zero perto da borda; e para completar, o fator $D_2(x)$ elimina ruído longe da borda. No entanto o produto $D_1(x)D_2(x)$ não resolve a interação das duas escalas, porque $D_2(x)$ faz uma pobre localização de segmento de alta curvatura da borda; assim, $D_2(x)$ pode ser igual a zero nas bordas, cancelando assim a boa localização de borda feita pelo $D_1(x)$.

Este problema pode ser contornado: basta fazer uma difusão espacial de influência de $D_2(x)$ sobre $D_1(x)$ através de um núcleo $U(y,x)$II.8, onde y são os vizinhos mais próximos de x. Desta forma, o máximo de erro possível tratado pelo



$U(y,x)$

Figura II.8: Núcleo de Interação de Múltiplas Escalas

$D_2(y)$ na localização de borda cresce com tamanho da sua escala. Se tamanho do núcleo $U(y,x)$ interage em escala com tamanho de $D_2(y)$, então $\sum_y D_2(y)U(y,x)$ será não nulo quando houver alguma célula $D_2(y)$ dentro do núcleo não nulo. Com isso, a equação $D_1 \sum_y D_2(y)U(y,x)$ pode restaurar a perda da localização de borda, já D_1 é muito provável seja não nulo, resolvendo finalmente o problema. Para normalizar o núcleo de vizinhos mais próximos, usa-se a soma $\sum_y U(y,x) = 1$. A ativação da célula de interação múltiplas escalas é definida pela equação II.8.

$$B_{12}(x) = D_1(x) \sum_y D_2(y)U(y,x). \tag{II.8}$$

II.8 Nível 6: Cooperação Orientada à Longa Distância

Tipo de Célula do Nível: Célula bipolar

Função Principal: Complemento de bordas eliminadas pelos ruídos

O objetivo desse nível é recuperar os *pixels* das bordas eliminadas pelo ruídos. Uma maneira possível de recuperação pode ser encontrada nas células bipolares da córtex visual com função de complemento de borda. As células bipolares são modelados pelo CC Loop do BCS(Bounday Contorn System) como células cooperativas. Só que, neste estágio de processamento, a complemento de borda usa uma versão *feedforward* (sem realimentação) do CC Loop.

A biporidade da célula cooperativa é justificada pelo fato de que a célula só é ativada se ambos os polos, ou lados, forem suficientemente ativados. A cooperação bipolar desempenha o papel de complemento de borda da seguinte forma: se ambos os polos do campo receptivo da célula cooperativa forem suficientemente ativados, isto significa que o centro do campo receptivo pertence à borda, então a célula deve ser ativada de acordo com células vizinhas no campo receptivo. Desta forma, se a ativação da célula neste estágio for modificada pelo ruído, as células vizinhas do campo receptivo no estágio anterior darão informações para auto-recuperação de borda eliminada pelos ruídos.

No processo de complemento de borda, explora-se o detector de escala grande, pois ele é mais insensível ao ruído, e as bordas detectadas são mais consistentes do que o filtro de escala pequena. Apesar do filtro da escala grande apresentar maior incerteza espacial, a consistência das bordas detectadas dá o suporte às células cooperativas na recuperação da parte da borda eliminada pelos ruídos .

A ativação da célula cooperativa bipolar é definida pela equação:

$$B_2(x) = D_2(x)Positivo(\sum_y D_2(y,k)O(y,x,k) - \delta), \qquad \text{(II.9)}$$

onde δ é uma limiar de cooperação, e k é a orientação correspondente à ativação máxima $D_2(x,k)$ definida na II.9. O núcleo de cooperação orientado $O(y,x,k)$, definido na figura II.9, é uma função normalizada tal que $\sum_y O(y,x,k) = 1$. A função $B_2(x)$ na equação II.9 depende da resposta do detector grande $D_2(x)$, que pode ser ruidoso. Por isso, se a cooperação $\sum_y D_2(y,k)o(y,x,k)$ for menor que limiar δ, então $D_2(x)$ é ruído, e consequentemente será eliminado pelo produto da equação II.9. Desta forma $B_2(x)$ é ativado somente se prover suporte cooperativo suficiente.

II.9 Nível 7: O Filtro CORT-X

No estágio de interação, o objetivo de localização de borda e atenuação de ruído foi almejado. O mesmo acontece com complemento de borda do estágio de cooperação. A soma das ativações das células de posicões correspondente às dos dois estágios

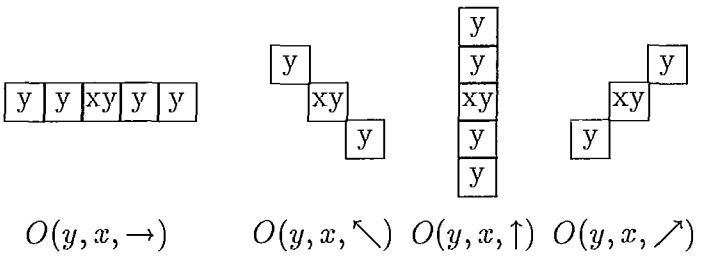


Figura II.9: Núcleo de Cooperação Orientada

mencionados para o filtro CORT-X, pode-se obter uma unificação de suas respectivas qualidades, ou seja, pode-se obter um filtro capaz de localização precisa de borda, eliminação de ruído e completar partes das bordas eliminadas pelo ruído. A equação da soma é dada da seguinte forma:

$$B(x) = B_{12}(x) + B_2(x). \tag{II.10}$$

Finalmente com este nível do filtro CORT-X, o compromisso entre localização de borda, eliminação de ruído e complemento de borda foi discutido intensivamente. Os resultados obtidos serão abordados e avaliados em um dos capítulos posteriores.

Capítulo III

Simulação Paralela e Distribuída de Filtro CORT-X, Sistema BCS e Sistema FCS

III.1 Introdução

Neste capítulo vamos discutir as principais idéias da simulação do filtro CORT-X,BCS (redes neuronais do tipo *forward on-center-off-surround*) e FCS (*recurrent on-center-off-surround*) em um ambiente paralelo e distribuído de processamento. Para isso, os aspectos de política de mapeamento explorando as peculiaridades do filtro, escolha de topologia hipercubo retangular, a otimização de comunicações entre processadores, sincronização de níveis das redes neuronais, escalonamento de processos nos processadores e gerenciamento de memória mais adequado para as redes neuronais mencionadas serão abordados com mais detalhes.

A seção III.2 procura justificar a escolha de processamento paralelo e distribuído ao invés do processamento serial e paralelo. Ela deverá argumentar a importância de comunicação dos neurônios para o entendimento conceitual das características funcionais dos sistemas BCS e FCS. Esta importância de comunicação justificará o ambiente paralelo e distribuído de processamento.

Para colocar a justificativa de uma forma mais prática, a seção III.3 apresentará algumas característica peculiares das redes neuronais CORT-X, BCS e FCS com objetivo de identificar aglomerações de comunicações entre os neurônios.

Permitindo assim, um estudo da viabilidade do processamento paralelo e distribuído de tal forma que explora as vantagem das características peculiares.

O estudo deverá facilitar a escolha de uma topologia de multiprocessamento mais adequada à topologia do CORT-X. Sempre tomando cuidado com a otimização de comunicações, a distância entre processadores a ser percorrido pela mensagem, a forma de sincronização e o gerenciamento de memória.

A topologia hipercubo retangular será escolhido. Citamos agora, alguns méritos desta escolha: a distância máxima de comunicação entre processadores é duas e o agrupamento de processos neurônios situados regiões de fronteira e regiões internas. Os neurônios das regiões intenas não se comunicam com processos neurônios vizinhos situados nos outros processadores, por esta razão, eles podem ser simulados sem a necessidade de usar comunicação lenta entre os processadores. Evitando assim, *overhead* de comunicação, ou melhor, otimização de comunicação entre os processadores.

A sincronização depende da forma de implementação de comunicações entre os neurônios. É claro conciliando com o recurso computacional disponível, por exemplo, a quantidade de memória disponível será uma fator crucial para escolha do tipo de sincronização e a forma de otimização de memória reservada para as áreas de trabalhos. Sempre levando em consideração, o compromisso entre a memória disponível e o grau de paralelismo.

III.2 Justificativa do Processamento Paralelo e Distribuído

A essência da tese é mostrar a viabilidade de um simulador paralelo e distribuído para as redes CORT-X, BCS e FCS [13] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30].

A primeira argumentação é justificar a escolha de processamento paralelo e distribuído ao invés da tradicional processamento serial e paralelo. A justificativa é oriunda das próprias características funcionais dos sistemas BCS e FCS,

logo, precisa-se entender como essas características contribuem para o desempenho de um ambiente distribuído de processamento.

Os sistemas BCS e FCS tornam-se possíveis graças à compreensão do conceito de especialização. A especialização se opõe à hipótese popular de módulos independentes, porque há interações entre os módulos que não podem simplesmente ser separados, pois os significados funcionais destes módulos não são captados pelo conceito de módulos independentes. Os conceitos funcionais destes sistemas baseia-se na Resolução Hierárquica de Incerteza, ou seja, o entendimento do sistema visual deriva do fato de que quando um estágio elimina um tipo de incerteza, geralmente gera um novo tipo de incerteza. O leitor deve está perguntando: Como eliminar a incerteza?

Revendo a teoria do sistema BCS e FCS [21], sabe-se que a necessidade de reduzir o efeito de variação de iluminação leva à necessidade de preenchimento de característica. A necessidade de preenchimento de característica leva à necessidade de sintetizar bordas capazes de restringir o preenchimento de características aos domínios perceptuais apropriados. A necessidade de sintetizar bordas leva à necessidade de sintetizar ao campos receptivos sensíveis a orientações. Tais campos receptivos são, porém, incapazes de restringir o preenchimento de características nos extremos das linhas ou cantos agudos. Então a certeza orientacional implica um tipo de incerteza posicional, que é inaceitável da perspectiva dos requisitos do preenchimento de características.

Aparentemente parece não haver solução. Felizmente, a incerteza posicional pode ser resolvida pela competição 'puxa-empurra" entre os campos receptivos sensíveis às orientações perpendiculares. Percebe-se que esta solução não pode ser obtida sem levar em consideração o conceito funcional sob a ótica da especialização, e não sob aquela de módulos independentes, dado que a descoberta da solução da incerteza posicional foi possível, graças ao requisito do prenchimento de características sobre sintetizador de bordas, ou seja, a existência de campos receptivos sensíveis a orientações, em particular, orientações perpendiculares. Portanto BCS (sintetizador de bordas) e FCS (preenchimento de características) não podem ser vistos como módulos independentes devido às interações funcionais entre os dois

sistemas.

Como as interações entre BCS e FCS são feitas via comunicações intensas entre os neurônios dentro dos dois sistemas, portanto necessitamos de mecanismos que facilitam essas comunicações. Por isso, a escolha do processamento paralelo e distribuído.

Para acrescentar, o sistema FCS é uma rede recorrente, ou seja, há interações mútuas entre os neurônios vizinhos que vão se propagando sobre toda a rede. A idéia de propagação pode ser melhor entendida pelo seguinte exemplo. Suponha que uma gota d'água cai numa lagoa, certamente haverá perturbação na superfície da lagoa que será propagada em toda a lagoa até as margens. Acrescentando, outra gota d'água no mesmo instante da outra gota d'água, as ondas propagadas certamente vão se encontrar em algum lugar no lago. Os locais do encontro possuim influências dos efeitos das perturbações das gotas d'águas. Entretanto, se as gotas d'água têm cores vermelho e verde, então a lagoa será colorida de acordo com a propagação dos pigmentos das cores (supondo mistura semi-homogênea), ou seja, a propagação da cores basea-se no decaimento Gaussiana em funcão da distância. A situação pode se tornar caótica se ocorrer uma chuva, onde cada gota d'água possui um pigmento colorido aleatoriamente. E para piorar a situação, há obstáculos na lagoa que impedem propagações de cores. Imagine agora, qual será a coloração da lagoa?

O processo de preenchimento de característica (FCS) funciona de forma semelhante a uma chuva colorida na lagoa, ou seja, um neurônio situado na FCS recebe uma característica como entrada e espalha-a para outros neurônios vizinhos, veja na figura III.1. Os obstáculos são os sinais vindo do BCS(localização de bordas) para restringir o processo de preenchimento de características, separando as regiões de coloração, como mostra a figura III.2.

A questão é saber qual é a coloração final em cada ponto da lagoa. O leitor dever ter percebido que num ambiente de processamento serial, o número de interações necessárias é enorme. Mesmo com processamento paralelo, os obstáculos na lagoa exigirão uma série de cálculos para achar interseções das ondas com os

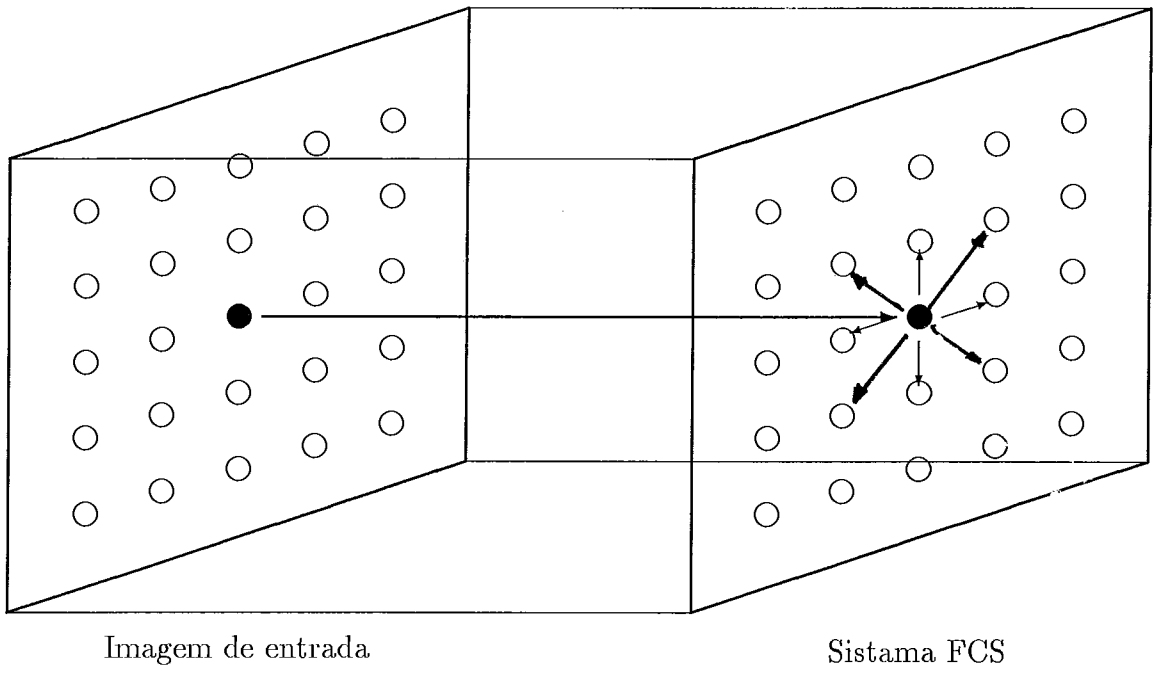


Figura III.1: Propagação de característica de um neurônio da rede neuronal FCS.

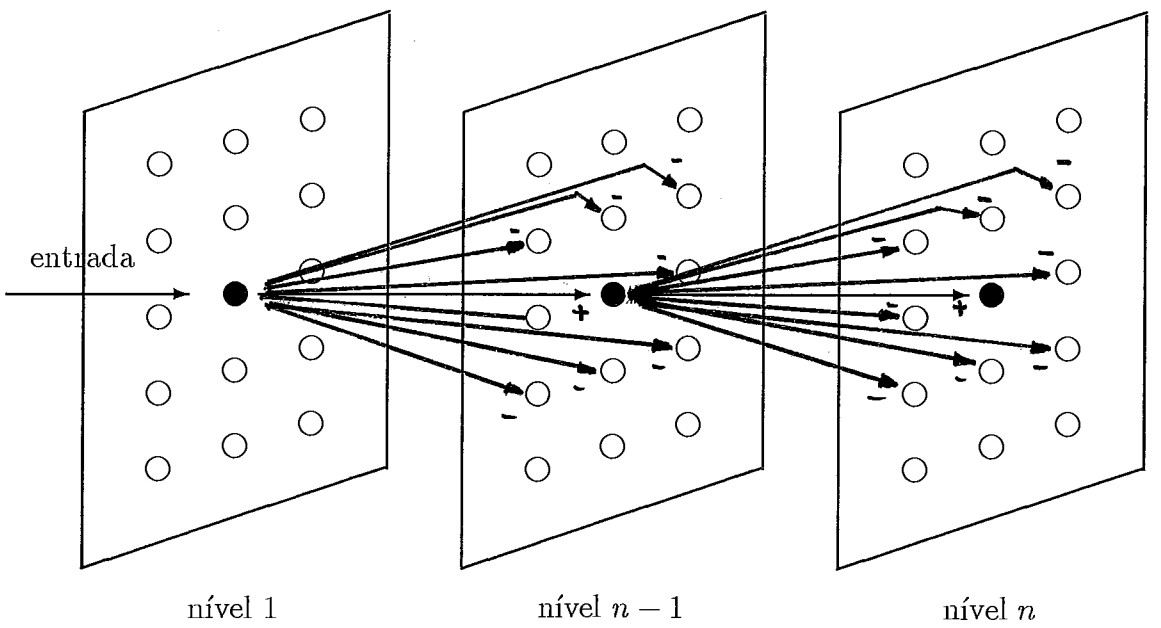


Figura III.2: Sinais vindos dos neurônios do BCS para restringir o processo de preenchimento de característica do FCS.

obstáculos. Por sorte, em um ambiente de processamento paralelo e distribuído, as influências dos obstáculos no processo da coloração são facilmente resolvidos, graças às comunicações de BCS a FCS, que dão informações da existência das influências dos obstáculos, que impedem a propagação das características no processo de preenchimento de características, por isso, sem requerimento de cálculos de interseções. Novamente, a importância de comunicações entre os sistemas BCS e FCS justifica a viabilidade da escolha de processamento paralelo distribuído.

III.3 As Características Peculiares da CORT-X, BCS e FCS

Os sistemas CORT-X, BCS e FCS possuem características peculiares que facilitam o ambiente distribuído para multiprocessamento. Uma das peculiaridades mencionadas está contida na topologia dos sistemas. Ao se observar a topologia com mais detalhes, percebe-se que todas as células das redes possuem vizinhos estritamente homogêneos, ou seja, as distâncias relativas das células vizinhas ao centro do campo receptivo são as mesmas para todas as células da rede. Essa característica permite um boa otimização de memória necessária para armazenar as posições das células vizinhas, porque uma tabela global das posições relativas dos vizinhos para cada processador é suficiente para que todas as células dentro do processador saibam quem são seus vizinhos.

Na revisão da CORT-X, vimos que as células do filtro possuem identificação quanto ao nível, escala, orientação e posição. Então, cada célula da CORT-X possui cinco dimensões de identificação. Com excecão da dimensão do nível, que é síncrono, uma vez que os vários níveis tais como máscara orientada, competição, interação, cooperação possuem uma sequência de execução em pipeline. As outras dimensões podem ser executadas assincronamente, logo elas devem ser paralelizadas ao máximo.

Outra peculiaridade relevante a ser considerada está na relação da vizinhança das células com as dimensões. É essencial que em uma dada célula em uma dimensão saiba-se quais as dimensões nas quais as células vizinhas estão

contidas. Essas informações facilitam a escolha da topologia dos multiprocessadores e otimização de comunicações. Por isso a relação de vizinhança e dimensão será descrita agora.

III.3.1 Relação de Vizinhança do Filtro CORT-X

O objetivo dos itens seguintes é mostrar em cada nível do filtro CORT-X, as relações de vizinhanças com as dimensões, nível de entrada e nível de saída. Facilitando assim identificação de peculialidades para a escolha de topologia dos multiprocessadores. de cada nível,

- Máscara Orientada

  Entrada : Nível de Imagem de Entrada

  Saída : Nível de Competição 1

  Vizinhança : Dimensão de Posição Bidimensional

- Competição 1

  Entrada : Nível de Mascara Orientada

  Saída : Nível de Competição 2

  Vizinhança : Dimensão de Orientação e Dimensão de Posição Bidimensional

- Competição 2

  Entrada : Nível de Competição 1

  Saída : Nível de Interação e Nível de Cooperação

  Vizinhança : Dimensão de Orientação

- Interação

  Entrada : Nível de Competição 2

  Saída : Nível de Cort-x

  Vizinhança : Dimensão de Posição Bidimensional

- Cooperação

  Entrada : Nível de Competição 2

  Saída : Nível de Cort-x

  Vizinhança : Dimensão de Posição Bidimensional

- Cort-X

  Entrada : Nível de Interação e Nível de Cooperação

  Saída : Nível de Saída do Filtro Cort-x

  Vizinhança : Não Tem

III.3.2 Relação de Vizinhança do Sistema BCS

A diferença entre o sistema BCS e o filtro CORT-X está no ciclo de realimentação do nível de realimentação ao nível de competição 1. A característica de realimentação dá ao sistema um potência considerável para deteção de contornos subjetivos. Por outra lado, ela exige uma sincronização mais sofisticada para resolver o problema de *deadlock*.

- Máscara Orientada

  Entrada : Nível de Imagem de Entrada

  Saída : Nível de Competição 1

  Vizinhança : Dimensão de Posição Bidimensional

- Competição 1

  Entrada : Nível de Mascara Orientada e Nível de Realimentação

  Saída : Nível de Competição 2

  Vizinhança : Dimensão de Orientação e Dimensão de Posição Bidimensional

- Competição 2

  Entrada : Nível de Competição 1

Saída : Nível de Interação e Nível de Cooperação

Vizinhança : Dimensão de Orientação

- Cooperação Orientada

  Entrada : Nível de Competição 2

  Saída : Nível de Cort-x

  Vizinhança : Dimensão de Posição Bidimensional e Dimensão de Orientação(orientação perpendicular)

- Realimentação

  Entrada : Nível de Cooperação Orientada

  Saída : Nível de Competição 1

  Vizinhanca : Dimensão de Posição Bidimensional

III.3.3 Relação de Vizinhança do Sistema FCS

O sistema FCS é oriundo da classe de rede neuronais do tipo *Recurrent On-Center-Off-Surround.* Os neurônios recebem e emitem potenciais para os mesmos vizinhos. Diferenciando o outra classe *Forward On-Center-Off-Surround* porque o campo receptor e campo emissor são diferentes, ou seja, os vizinhos são diferentes na emissão e recepção de potenciais. A simulação de FCS é mais difícil devido a necessidade de sincronismo de estabilidade de todas as células nervosas e o sistema BCS.

- Sistema FCS (Feature Contour System)

  Entrada : Nível de Cooperação do sistema BCS

  Saída : Sistema FCS

  Vizinhança : Dimensão de Posição Bidimensional

III.4 Escolha da Topologia dos Multiprocessadores

Nesta seção, justificamos a proposta de uma topologia Hipercubo Retangular (nesta topologia, as duas dimensões inferiores do hipercubo são formados por uma grade retangular) para os multiprocessadores ou multicomputadores. Na seção anterior, vimos que as comunicações das células são feitas entre os níveis do filtro CORT-X e existem intensas comunicações entre as células de níveis vizinhos. Por isso, para obter uma simulação com bom desempenho é necessário minimizar essas comunicações inter-níveis. Logo, é razoável evitar o mapeamento de níveis em processadores diferentes.

Já sabemos que as dimensões de escala, orientação e posição podem ser paralelizadas ao máximo. A questão é saber como fazer isso. Como a topologia da rede neuronal já foi vista na revisão do filtro CORT-X, basta escolhermos uma topologia de multi-computadores mais adequada para esta aplicação específica.

Na seção anterior, ressaltamos as peculiaridades do filtro CORT-X. Uma das peculiaridades é a relação vizinhança das células com as dimensões. De acordo com essa relação, percebe-se que em todas as células dos níveis da CORT-X, com excecão do nível de competição 2, as posições das células vizinhas estão situadas em um plano bidimensional. Assim, essas células vizinhas podem ser mapeadas numa grade retangular, com posição situada a uma distância dx e dy do centro do campo receptivo. Os outros níveis da CORT-X podem ser mapeados nas outras dimensões do hipercubo. A figura III.3 tem mais detalhes.

As discussões feitas para o filtro CORT-X são válidas para o BCS e o FCS, logo pode-se usar a topologia hipercubo retangular para este sistemas. Nas próximas seções deverão explorar com mais detalhes as vantagens das características desta topologia. Com isso justificamos a escolha de uma rede de interconexão de matriz retangular nas duas dimensões inferiores de uma topologia hipercubo.

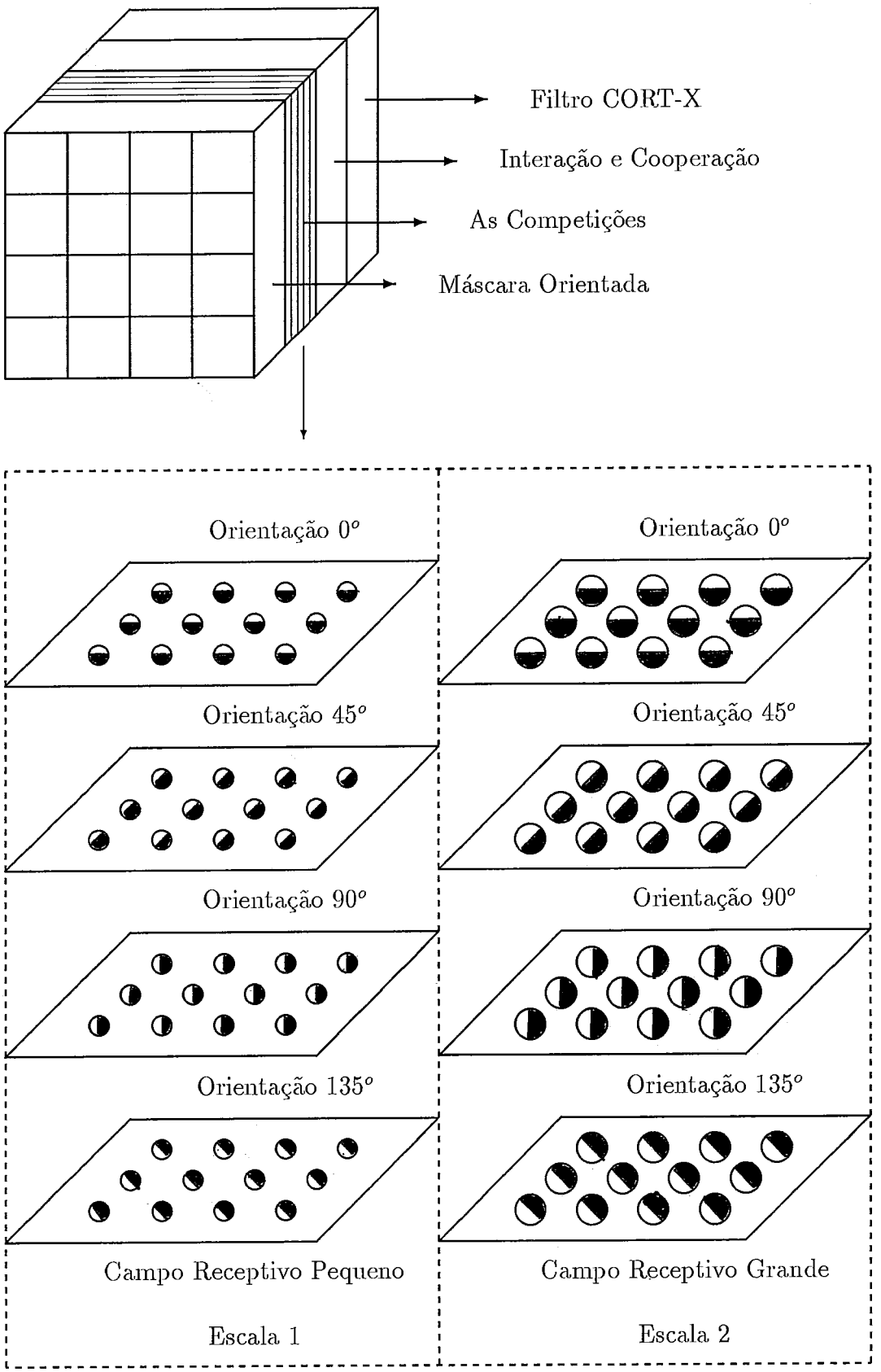


Figura III.3: Topologia de hipercubo retangular de dimensão 3

III.5 Política de Mapeamento Explorando Peculiaridades

Considerando cada célula do CORT-X como sendo um processo, temos então cerca de 1.2 milhões de processos a serem mapeados nos processadores. Infelizmente, não dispomos de multiprocessadores com essa grandeza de processadores. Naturalmente, há uma necessidade de se mapear estes processos nos processadores disponíveis. A seção anterior fez uma breve descrição da política de mapeamento. O detalhamento do mapeamento das várias dimensões da CORT-X na topologia do hipercubo retangular será discutido agora. A figura III.3 auxiliará a visualização da correspondência da dimensão do filtro CORT-X com a dimensão do hipercubo retangular.

A política de mapeamento das várias dimensões da rede em dimensões do hipercubo depende de números de processadores disponíveis. A dimensão do hipercubo pode variar de 2 a 5 dimensões, porque existe uma prioridade no mapeamento das dimensões referentes as posições das células para otimizar comunicações interprocessadores. Logo a dimensão mínima do hipercubo é dois e só cresce quando houver um eficiente mapeamento nestas dimensões. Como a quantidade de processos nestas dimensões é grande, um mapeamento eficiente requer uma quantidade considerável de processadores na rede. Conseqüentemente, a dimensão cresce quando houver uma quantidade considerável de processadores. Felizmente, há computadores com arquitetura altamente paralela como a *Connection Machine* [32] com milhares de processadores que permitem processamento altamente paralelo das redes neuronais contendo milhões de neurônios.

Como vantagem principal, esta topologia permite o mapeamento eficiente do filtro CORT-X numa arquitetura maciçamente paralela, porque cada processador da rede só se comunica com os quatros processadores vizinhos interconectados diretamente, os processadores situados à esquerda, à direita, em cima e embaixo do presente processador. Observe a figura III.4. A implementação do simulador em uma rede de *Transputers* é motivada devido à esta vantagem de comunicação direta com quatro processadores. Por sorte, o *Transputer* possui justamente quatro *links* bidirecionais que fazem comunicações eficientes com quatro *Transputers*

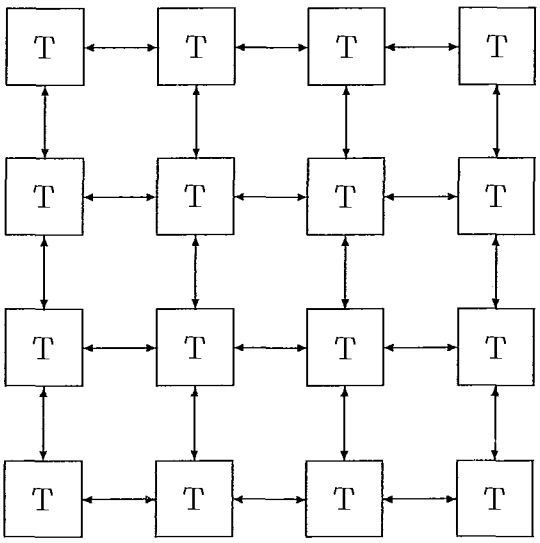


Figura III.4: Topologia de uma rede retangular.

vizinhos. Intuitivamente, a topologia do filtro CORT-X, a topologia do hipercubo retangular e a facilidade de comunicação entre *Transputers* casam-se perfeitamente, pois a topologia do filtro CORT-X pode ser mapeada elegantemente na topologia hipercubo retangular, e esta mapeada com eficiência numa rede de *Transputers*. O mapeamento discutido nesta seção é válido par os sistemas BCS e FCS.

III.6 Otimização das Comunicações entre Processadores

A prioridade de mapeamento das dimensões de posição na seção anterior é motivada pelo estudo das características peculiares das redes neuronais do tipo *on-center-off-surround* como BCS , FCS e CORT-X. As células de tais redes são sensíveis ás informações locais e às posições onde se encontram. Por isso, a fronteira da região de vizinhança ou campo receptivo possui uma distância máxima dx e dy do centro do campo receptivo. O Mapeamento Retangular na topologia Hipercubo Retangular (veja na figura) III.5 permite uma boa otimização de comunicacões entre processadores, se as condições $dx <= gradex$ e $dy <= gradey$ forem satisfeitas. Estas condições garantem que a distância máxima de comunicacões entre processadores

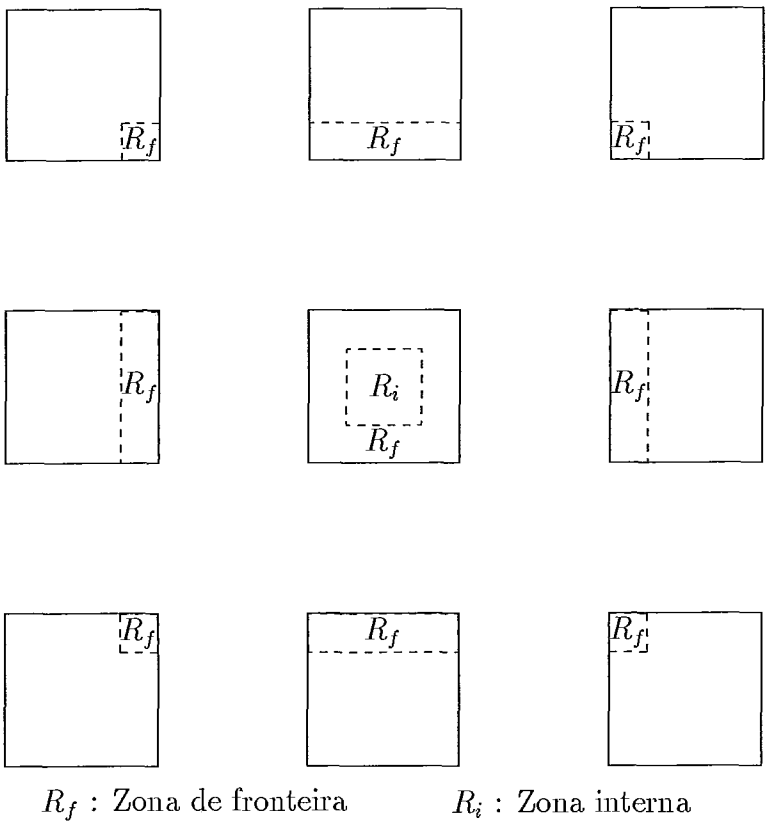
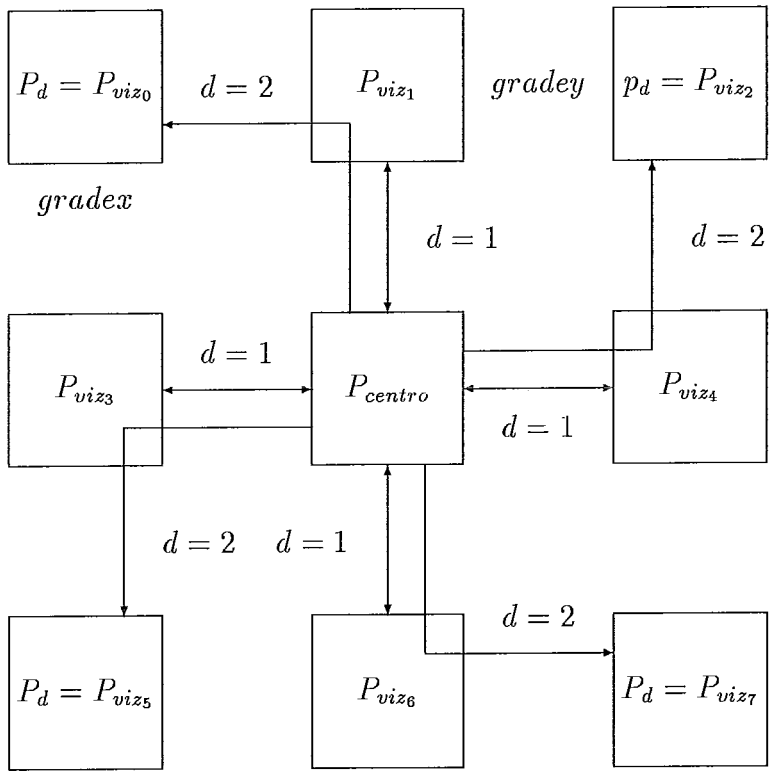


Figura III.5: Regiões de vizinhanças dos neurônios nos processadores.

seja iqual a dois e também garantem a otimização de número de comunicações entre processadores.

As células que se comunicam com as células vizinhas de processadores vizinhos são mostradas nas partes sombreadas da figura III.5. Percebe-se que o número de células dentro de uma grade que se comunicam com os outros processadores estão localizadas na fronteira. Se $dx << gradex$ e $dy << gradey$ então há um grande números de células que não se comunicam com outros processadores, diminuindo assim, um número considerável de comunicações entre processadores. A distância máxima de comunicação entre processadores pode ser vista na figura III.6. Na figura, mostra-se que os processadores P_d situados nas diagonais do P_c são processadores mais distantes que possuem comunicação com P_c. Considerando a distância de comunicação entre processadores como sendo o número de processadores que a mensagem precisa viajar para chegar ao seu destino, então a distância máxima de comunicacão entre processadores da rede é dois. A pequena distância



d = distância entre os processadores
P_{centro} = processador de referência
P_{viz} = processadores vizinhos ao processador P_{centro}

Figura III.6: Distância máxima de comunicação entre processadores.

máxima de comunicação entre processadores dá suporte para o projeto de arquitetura de processadores altamente paralelos, pois esta distância é invariante com relação ao número de processadores existentes, uma vez que os processadores não precisam enviar mensagens a longa distância para comunicarem entre si.

Enfim, o problema tradicional de *overhead* de comunicação é substancialmente atenuado pela otimização de números de comunicações e distâncias de comunicações entre processadores. Isso tornou-se possível, graças ao casamento perfeito no mapeamento entre a topologia da redes CORT-X,BCS e FCS, topologia Hipercubo Retangular e facilidades de comunicação entre *Transputers*.

III.7 Sincronização entre Vários Níveis de Processamento

Sincronização eficiente é essencial para processamento paralelo e distribuído, por isso são embutidas nos projetos de hardware de multiprocessadores MIMD, as implementações sofisticados de métodos de sincronização tradicionais. Nos projetos de *hardware*, concentram-se três grupos de implementações: implementações baseadas em semáfaros, implementações baseadas em monitores e implementações baseadas em trocas de mensagens. O artigo [15] faz uma boa revisão dos métodos de sincronização mencionados. Nesta seção, concentraremo-nos na discussão dos métodos de sincronização baseados em troca de mensagens nos *Transputers*.

Trocas de mensagens podem ser consideradas como uma abordagem diferente de sincronização, uma vez que, todas as comunicações entre processos são feitas via transmissões explícitas de valores entre si: em vez de ler e escrever variáveis compartilhadas, os processos enviam e recebem mensagens. A comunicação via tracas de mensagens é um recurso específico do domínio dos sistemas distribuídos, por exemplo, *Transputers*, apelidados de processadores de comunicação pelo suas facilidades de comunicação.

A implementação de *send* e *receive* possui dois parâmetros primários: Como os processos origens e destinos são nomeados? Como as comunicações

são sincronizadas? Tranputer usa nomeação direta no emissor e receptor porque há correspondência direta com os *links* de comunicação ponto a ponto. O segundo parâmetro relaciona-se com o sincronimo dos processos que pode ser síncrono ou assíncronos. Trocas de mensagens assíncronas permite alto grau de paralelismos, entretanto, este tipo de comunicação requer uma quantidade ilimitada de *buffers*. Por outro lado, trocas de mensagens síncronas não usam *buffers*, pois em ambos, emissores e receptores são capazes de bloquear-se. Assim uma troca de mensagem sempre representa um ponto de sincronismo, o emissor conhece o estado do receptor e o receptor sempre recebe informação do estado atualizado. A comunicação baseada na trocas de mensagens síncronos exige certos cuidados dos programadores, uma vez que os processadores podem se bloquear pela espera infinita de mensagens, provocando *deadlock* pelos bloqueios um ao outro sobre as mensagens.

O uso de *buffer* de entrada e *buffer* de saída permite o processamento concorrente dos neurônios situados nas regiões de fronteira R_f. Para evitar o uso de quantidades ilimitadas de *buffers*, introduzimos três canais *put*, *request* e *get* para controlar o *buffer*, o controle segue a mesma filosofia usada pelo Alan Burns[1]. Quando o *buffer* está cheio o canal *put.in* é bloqueado pelo controle do *buffer*, este bloqueio é transitivo nos canais intermediários até atingir o canal de *get.out* dos *buffers* de saída dos processadores vizinhos, evitando envios de mensagens neste canal. Por outro lado, se o *buffer* está vazio o canal *get.in* é bloqueado, consequentemento os processos que recebem mensagens do canal ficam esperando a liberação do canal. Da maneira como os *buffers* são usados, percebe-se que os tamanhos dos *buffers* determinam grau de paralelismo de processamento dos neurônios situados nas regiões de fronteira entre os processadores vizinhos.

O método de sincronismo usado para os neurônios situados na região interna R_i depende de maneira como as comunicações entre este neurônios são implementadas. O primeiro método usa canais lógicos de comunicações que permitem os níveis das redes neuronais, tais como CORT-X, BCS, serem executados em pipeline e os neurônios das redes do tipo *recurrent on-center-off-surround*, FCS, serem executadas em paralelo. O outro método de implementação de comunicações entre este neurônios é acesso direto a memória, o presente método não permite processamento

paralelo dos neurônios e exige sincronismo para mudança de níveis de processamento do filtro CORT-X. A vantagem deste método será discutida na seção seguinte.

III.8 Gerenciamento de Memória

Em um sistema distribuído, cada processador tem sua memória local e as comunicações entre processadores são feitas via trocas de mensagens. Por isso há necessidade distingüir diferentes tipos de uso da memória para facilitar o gerenciamento. A memória pode ser usada para armazenar códigos dos processos, área de trabalho dos processos, mensagens temporárias de comunicações, estruturas de dados e canais de comunicação. Essa distinção facilita a otimização da memória de trabalho, que é fundamental em um ambiente de pouca memória disponível. Por exemplo, a criação de um processo executado em paralelo exige uma área de trabalho, mais ainda, se um processo deseja-se comunicar com outros, há necessidade de área da memória para armazenar os canais de comunicação. Por isso, tem-se que tomar cuidado ao paralelizar os processos, pois o paralelismo máximo nem sempre é a melhor solução.

No entanto, a otimização de memória pode degradar o grau de paralelismo entre os processos. Em contra partida, há um ganho de tempo computacional quando os processos são executados dentro do mesmo processador e não comunicarem com outros processadores. Este ganho de tempo computacional é oriundo da eliminação de tempo de simulação de paralelismo dos processos no mesmo processador. Desta forma, os processos são executados seqüencialmente, eliminando *overhead* de controle dos processos. Logo, o processador só precisa armazenar um código para os processos que não se comunicam com processadores vizinhos, já que todos os neurônios têm o mesmo código e pode liberar espaços ocupados pelas áreas de trabalhos dos processos neurônios executados em paralelo.

A importância da característica mencionada acentua quando se trata do NCP1. Em cada nó do NCP1, há dois processadores: um transputer (responsável pela comunicações) e um processador vetorial i860 (responsável pelo processamento). Nesse caso, a memória do transputer servirá como *buffer* de comunicação com outros nós do NCP1. Infelizmente, o projeto de placa do i860 ainda não está disponível.

Sendo assim, na implementação, o transputer tem dupla função: gerenciamento de comunicação e processamento. Mas, a distinção do uso da memória certamente facilitará a migração de processamento dos processos neurônios para o processador i860.

A otimização da estrutura de dados dos neurônios tornou-se possível graças a caractrística de homogeneidade dos campos receptores de todos os neurônios situados dentro mesmo nível da rede neuronal. Sendo assim só se precisa de uma tabela de distâncias relativas dx e dy dos neurônios vizinhos de cada nível por processador. As identificações dos neurônios vizinhos tornam-se fáceis, basta somar as distâncias relativas com as identifições do neurônio do centro do campo receptivo.

Resumindo, a otimização de memória para armazenar o código dos processos e a otimização de estrutura de dados para armazenar a identificação dos neurônios vizinhos são ótimas porque o código dos processos e a tabela de identificação dos neurônios vizinhos são armazenados somente uma vez por processador.

Capítulo IV

Implementação do Simulador Paralelo e Distribuído de Filtro CORT-X

IV.1 Introdução

No capítulo anterior, nós vimos as principais idéias da definição do simulador quanto à topologia dos multiprocessadores, ao mapeamento das redes neuronais, à otimização de comunicações entre processadores, à sincronização de mudança entre vários níveis de processamento e ao gerenciamento de memória. Para implementar o simulador precisamos colocar as idéias de forma prática, por isso, precisamos conhecer o ambiente de implementação, tais como *hardware* (equipamentos disponíveis) e *software*(linguagem de programação). A junção dos recursos disponíveis com as idéas da simulação possibilitará meios de integração de teoria e prática.

Por isso, a seção seguinte descreverá o ambiente *hardware* e *software* de implementação. Esta seção fará breves descrições sobre a câmera CCD, a placa de captura e exibição chamada de Harlequin, o processador de comunicação *Tranputer*, o processador *pipeline Intel 80860*, o NCP1(Núcleo de Processamento Paralelo 1) e a linguagem de programação OCCAM.

A outra seção deverá mapear a filosofia da simulação nas ferramentas mencionadas. Ela deverá prover os meios pela quais os mecanismos, tais como: gerenciamento de memória, escalonamento dos processos, otimização de comunicações

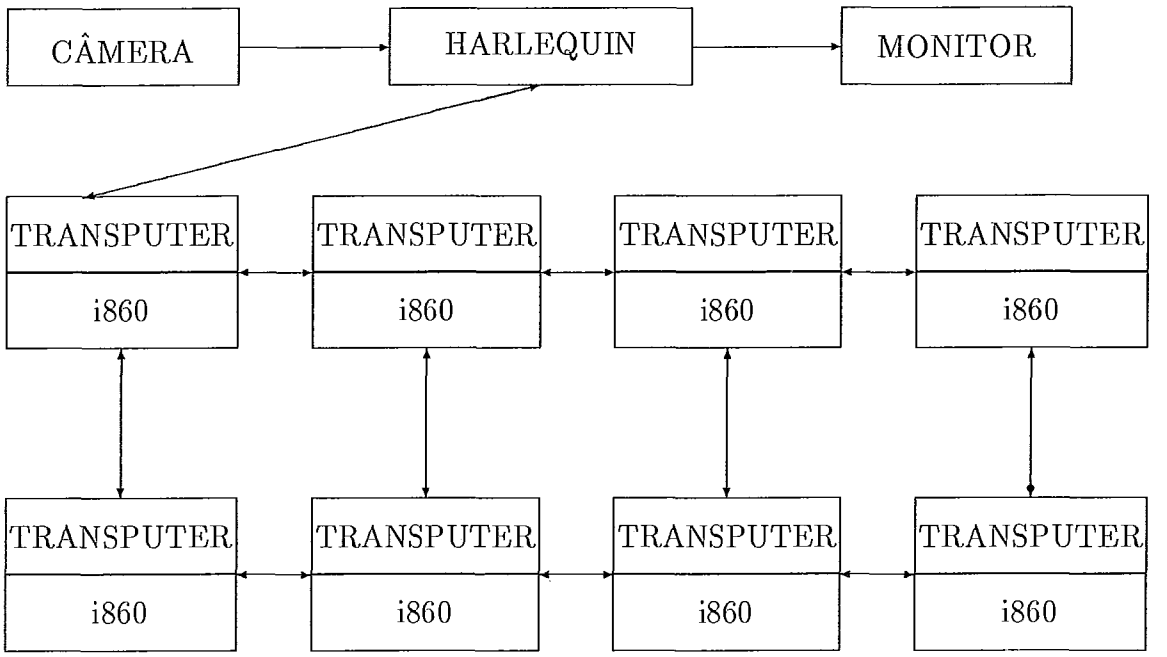


Figura IV.1: Fluxo de informação no ambiente de implementação.

entre processadores e sincronização, são implementados.

IV.2 Descrição do Ambiente de Implementação do Simulador

IV.2.1 Descrição do Recurso de *Hardware* Utilizado pelo Simulador

Nesta seção descreveremos o ambiente de implementação do simulador. A entrada do simulador é uma imagem digitalizada capturada por uma placa *Harlequin FrameGabber/Graphics* que por sua vez recebe uma imagem analógica capturada por uma câmera CCD. A figura IV.1 descreve o fluxo de informações com mais detalhes. Tendo-se a imagem digitalizada e armazenada, pode-se então enviá-la ao NCP1 para ser processada. O NCP1 consiste de uma rede de nós formados por um processador de comunicação *Transputer* e um processador de *pipeline* i860. A comunicação entre os dois processadores está centralizada via memória bipolar, a qual eles podem acessá-la simultaneamente. Desta forma, a comunicação entre o *Transputer* e o i860 seja eficiente. A comunicação entre os nós é feita via *links* dos *Transputers*, por isso

chamados de processadores de comunicações. O resto da seção descreverá com mais detalhes as características dos componentes deste ambiente.

Câmera CCD

A câmera CCD utilizada é um *HandScan* da *Sony*, contendo seguintes características.

- Freqüencia de Captura: 30 frames por segundo;
- Resolução: 512x486 pontos;
- Sinal de Saída: imagem analógica do padrão NTSC.

Harlequin

A placa *Harlequin* é projetada para atender uma estação de trabalho dedicada aos sistemas de visão e é baseada na alta performance do *Transputer*. Ela combina uma unidade de captura de imagem de TV com uma unidade gráfica de alto desempenho em uma única placa com *interface* ao computador pessoal da linha IBM-PC. A placa *Harlequin* pode ser usado em conjunção com NCP1(Núcleo de Processamento Paralelo) para estender a potência de processamento de desempenho em torno de milhares de mips.

A unidade Harlequin tem a capacidade de exibir uma imagem de resolução 512x512 pontos de 256 cores, podendo usar *palette* de 262.144 cores. A placa tem um *Transputer* T800 com 1 *Byte* de RAM dinâmica, tendo com suporte a liguagem OCCAM, que foi especialmente projetada para aproveitar todo o potencial do *Transputer*. Além disso, ela possui dois *buffers* de memória de vídeo que faz transferência dual automática, tal que 99% desta memória é disponível para processamento de aplicações.

As características detalhadas da placa *Harlequin* são:

- Captura quadro de Imagem,

  - dois *buffers* de vídeo 512x512 pontos (8 bits por pontos)
  - 4 canais de entrada multiplexada para TV
  - aceita sinais padronizados monocromáticos CCTV e NSTC
  - flexível sincronização de vídeo e *display*
  - permite captura e a exibição simultânea de *buffers* diferentes

- Exibição gráfica de cores,
  - gera 512x512 pontos de 8 bits de cores simultaneas.
  - saída RGB de 18 bits gerado pela tabela de *pallete* IMS G170
  - compatível com *software* gráfica Immos B007

Transputer

O *Transputer* pertence à família de dispositivos VLSI programáveis, incluindo controlador de disco rígido, processador de ponto flutuante, processador gráfico, dispositivo de processamento de sinal e processador de próposito geral de 32 bits. Um *Transputer* padrão é ilustrado na figura IV.2. Como vemos na figura, o *Transputer* contém memória interna e quatro *links* de comunicação, com os quais se conecta diretamente com outros *Transputers* Este *links* são conectadas ao processador principal via quatro *interfaces* de *links*. A potência do *Transputer* vem da sua capacidade simultânea de se comunicar com quatro *links* e de executar um processo interno. Este facilidade só foi adquirida graças às *interfaces* que permite gerenciamento independente de comunicações dos *links* de acesso direto à memória. O *Transputer* utilizado na implementação tem capacidade de tranmitir 10 *mbytes* por segundo por link, totalizando 40 *mbytes* por segundo nas suas quatro *links*.

As principais vantagens das comunicações ponto a ponto entre os *Transputer* são:

1. Não necessitar de barramento de alto desempenho para comunicação;
2. A velocidade média de comunicação não satura quando processos extras são adicionados;

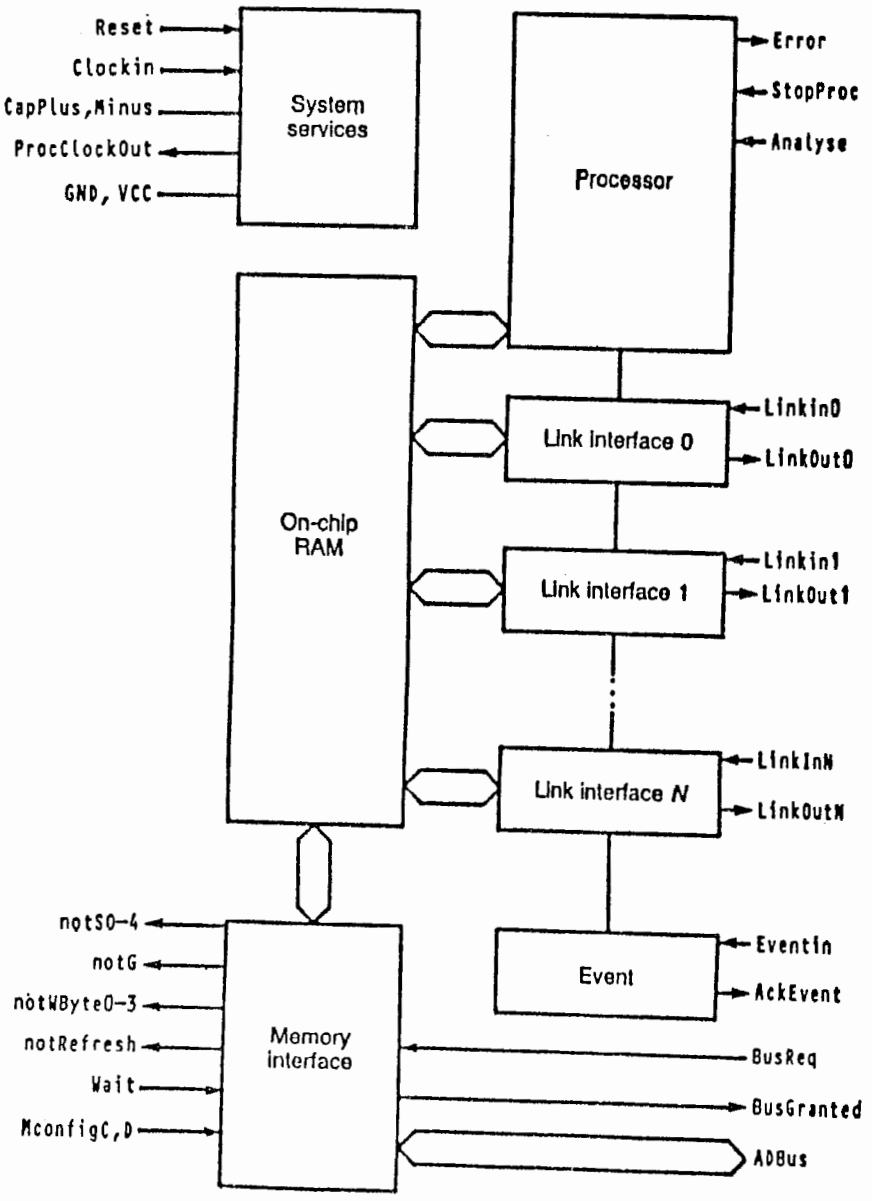


Figura IV.2: Visão esquematizada de um *Transputer* de propósito geral.

3. Sistemas de portes arbitrárias pode ser construídos, desde que todas as conexões sejam locais e distância curta; e

4. O *layout* das placas é fácil de ser projetado.

As principais desvantagens da comunicação ponto a ponto é o número de processadores intermediários (distância entre processadores fonte e destino) que as mensagens tenham que percorrer até atingir o processador destino.

Descrevemos agora, as características físicas do *Transputer*:

- *Transputer* é uma pastilha VLSI CMOS;
- Umas pastilha de memória rápida RAM estática
- 32-bits *Transputer* de acesso à 4 gigabytes de memória
- Um tranputer pode funcionar na velocidade entre 10 a 100 MIPS
- O ciclo de processo tem uma duração de 50ns, o ciclo de 25ns é possível
- Um link transfere dados à taxa de 10, 20, ou 40 megabits por segundo, uma taxa mais veloz também é possivel.
- Falha de sincronização é menor que 0.1 FIT(uma falha em dez mil milhões horas de operação no dispositivo.
- *Transputer* com mais de 4 *links* são tecnicamente possível.

IV.2.2 Descricão do *Sofware* Utilizado pelo Simulador

Dentro das várias linquagens de programação paralelas disponíveis, a linguagem OCCAM possui maior aproveitamento do alto desempenho do *Transputer*, por isso, esta linguagem foi escolhida para implementação do simulador. Vamos só descrever as características essenciais da OCCAM para o entendimento da integração mencionada. O leitor pode consultar [1] para saber mais informações sobre a linguagem occam.

A Linguagem Occam

Vamos restrigir a nossa descrição da linguagem OCCAM em três construdores de processos: SEQ, PAR e ALT, e nas operações de enviar e receber de canal de comunicação.

Na OCCAM, a comunicação entre os processos pode ser feita através de canais lógicos, onde são feitas as trocas de mensagens. As operações de enviar e receber são descritos agora.

```
ALT
   canal.entrada ? temp – receber mensagem
      canal.saida ! temp – enviar mensagem
```

O construdor SEQ faz a execução sequencial de uma coleção de processos, esta coleção pode ser de qualquer tamanho. O processo SEQ terminará quando o último subprocesso terminar. O sintaxe é descrito agora :

```
SEQ
   processo 1
   processo 2
   ⋮
   processo n
```

Os processos concorrentes são definidos pelo construtor PAR, todos os subprocessos são completamente independentes uns aos outros, a não ser que hajam interações explícitos através de canais de comunicações. O primeiro uso do construdor PAR é expressar concorrência e o outro uso é introduzir característica não determinístico no programa. Um processo PAR termina quando e somente quando todos os seus subprocessos terminarem. Os subprocessos do construtor PAR requerem *overhead* de areás de trabalhos, por isso os subprocesso devem ser

executados em concorrência quando eles requerem os dois usos mencionados. O contrutor PAR tem a seguinte sintaxe.

```
PAR
  processo 1
  processo 2
  ⋮
  processo n
```

O construtor ALT é frequentemente usado para multiplexar um número canais de entradas para um canal de saída ou para selecionar um código do processo servidor de acordo com o canal de entrada. O presente exemplo mostra os usos do construtor ALT, os dois primeiros canais de entradas são multiplexadas para canal de saída e os dois últimos canais de entradas selecionam códigos diferentes de execução.

```
VAL INT max IS 5:
CHAN OF REAL32 canal.entrada0, canal.entrada1:
CHAN OF REAL32 canal.entrada2, canal.entrada3:
CHAN OF REAL32 canal.saida:
REAL32 temp:
PAR
  WHILE TRUE
    ALT
      canal.entrada0 ? temp
        canal.saida ! temp
      canal.entrada1 ? temp
        canal.saida ! temp
      canal.entrada2 ? temp
        -- alguma ação
      canal.entrada3 ? temp
        -- outra ação
```

IV.3 Integração do Ambiente de Implementação com Filosofia da Simulação

IV.3.1 Introdução

A integração do ambiente de implementação com filosofia da simulação consiste no mapeamento das idéias apresentadas do capítulo anterior nos recursos disponíveis descritos da seção anterior. Devemos discutir os meios pela quais os mecanismos, tais como gerenciamento de memória, otimização de comunicações entre processadores, sincronização e escalonamento dos processos, são implementados. Antes de detalhar tais mecanismos, faremos uma introdução ao princípios gerais de implementação.

Relembrando o objetivo do simulador, devemos capturar uma imagem de uma câmera de vídeo que envia uma imagem analógica à placa *Harlequin* onde ela será digitalizada. Logo em seguida, a nova imagem é enviado ao NCP1 que simula uma rede neuronal. Esta rede deve transformar a imagem digitalizada em uma imagem contendo o resultado da simulação e novamente enviá-la à placa Harlequin para ser exibida num monitor. Par obter processamento em tempo real, toda a seqüência de execução deve ser feita no máximo 1/30 segundos. Este tempo de processamento será usado no próximo capítulo para avaliar o desempenho dos recursos disponíveis.

De acordo com objetivo traçado, devemos identificar tarefas a serem implementadas. Para isso, usamos o princípio de dividir para conquistar. Para facilitar a divisão de tarefas, nós atribuimos as tarefas aos dois processos centrais denominados de processo mestre e processo servidor. O processo mestre é responsável pelo gerenciamento do simulador, controlando os processos servidores, atribuindo local e ordem de execução. Conseqüentemente, o processo servidor segue a orientação do processo mestre. Desta forma, as tarefas serão alocadas sob o aspecto funcional do processo mestre ou processo servidor. Então, temos as seguintes tarefas:

- Processo Mestre:
  - Comunicar com Usuário;
  - inicializar as variáveis;
  - aquisição de Imagem;
  - escalonar processos neurônios nos processadores disponíveis;
  - definir rotas;
  - controlar processos servidores:
    * inicializar processos servidores;
    * emitir valores iniciais ao processos servidores;
    * sincronizar os processos servidores;
    * emitir parada momentânea dos processos servidores; e
    * emitir término dos processos servidores;
  - exibição de imagem.
- Processos Servidores:
  - Inicialização;
  - definir rotas;
  - prover meios de comunicação entre os neurônios;
  - receber valores iniciais; e
  - controlar subprocessos neurônios.

Uma vez identificadas as tarefas, vamos esclarecer as funcões elas exercem. Quando necessário, procuraremos buscar a integração do *sofware* e *hardware* juntamente o detalhamento das tarefas.

IV.3.2 Processo Mestre

O objetivo do mestre é gerenciar as tarefas gerais do simulador de tal forma que haja harmonia entre várias etapas de execução dos processos servidores. Como as tarefas

do processo mestre já foram identificadas, precisamos agora esboçar as relações de precedência de execução das tarefas através de um pseudo-algoritmo.

```
{... Processo Mestre
  SEQ
    ... Ler parâmetros da rede neuronal CORT-X
    ... Ler configuração da rede NCP1
    ... Mapeamento dos neurônios nos processadores
    ... Enviar inicialização aos nos folhas da rede NCP1
    WHILE NOT fim
      SEQ
        ... Aquisição de imagem
        PAR
          ... Emitir dados de entrada ao servidores
          ... Receber resultados dos servidores
          ... Emitir controle aos servidores
          ... Receber controle dos servidores
          ... Detecção de estabilidade para FCS
        ... Exibição da imagem resultante
        ... Testar o fim do ciclo
    ...Emitir Termino de execução aos servidores }
```

O conceito de algumas da tarefas do mestre são óbvias, e suas implementações são meramente técnicas. Só abordaremos as tarefas que exigem uma explicação mais detalhada.

Tarefas de Emissão de Inicialização, Emissão de Entrada e Emissão de Controle

As tarefas de emissão de inicialização, emissão de dados de entrada e emissão de controles precisam de uma política de roteamento. Esta política é baseada na estrutura da topologia hipercubo retangular discutida no capítulo anterior. De acordo com a discussão, são duas as dimensões do hipercubo, pois o multicomputador NCP1

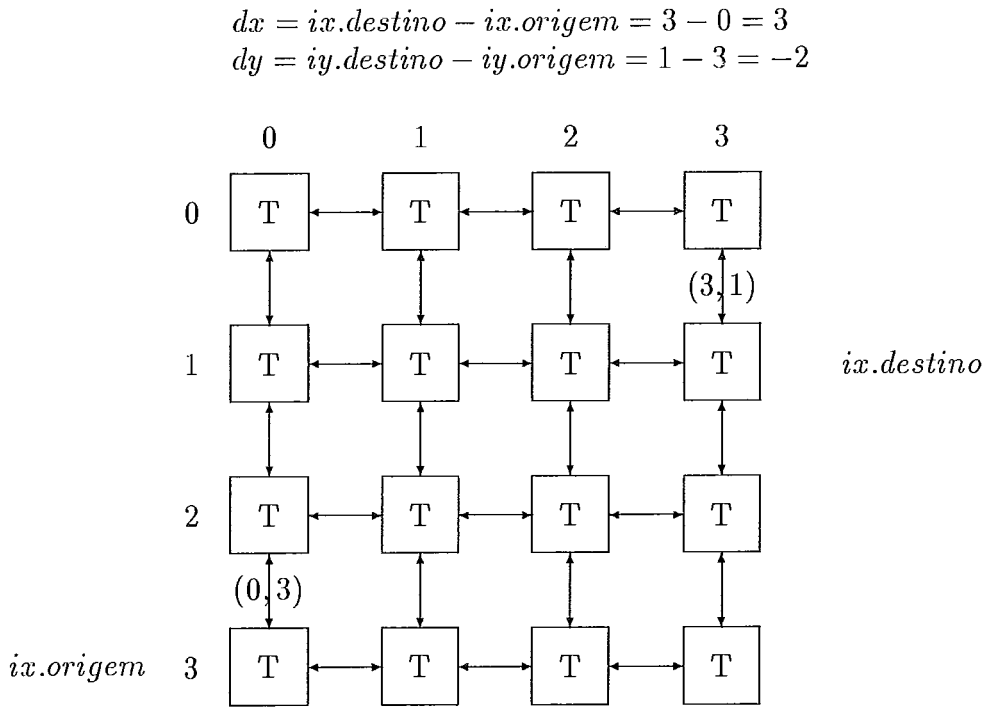


Figura IV.3: Posicionamento do processador na topologia de rede retangular.

disponível para implementação possui baixa granularidade. Então o hipercubo bi-dimencional é equivalente a uma rede retangular. Cada nó da rede retangular é identificado por ix e iy, sendo ix no eixo x e iy no eixo y.

$$ix = p/p.hor$$

$$iy = p - (dy * p.hor) + 1,$$

Esta rede possue características que facilitam a definição da política de roteamento. A facilidade é dada da sequinte forma: dado um nó origem da rede, qualquer nó destino da rede está situado a distância dx e distância dy do nó origem, sendo que dx distância no eixo x e dy distância no eixo y, ver na figura IV.3. As distâncias dx e dy pode ser calculada por :

$$dx = ix.destino - ix.origem$$

$$dy = iy.destino - iy.origem$$

Uma vez que temos a localização do processador destino por meio de deslocamento *dx* e *dy* em relação ao processador origem, se embutirmos este deslocamento em cada mensagem de comunicação entre processadores, torna-se fácil uma mensagem chegar ao destino, basta que *dx* e *dy* sejam iguais a zero. Detalharemos esse mecanismo de comunicação na seção do processo servidor.

Tarefa de Mapeamento dos Neurônios nos Processadores

Outra tarefa que merece um detalhamento maior é a tarefa de mapeamento dos neurônios nos processadores. Como haviamos dito o mapeamento da CORT-X será feito num hipercubo retangular bidimencional. As dimensões de nível, escala e orientação da topologia do filtro serão projetadas na topologia da rede retangular, ressaltando que a dimensão de nível será projetado no tempo de execução, ou seja, os níveis serão executados seqüencialmente (ver na figura III.3).

Assim, a topologia projetada de dimensão dois do filtro CORT-X é facilmente mapeada na topologia hipercubo retangular bidimensional. Supondo que cada grão do hipercubo retangular seja um tranputer, então temos uma matriz *nxCubo* por *nyCubo* de *Transputers*, sendo que *nxCubo* é o número de *Transputers* no *eixox* e *nyCubo* é o número de *Transputers* no *eixoy*. Adotando, *nxCortx*, número de neurônios no *eixox* e *nyCortx*, número de neurônios no *eixoy*, do filtro CORT-X, as divisões *nxCortx/nxCubo* e *nyCortx/nyCubo* podem traçar limites que dividem a topologia projetada do filtro CORT-X em grades retangulares. Todos os processos neurônios situados dentro da grade retangular seão alocados para serem processados em um processador. O algoritmo para calcular os limites da grade para cada processador é :

```
p.hor := nxCortx/nxCubo
p.ver := nyCortx/nyCubo
SEQ p := 0 for Nro.Processador
  SEQ
    ver := p/p.hor
    hor := (ver*p.hor)
```

```
Limite.Esq[p]:= DIMENSAO.X*hor
Limite.Dir[p]:= DIMENSAO.X*(hor+1)
Limite.Inf[p]:= DIMENSAO.Y*ver
Limite.Sup[p]:= DIMENSAO.Y*(ver+1)
```

Tarefa de Controle

As tarefas de controle servem para controlar sincronização entre vários níveis de execução, que é necessária devido ao mapeamento dos níveis do filtro CORT-X sobre o tempo, ou seja, um nível só começa o processamento quando todas as células do nível anterior terminarem de processar. É importante ressaltar que este mapeamento só é viável quando houver escassez de memória, caso contrário as células deste nível podem ser processadas em *pipeline*.

Tarefa de Detecção de Estabilidade do Sistema FCS

A tarefa de detecção de estabilidade de FCS não foi implementada em OCCAM porque FCS está fora do escopo deste trabalho. No entanto, vamos dar uma idéia de como ela pode ser implementada. De acordo com o relatório técnico [16], o processo mestre deve armazenar os vários eventos de processamento dos processos neurônios até que um novo estado global seja atingido. A detecção de estabilidade é feita através da comparação do estado global atual com o anterior. O mestre deve armazenar $n^2 + n$ potenciais de neurônios, sendo n números de neurônios do FCS. Percebe-se o processo mestre deve exigir grande quantidade de memória. Para solucionar este problema, sugerimos que o mestre armazene somente eventos mais recentes, eliminando assim os eventos intermediários. Desta forma, o processo mestre só precisa armazenar $2n$ potenciais, reduzindo uma quantidade considerável de memória.

IV.3.3 Processo Servidor

Sob orientação do processo mestre, o objetivo dos processos servidores é o gerenciamento dos subprocessos neurônios. Para isso, ele precisa prover condições de comunicações entre os subprocessos neurônios, de tal forma que, o fato dos subprocessos vizinhos estarem em outros processadores torna-se transparente.

Para entender melhor o processo servidor, faremos uma breve descrição das características funcionais do comportamento do servidor. Acompanhando a figura IV.4, percebe-se que o processo servidor possue quatro processos vizinhos, conectados através dos *links* bidirecionais. O servidor deve está sempre atento para receber as mensagens vindos destes *links*, e armazená-los no *buffer fifo*(first in and first out) circular de entrada. Em seguida, as mensagem são tratadas para averiguar seus destinos. Caso a mensagem pertença ao processo servidor ela é colocada nos *buffers* internos para processamento posterior, senão elas são colocadas diretamente para o *buffer* de saída para ser enviada aos processos servidores vizinhos. As mensagens dentro destes *buffers* internas são lidas e enviadas aos subprocessos neurônios através de canais de comunicações. Só então, os subprocessos neurônios destinos serão acordados para processamento pelos eventos das chegadas das mensagens. Percebe-se que os *buffers* internos evitam *overhead* de processamento das mensagens internas ao servidor, assim, as mensagens em trânsitos chegarão aos seus destinos com mais eficiência. Além disso, há uma distinção de rotinas que fazem *interface* com os processos servidores vizinhos e de rotinas que não os fazem. Por isso, na migração de uma rede de *Transputer* para NCP1 com i860, as rotinas de interfaces continam sendo executadas pelo *Transputer* e as rotinas internas passam a ser executadas pelo i860.

Uma vez que, uma noção rápida do fluxo funcional do processo servidor já foi dada, introduzimos agora, um pseudo-algoritmo que mapea as características funcionais deste processo. Em seguida, devemos esclarecer as rotinas deste algoritmo.

```
{...Processo Servidor
  PAR
```

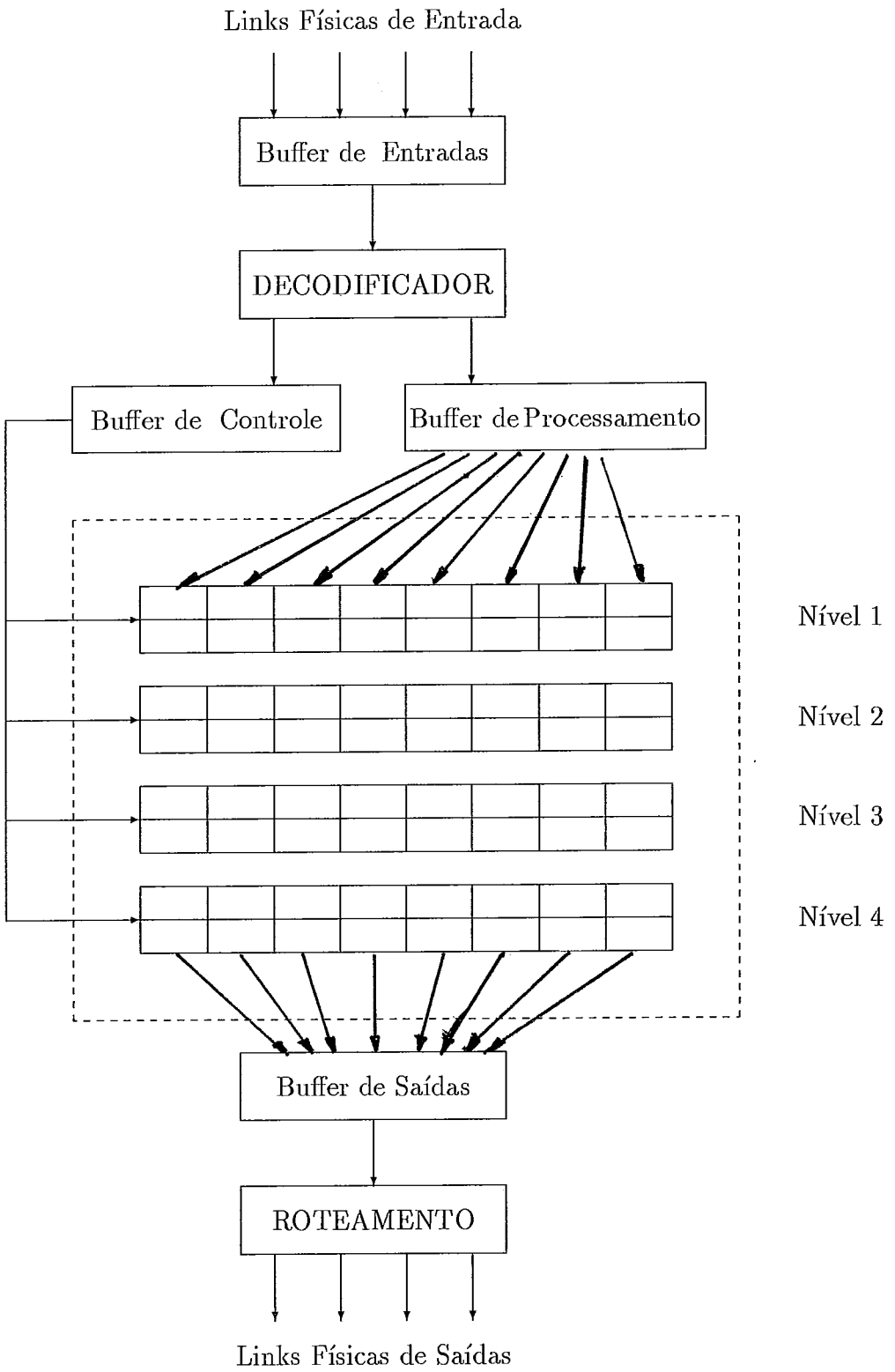


Figura IV.4: Descrição Funcional do processo servidor.

... Receber mensagens dos *links* e enviar para *buffer* de entrada
... Controlar *buffer* de entrada
... Controlar *buffer* internos
... Controlar *buffer* de saida
... Filtrar canais internas de saidas para *links* externos
... Sincronizar de Niveis do filtro CORT-X
... Converter identificador global de mensagens em canais
locais de comunicações com subprocessos neurônios
SEQ para cada nivel da rede
SEQ
PAR para cada subprocessos neurônios de cada nivel
SEQ
... Receber Potencial dos neurônios vizinhos
... Calcular o potencial do neurônio
... Emitir potencial para neurônios vizinhos
... Sincronizar mudança de nivel }

Antes de esclarecer as rotinas do algoritmo, vamos discutir as comunicações entre processadores. As informações *dx* e *dy* em cada mensagem são suficientes para o algoritmo de roteamento detectar o destino em uma topologia de rede retangular. De acordo com esta topologia, uma mensagem pode vir dos quatro lados: esquerdo, direito, superior e inferior. Se a mensagem vier do lado esquerdo, então ela está se deslocando no sentido direito. Para atualizar a distância entre os processadores a ser percorrida pela mensagem, basta fazer um decremento de 1. Veja na figura IV.5, as atualizações das mensagens vindas dos outros lados.

Quanto a emissão de mensagens para outros processos servidores, alguns casos devem ser observados para facilitar a escolha de um *link* dentro dos *links* possíveis, para a mensagem chegar ao seu destino. O objetivo é diminuir o congestionamento do tráfego das mensagens, pela distribuição destas pelos *links* possíveis através de uma convencão adotada de acordo com destinos das mensagens. Veja as figuras IV.6 e IV.7 para averiguar os casos possíveis e a convencão adotada. As seguintes subseções darão o detalhamento das rotinas do processo servidor.

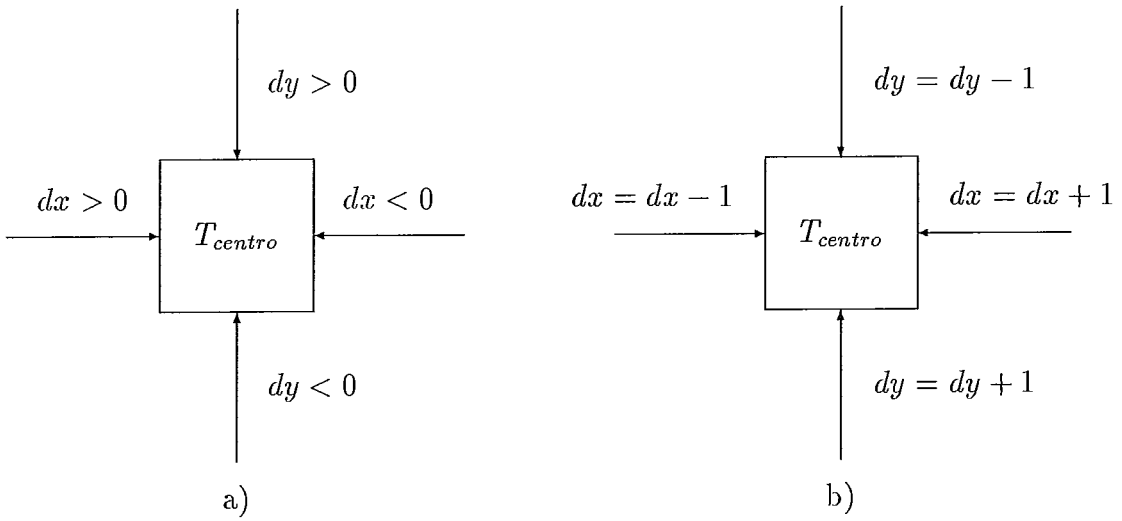


Figura IV.5: a)Convenção dos sentidos de deslocamento das mensagens de acordo com valores de dx e dy. b)Atualização dos valores dx e dy de acordo com sentido da chegada das mensagens.

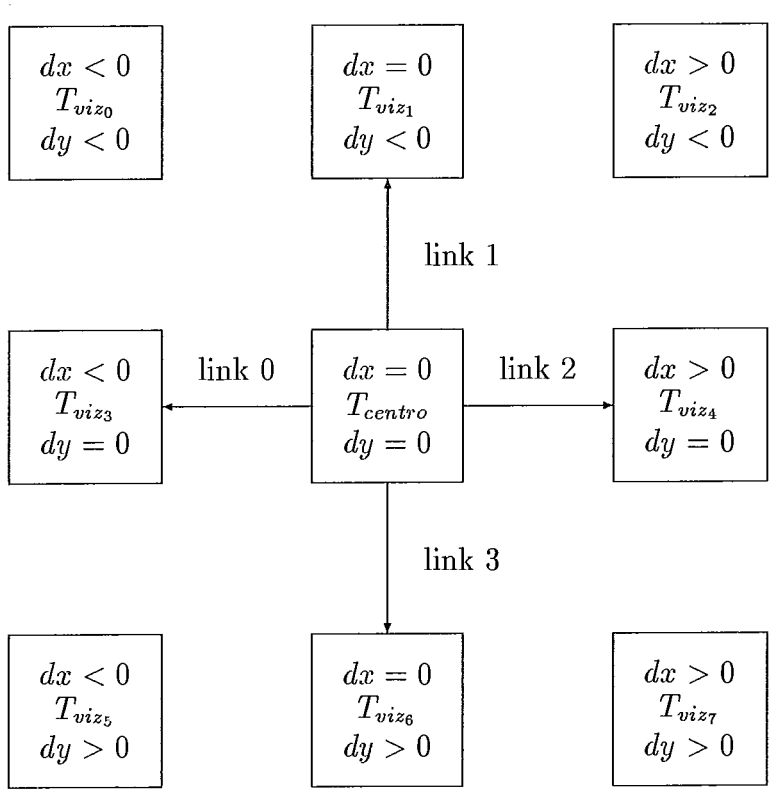


Figura IV.6: Casos possíveis pela qual uma mensagem pode se deslocar pelos *links* possíveis.

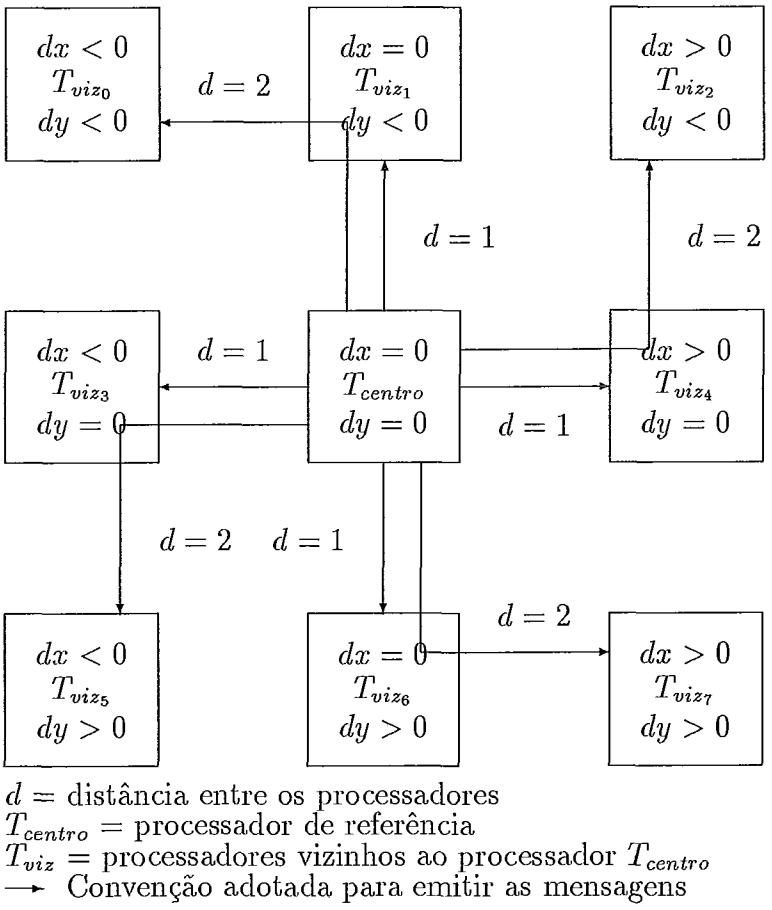


Figura IV.7: Convencão adotada para emitir mensagens pelos *links* possíveis.

Comunicação do Meio Externo com o Processo Servidor

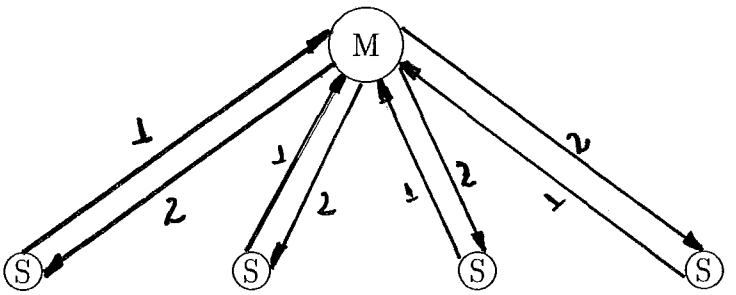
A rotina de receber mensagens dos *links* e enviá-los ao *buffer* de entrada é muito simples, basta usar ALT da linguagem OCCAM da seguinte maneira:

```
WHILE NOT fim
    ALT
        link0.in ? msg
          dx:= dx - 1
          put.buffer.in ! msg
        link1.in ? msg
          dy:= dy - 1
          put.buffer.in ! msg
        link2.in ? msg
          dx:= dx + 1
          put.buffer.in ! msg
        link3.in ? msg
          dy:= dy + 1
          put.buffer.in ! msg
```

O comando ALT fica em estado de espera até que uma mensagem chega em um dos canais, em seguida, um subprocesso que atualiza dx ou dy, e o envia para o *buffer* de entrada através do canal *put.buffer.in.* O processo ALT repete até que o *flag* de fim seja verdadeiro.

Sincronização da Mudança de Nível de Simulação

A sincronização (vide figura IV.8) é feita via interação com processo mestre, Todos os processos servidores enviarão suas mensagens de controle ao processo mestre para comunicar a mudança de nível de processamento. Assim que processo mestre receba todas as mensagens de controle, ele enviará mensagens de sincronização a todos os processo servidores, avisando-os para continuarem o processamento de níveis posteriores. Nota-se que o sincronismo evita a mistura de mensagens de controle enviado



M = Processo Mestre
S = Processo Servidor
1 = Mensagem de Controle, Saída
2 = Mensagem de Init, Controle, Entrada, Processamento

Figura IV.8: Sincronização de mudança de nível.

pelo processo mestre com as mensagens de processamento do nível anterior ao sincronismo. No entanto, o sincronismo não evita a mistura de mensagens de controle com mensagens de processamento do nível posterior. Para tornar o sincronismo de mudança de nível mais eficiente, pode-se usar trocas de mensagens de controle entre os processos servidores vizinhos, desta forma os servidores vizinhos ficam cientes do evento de mudança de nível do presente processo servidor. Logo, os processos servidores não precisam esperar o sincronismo de todos os processos servidores, como acontece no controle mestre-servidores, para continuarem os seus processamentos. Obtendo assim, o paralelismo e o desempelho almejados.

Justificativa do Uso de *Buffer*

A utilização de *buffer* de entrada permite a comunicação assíncrona dos processos servidores. A implementação do *buffer* usa a mesma filosofia empregada pelo [1]. Usando este recurso, os *links* ficam liberados para receber outras mensagens posteriores.

Outra utilização de buffer é facilitar o sincronismo de transição de mudança de nível que esclarecemos agora. Nessa transição de mudança de nível, o

buffer de entrada pode conter mensagem de controle ou mensagem de processamento do nível posterior à sincronização vinda de outros processadores. Pode ocorrer o caso de as mensagens chegadas antes da mensagem de sincronização não serem consumidas, causando assim, *deadlock*. Para resolver este problema, empregamos o uso de *buffers* internos de usos diferentes: *buffer* de processamento e *buffer* de controle. Desta forma, as mensagens não consumidas são armazenadas no *buffer* de processamento, possibilitando que a mensagem de sincronização seja armazenada no *buffer* de controle para ativar eventos de mudança de nível.

Converter Identificadores Globais em Canais Locais de Comunicações com Neurônios

A conversão dos identificadores globais nas mensagens exige uma explicação mais precisa de como os neurônios se comunicam, externamente e internamente com processo servidor. A figura III.5 mostra que existem duas regiões de neurônios. A região R_f delimitada com tons escuros é uma zona de fronteira. Todos os neurônios dentro desta zona podem comunicar-se com neurônios vizinhos situados em processos servidores vizinhos. O processamento destes neurônios deve ser feito em paralelo e em alta prioridade para ser mais eficiente.

A outra região não se comunica com neurônios dos processos servidores vizinhos. Para este casos os processos neurônios podem ser processados em baixa prioridade. Percebe-se que esses processos podem ser processados serialmente, desde que utiliza o sincronimo de transição de nível do filtro CORT-X mencionado. Sob ponto de vista da migração do simulador para i860, este processos neurônios podem ser totalmente processados (serialmente ou paralelamente) pelo i860 sem ter a preocupação com comunicação entre os processadores, inclusive usando operações aritméticas em *pipeline* (potência real do i860).

Nas mensagens de processamento, os identificadores globais dos neurônios destinos devem ser decodificados e convertidos em endereços das canais internas de comunicação. Assim, as mensagens chegadas nesses canais causam eventos que vão acordar os processos neurônios situados na zona de fronteira para serem

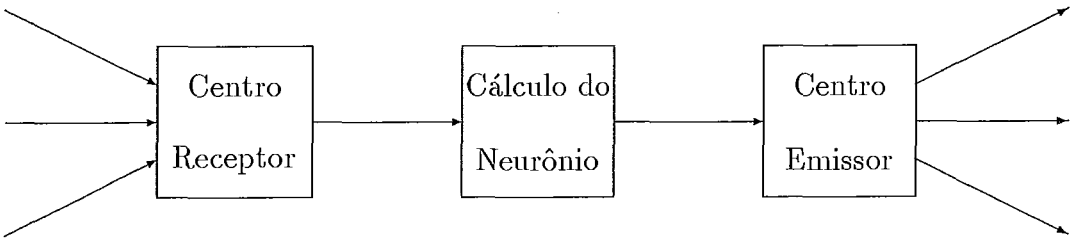


Figura IV.9: Processamento de um neurônio

processados, concretizando o processamento paralelo desses processos. Em sequida, esclareceremos a decodificação das mensagens e a conversão dos identificadores globais em identificadores internos.

Basicamente, há três tipos de mensagem de processamento. Cada mensagem contém informações de identificação do neurônio destino, por exemplo, nível, escala, orientação, posição x e y. Só a posição global x, y é convertida em posição local ix, iy devido ao projeção da topologia do filtro CORT-X em topologia de hipercubo retangular. Tendo estas informações, as potênciais contidas nas mensagens são enviados aos neurônios destinos pelos canais identificadas, quanto ao nível, à escala, à orientação e à posição interna.

Processamento do neurônio

Cada neurônio da rede neuronal simulado é composto por três partes: centro receptor, cálculo do neurônio e centro emissor, veja figura IV.9. O centro receptor fica em estado de alerta. Logo que um potencial do neurônio vizinho chega ao neurônio, o cálculo necessário será feito de acordo com tipo de neurônio. Se ele receber os potenciais de todos os neurônios do campo receptivo então gera um evento para centro emissor que tem a responsabilidade de enviar seu potencial calculado a todos os neurônios do campo emissivo via canais internos de emissão.

Filtrar canais internas de saídas no *buffer* de saída

Esta rotina fica em estado de alerta para capturar os potenciais emitidos pelos canais internos de emissão. O potencial recebido de um dos canais e a identificação global convertida da identificação interna deste canal são codificados de acordo com protocolo da mensagem, montando a mensagem que será enviada ao *buffer* de saída. Do buffer, a mensagem será emitido para processos servidores vizinhos, completando enfim, o ciclo de processamento do processo servidor.

Capítulo V

Resultados Obtidos pelo Simulador Paralelo e Distruído do Filtro CORT-X em uma rede de Transputers

V.1 Introdução

Neste capítulo apresentaremos dois tipos de resultados, o primeiro tipo tenta mostrar a qualidade do filtro CORT-X quanto ao compromisso entre a detecção de borda, a eliminação de ruído e o complemento de borda. Para isso, os resultados intermediários de cada nível do filtro CORT-X serão apresentados e os ajustes de parâmetros das equações serão feitos, para aproveitar melhor as características deste filtro.

O segundo tipo de resultado procura avaliar o desempenho do simulador. Esta avaliação deve ressaltar a eficiência de comunicação entre os processadores para averiguar o recurso computacional necessário para o processamento em tempo real. Para isso, os processos neurônios situados na zona de fronteira são isolados e processados para medir somente tempo de comunicação, testando assim, a potência dos *links* do *Tranputer*. Esta medida de tempo deverá dar suporte para avaliação da capacidade do *Transputer* em processar um número de neurônios em tempo real, uma vez que, o gargalo de um sistema de computação paralelo de memória distribuída está na comunicação lenta entre os processadores.

O tempo de processamento pode ser medido pela subtração do tempo normal de processamento e tempo de comunicação. Este tempo pode ser comparado com a potência do i860, o que dará condições de averiguar o compromisso da junção do *Transputer* com o i860. O compromisso se resume na sequinte dúvida: '*Transputer* tem condições de dar suporte de comunição entre os nós do NCP1 e o i860?".

O restante do capítulo discutirá os resultados do compromisso entre a localização de borda, eliminação de ruído e complemento de borda, e a avaliação do desempenho do simulador para dar uma resposta à dúvida mencionada.

V.2 Resultados do Compromisso entre A Localização de Borda, A Eliminação de Ruído e o Complemento de Borda

Para avaliar os resultados do compromisso entre a localização de borda, a eliminação de ruído e o complemento de borda, precisamos averiguar os resultados intermediários de cada nível do filtro CORT-X. Desta forma, podemos observar como máscaras orientadas percebem as orientações das bordas, como múltiplas escalas espaciais são exploradas para obter uma boa localização de borda com curvatura acentuada e eliminação de ruído, e como completar as bordas rompidas pelo ruído. Em seguida, mostraremos os resultados de cada nível do filtro CORT-X.

V.2.1 Captura de Imagem de Entrada

A imagem de teste é uma imagem digitalizada de uma onça pintada, ela foi escolhida justamente por causa de sua textura complexa para fins de teste(ver a foto V.1. A imagem foi digitalizada pela placa TARGA-32 para obter uma boa qualidade de cores. Cada ponto da imagem é composto por três componentes primários de cores, vermelho, verde e azul. Cada componente é armazenado em um *byte*, portanto possui uma variação de 0 a 255 tons, totalizando em torno de 16 milhões de cores pela permutação dos três componentes. Esses componentes são importantes para

Figura V.1: A foto de uma imagem de onça digitalizada pela placa gráfica TARGA-32

pesquisa futura de percepção de cores baseada em [21]. Na implementação, só usamos o componente vermelho, como mostra a foto V.2, para testar a qualidade do filtro CORT-X.

V.2.2 Máscara Orientada

O objetivo dos níveis de máscaras orientadas é detectar as orientações das bordas. A qualidade de detecção de orientações é avaliada através das fotos V.3, V.4, V.5 e V.6. Percebe-se que em cada foto a intensidade de cada ponto da imagem é ativada mais fortemente de acordo com as direções das máscaras orientadas. Por exemplo, a região 0^0 marcada nas fotos indica uma borda de inclinação 0^0, por isso, a região é mais ativada na foto V.3 que contém máscaras de orientação de 0^0. Seguindo o raciocínio, as regiões 45^0, 90^0 e 135^0 são mais ativadas fotos V.4, V.5 e V.6, de acordo com suas respectivas orientações.

Vamos analisar agora os ajustes dos parâmetros α_s e β_s da equação II.5. Os parâmetros dados pelo artigo [6] são mostrados pelo teste 1 da tabela V.1. Usando estes parâmetros, obtemos péssimos resultados na simulação, vide a foto

Figura V.2: A foto mostra o componente vermelho da imagem da onça

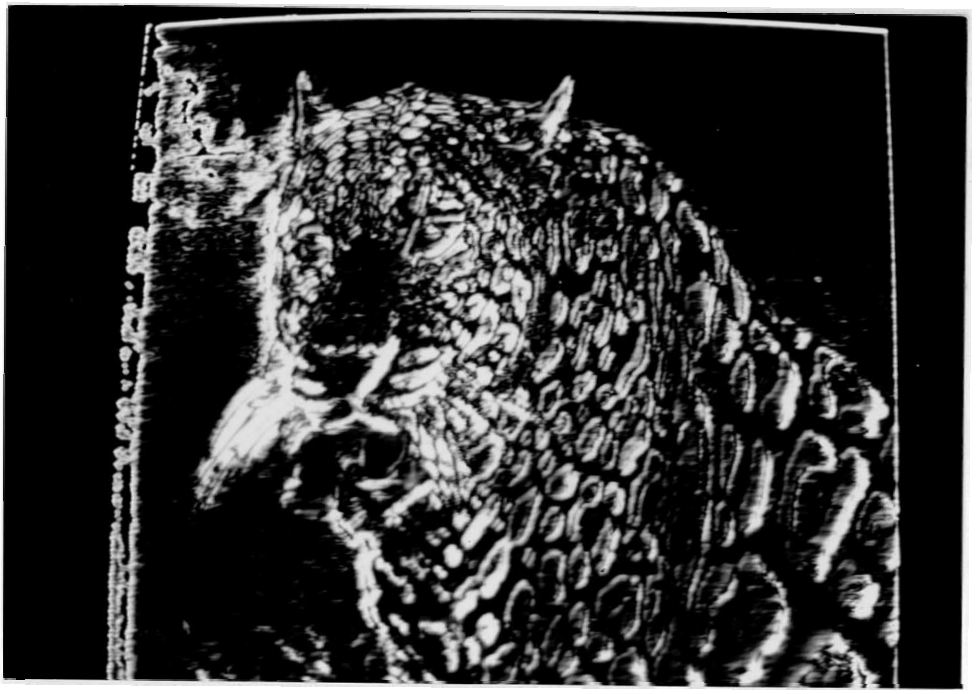


Figura V.3: A foto mostra o resultado de sensibilidade da máscara orientada com orientação de 0^0.

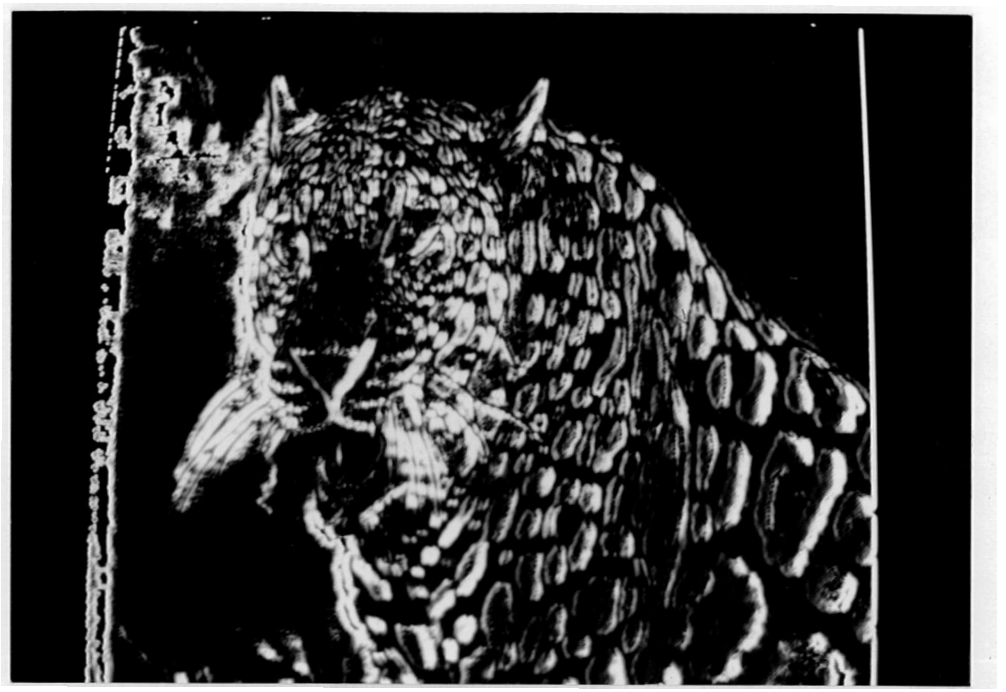


Figura V.4: A foto mostra o resultado de sensibilidade da máscara orientada com orientação de 45^0.

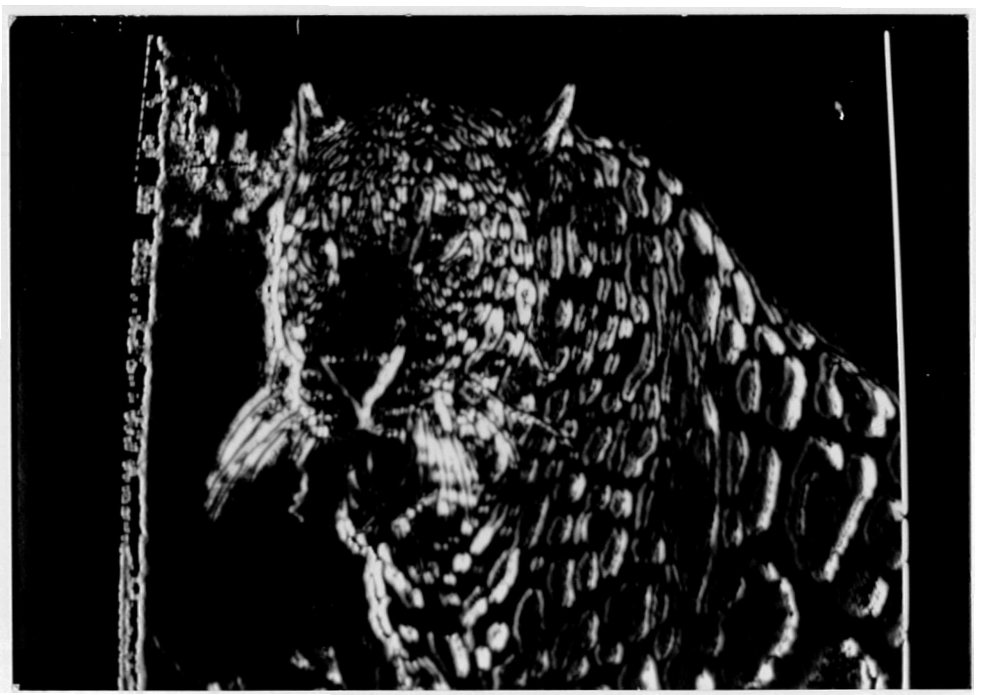


Figura V.5: A foto mostra o resultado de sensibilidade da máscara orientada com orientação de 90^0.

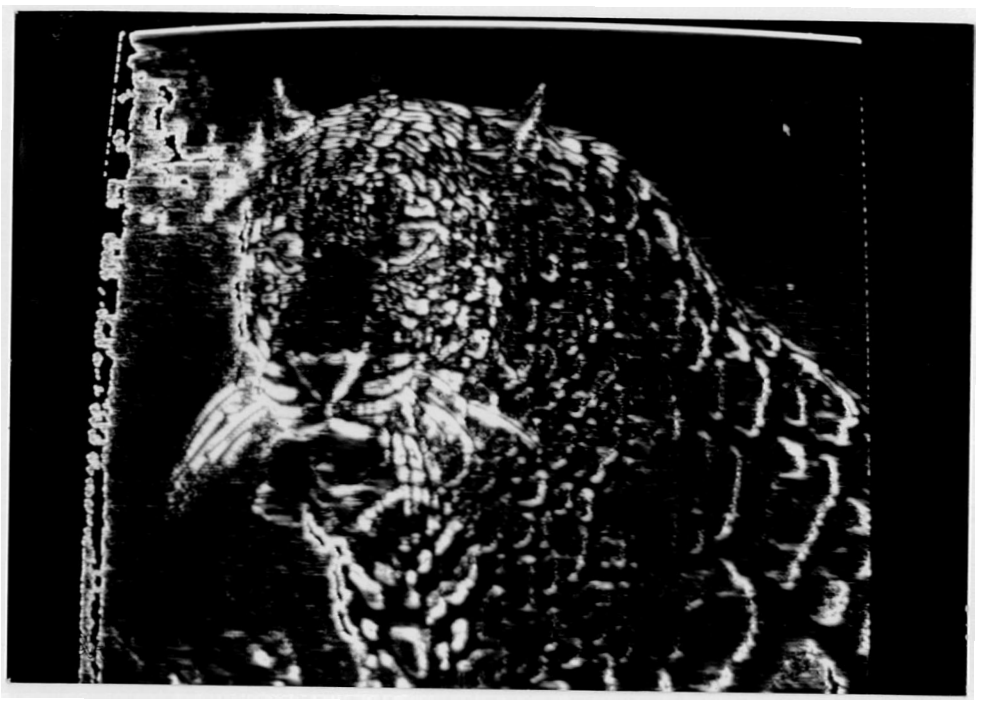


Figura V.6: A foto mostra o resultado de sensibilidade da máscara orientada com orientação de 135^0.

V.7.

A explicação para esses péssimos resultados está na ausência de normalização dos parâmetros em relação ao tamanho das variações dos valores da imagem de entrada, ou seja, "qual a importância da diferença de contraste entre o campo receptivo esquerdo e o campo receptivo direito para pertencer a uma borda?"

Percebe-se na equação II.5 que o parâmetro α_s sempre é maior que 1. Quando maior for o parâmetro α_x maior será a diferença de contraste para tornar-se uma borda. No caso do teste 1, os parâmetros α_s são grandes em relação a os valores da imagem de entrada, por isso, resultados péssimos foram obtidos na simulação.

Nos testes 2, 3 e 4, procuramos ajustar os parâmetros α_s mais próximos de 1 para obter resultados melhores. As fotos V.8, V.9 e V.10 mostram essa tendência de melhores resultados. No entanto, deve-se tormar cuidado para os valores de α_s muito próximo de 1, porque a diferença de contraste detectada pode ser causada pelo ruído(ver a foto V.10).

O parâmetro β_s que serve para controlar a quantidade diferença de

teste	α_1	α_2	β_1	β_2	γ_1	γ_2	σ
1	1.4	2.0	0.3	0.3	1.0	1.0	0.012
2	1.005	1.1	-	-	-	-	-
3	1.01	1.1	-	-	-	-	-
4	1.1	1.1	-	-	-	-	-
5	1.01	1.2	0.005	0.005	-	-	-
6	1.01	1.2	0.05	0.05	-	-	-
7	1.01	1.2	0.3	0.3	-	-	-
8	1.01	1.2	0.005	0.005	0.01	0.01	-
9	1.01	1.2	0.005	0.005	1.0	1.0	-
10	1.01	1.2	0.005	0.005	5.0	5.0	-
11	1.01	1.2	0.005	0.005	1.0	1.0	0.005
12	1.01	1.2	0.005	0.005	1.0	1.0	0.05
13	1.01	1.2	0.005	0.005	1.0	1.0	0.3

Tabela V.1: Lista dos valores utizados para testar os parâmetros das equações das células filtro CORT-X

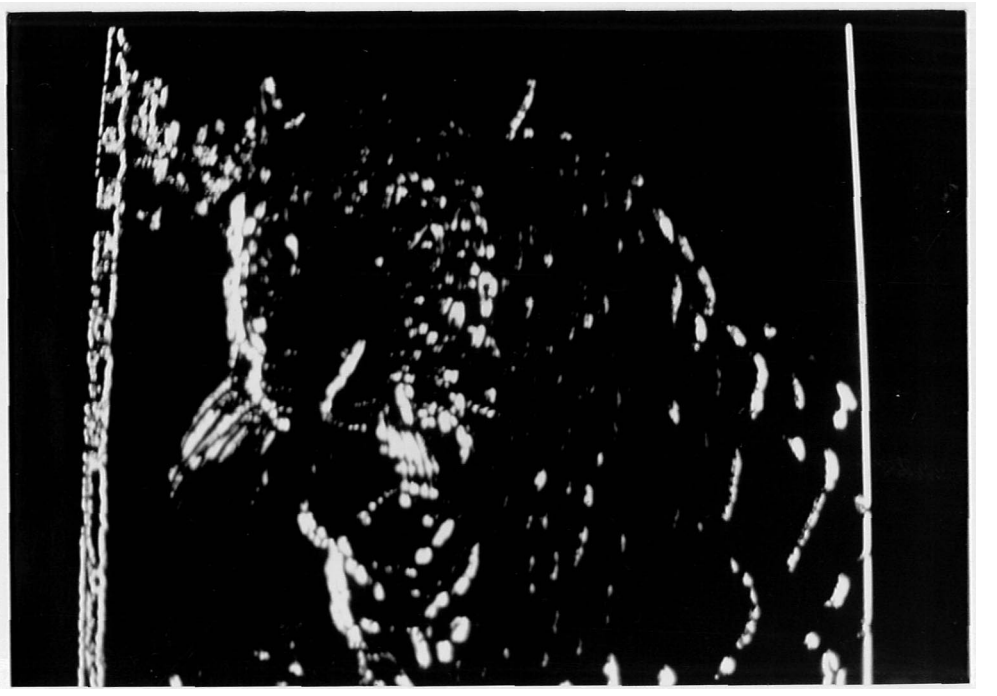


Figura V.7: A foto mostra o resultado de sensibilidade da máscara orientada com orientação de 90^0, cujos parâmetros são dados pelo teste i

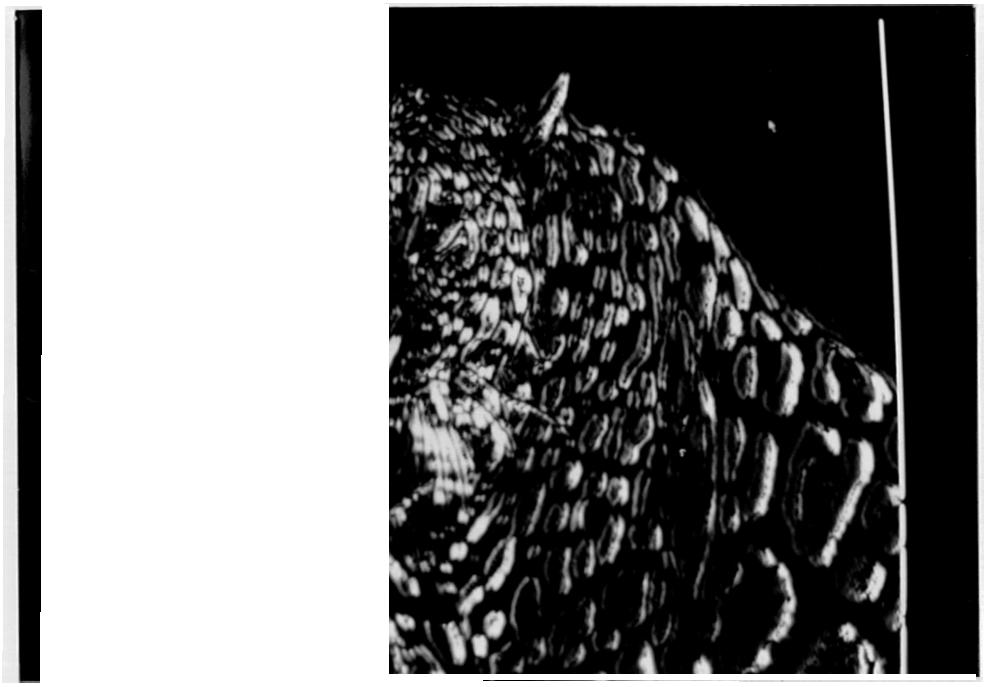


Figura V.8: A foto mostra o resultado de sensibilidade da máscara orientada com orientação de 90^0, cujos parâmetros são dados pelo teste 2

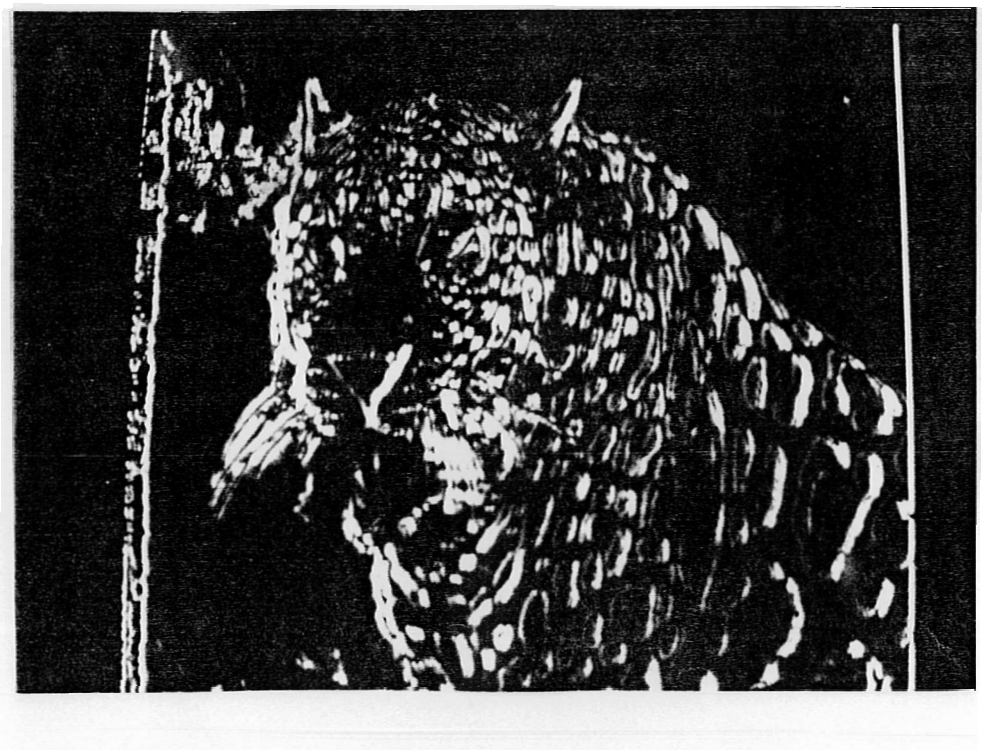


Figura V.9: A foto mostra o resultado de sensibilidade da máscara orientada com orientação de 90^0, cujos parâmetros são dados pelo teste 3

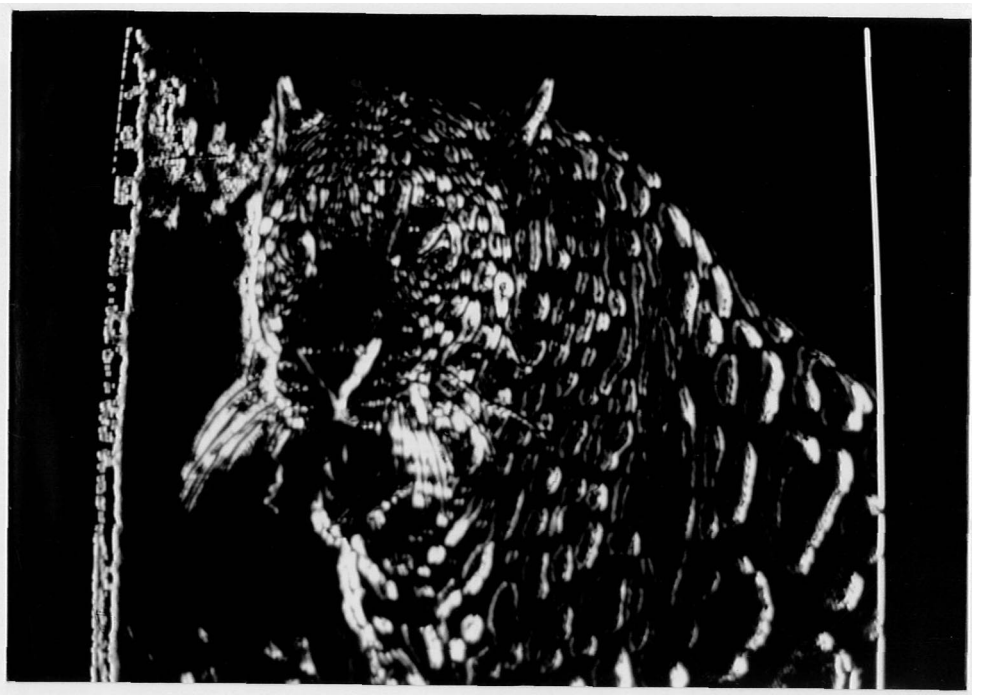


Figura V.10: A foto mostra o resultado de sensibilidade da máscara orientada com orientação de 90^0, cujos parâmetros são dados pelo teste 4

contraste, é essencial para se tornar borda. Os testes 5,6 e 7 procuram testar esse tipo de controle e os fotos V.11, V.12 e V.13 mostram os resultados obtidos. Os valores de baixas intensidades da foto V.11 que são inferiores ao limiar β_s deixam de existir na foto V.12 e mais ainda na foto V.13.

V.2.3 Competição Espacial

Vamos avaliar agora, a eficiência da competição posicional em eliminar ruídos pertos das bordas. Comparando as regiões marcadas nas fotos V.11 e V.15, percebemos que os ruídos na vizinhança da borda deixaram de existir por causa de competição espacial e as intensidades dos pontos na foto V.15 são atenuadas.

No entanto só essa foto é insuficiente para avaliar a importância da competição espacial em eliminação de ruídos, por isto, os testes 8, 9 e 10 foram feitos para efeitos de comparação. No teste 8, os parametros γ_s são ajustados para baixa competitividade, por isso, o efeito de eliminação de ruído deve ser atenuado, vê a foto V.14.

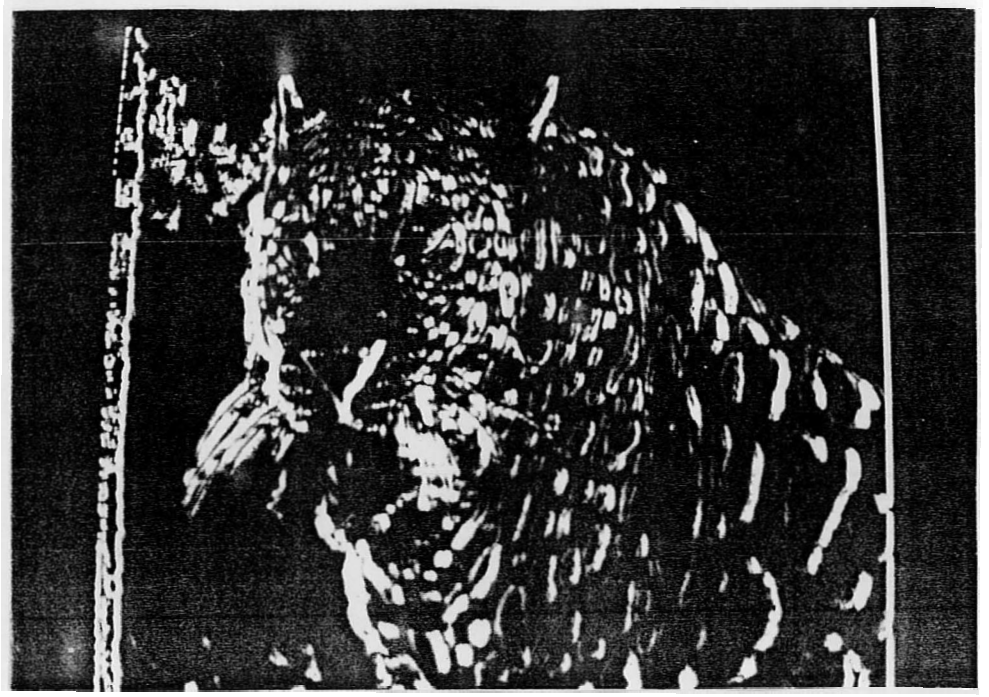


Figura V.11: A foto mostra o resultado de sensibilidade da máscara orientada com orientação de 90^0, cujos parâmetros são dados pelo teste 5

Figura V.12: A foto mostra o resultado de sensibilidade da máscara orientada com orientação de 90^0, cujos parâmetros são dados pelo teste 6

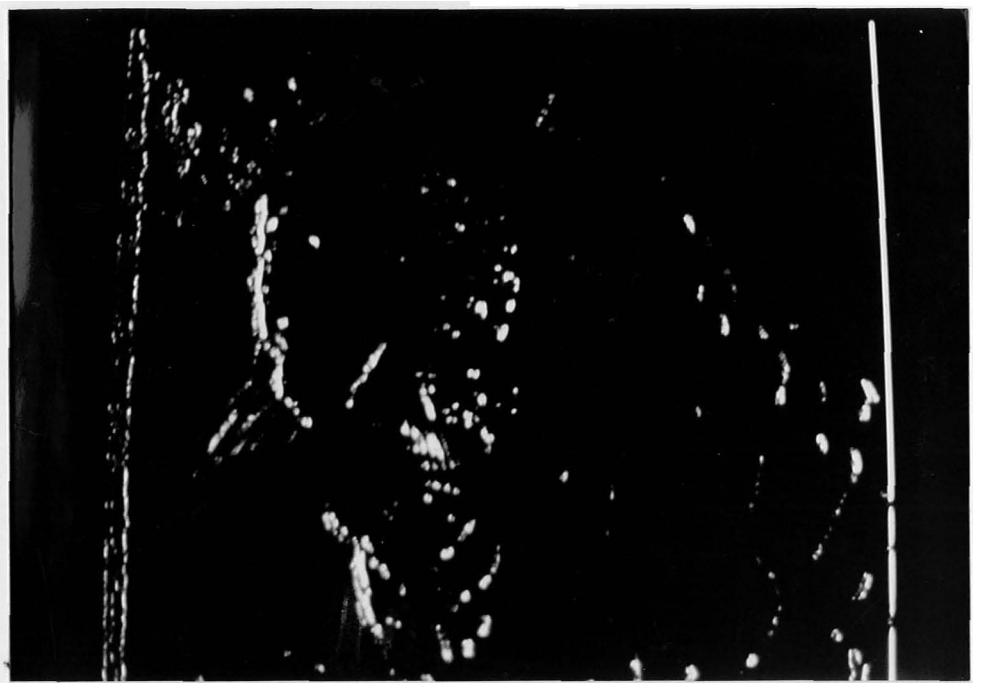


Figura V.13: A foto mostra o resultado de sensibilidade da máscara orientada com orientação de 90^0, cujos parâmetros são dados pelo teste 7

Entretanto, no teste 10, o fenômeno de alta competitividade possui o efeito colateral de baixar muito as intensidades dos pontos da imagem devido a inibição mútuas nas regiões de vizinhanças; a foto V.16 indica esta característica. Por isso devemos ajustar melhor o grau de competitividade para obter uma boa eliminação de ruídos sem comprometer o efeito colateral mencionado. O teste 9 mostrado na foto V.15 possui esse compromisso.

V.2.4 Competição de Orientação

A visualização dos resultados da competição de orientação é muito simples, basta observar a foto V.21. É possível notar que a foto possui melhores resultados do que as fotos V.17, V.18, V.19 e V.20, devido à política de 'vencedor leva tudo" usada pela competição de orientação.

V.2.5 Interação de Múltiplas Escalas Espaciais

A interação de múltiplas escalas espaciais procura explorar o compromisso entre localização de borda, eliminação de ruído e o complemento de borda de diferentes

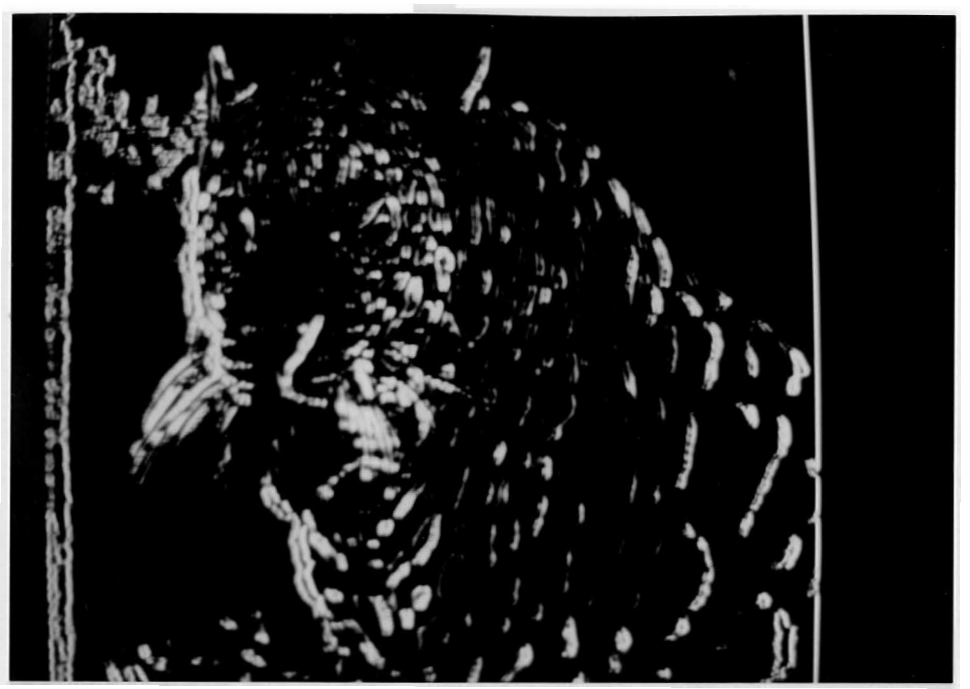


Figura V.14: A foto mostra o resultado da competição espacial com orientação de 90^0, cujos parâmetros são dados pelo teste 8

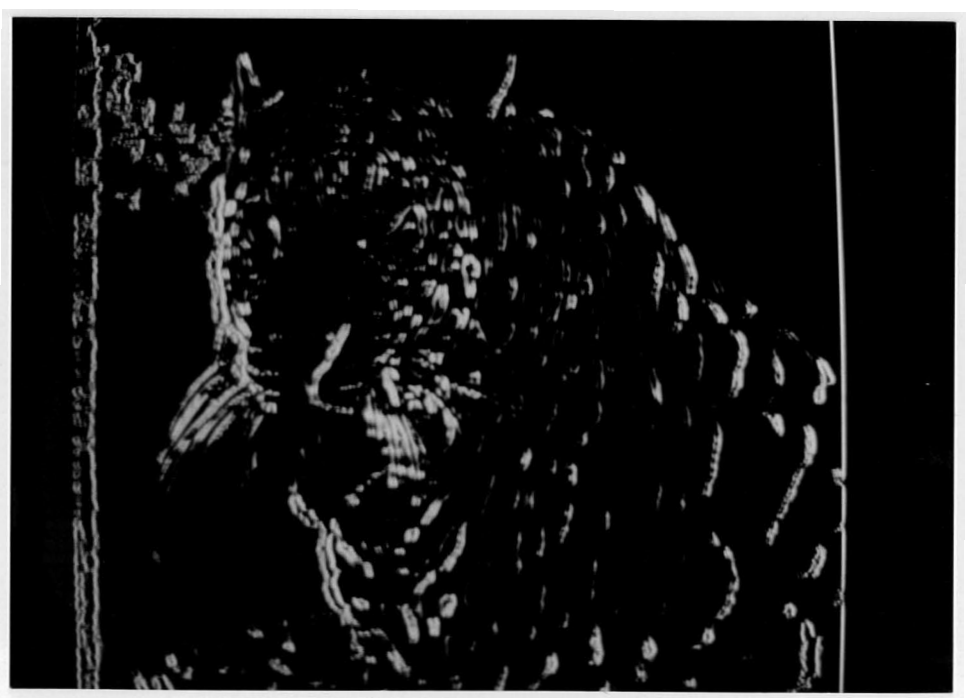


Figura V.15: A foto mostra o resultado da competição espacial com orientação de 90^0, cujos parâmetros são dados pelo teste 9

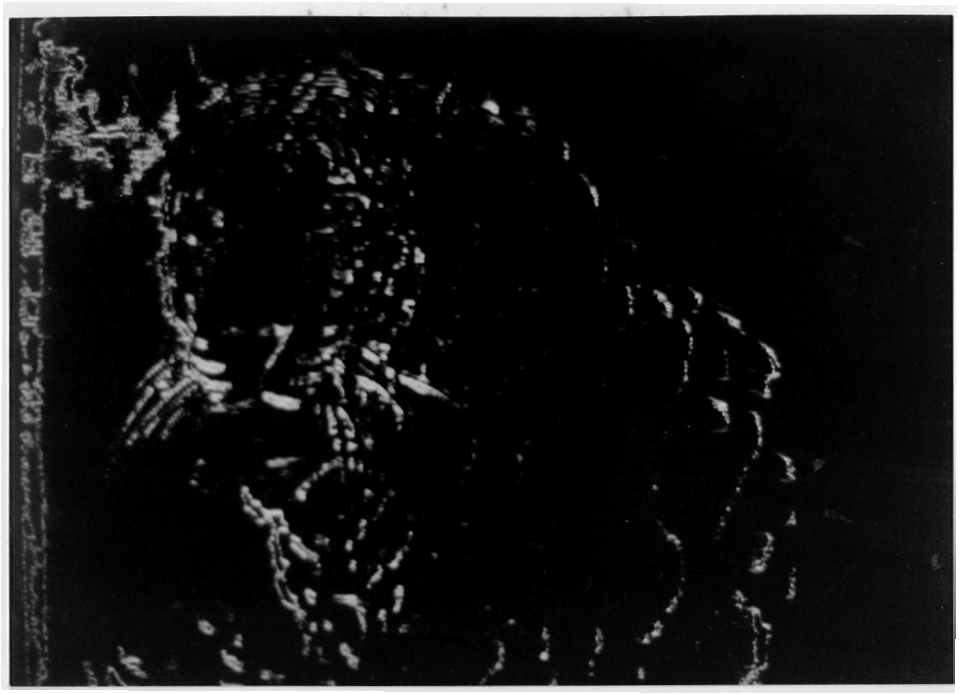


Figura V.16: A foto mostra o resultado da competição espacial com orientação de 90^0, cujos parâmetros são dados pelo teste 10

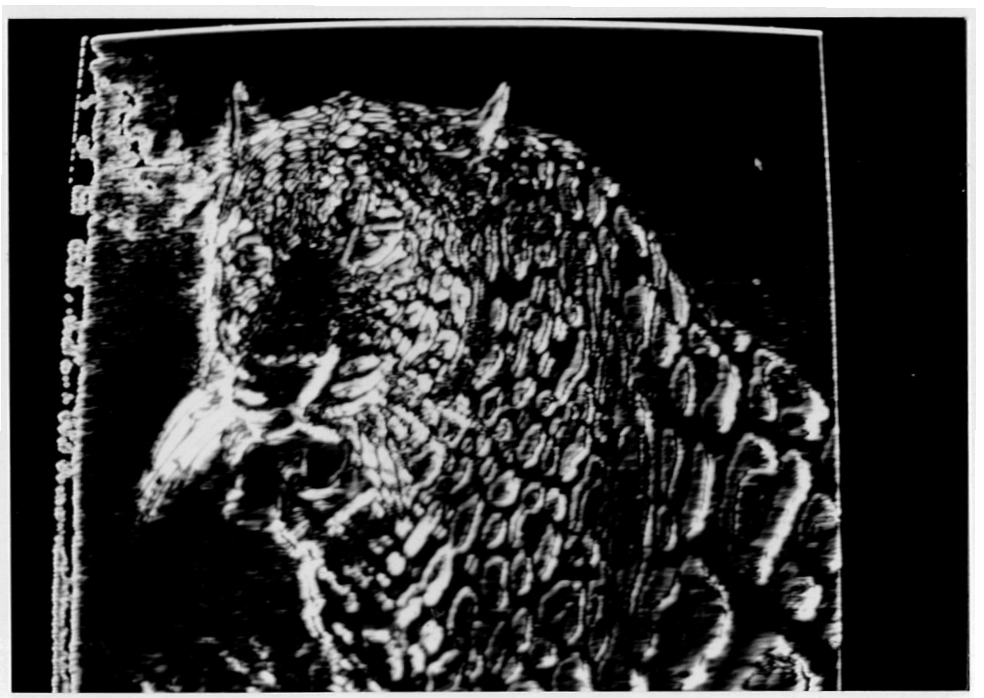


Figura V.17: A foto mostra o resultado da competição espacial com orientação de 0^0, cujos parâmetros são dados pelo teste 9

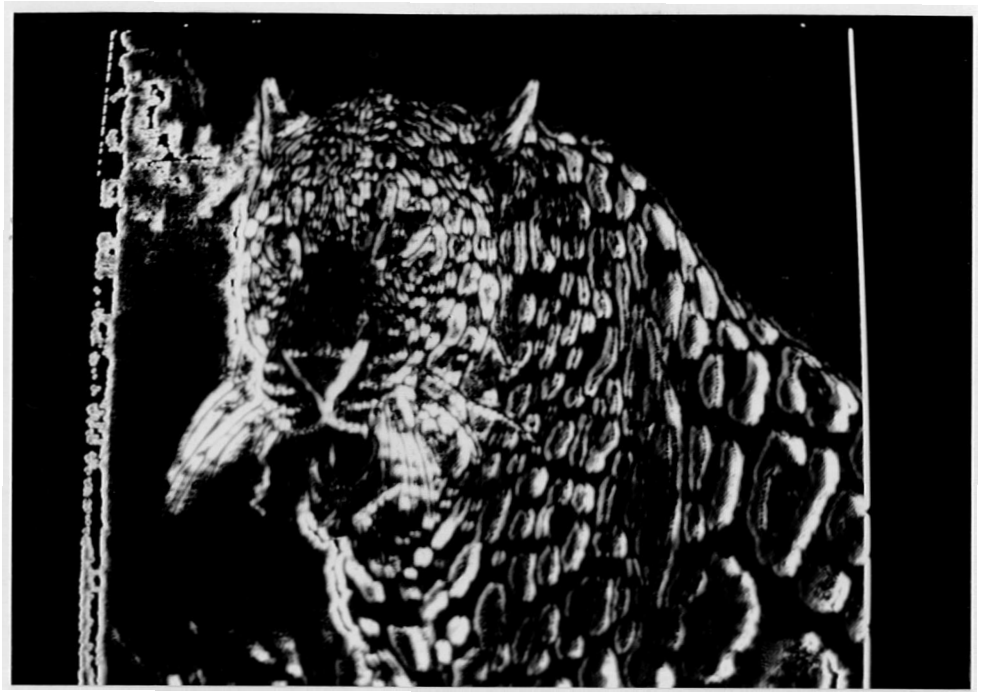


Figura V.18: A foto mostra o resultado da competição espacial com orientação de 45^0, cujos parâmetros são dados pelo teste 9

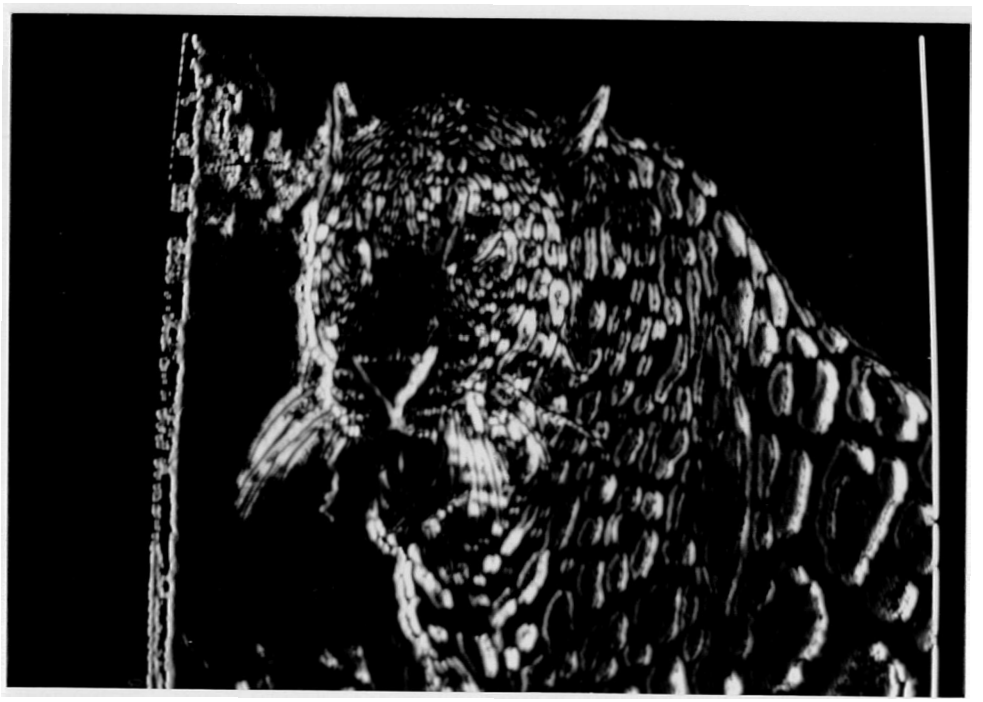


Figura V.19: A foto mostra o resultado da competição espacial com orientação de 90^0, cujos parâmetros são dados pelo teste 9

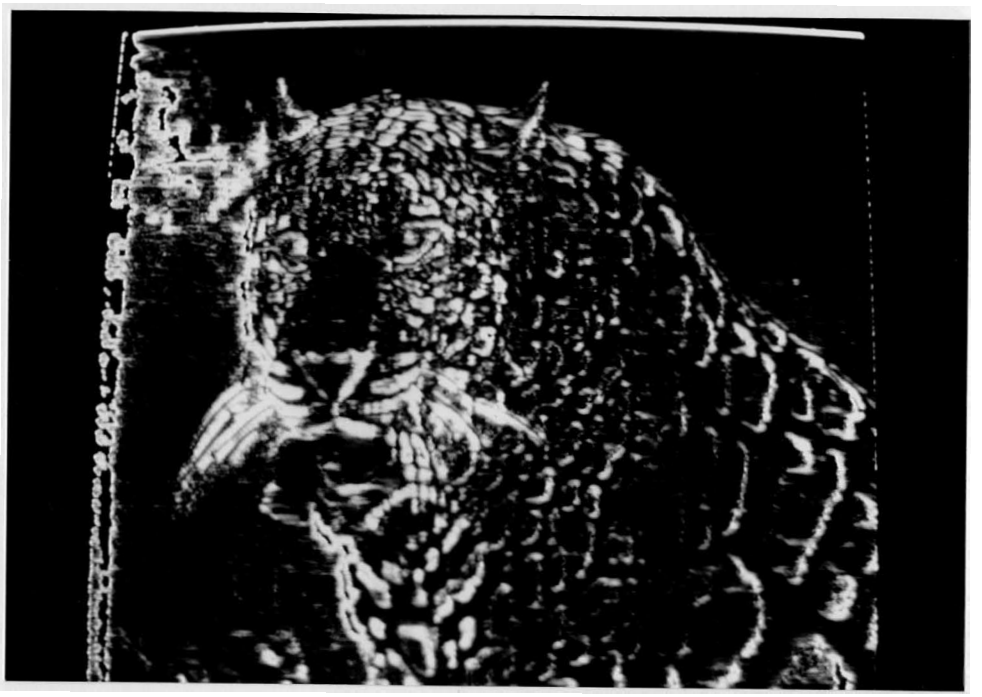


Figura V.20: A foto mostra o resultado da competição espacial com orientação de 135^0, cujos parâmetros são dados pelo teste 9

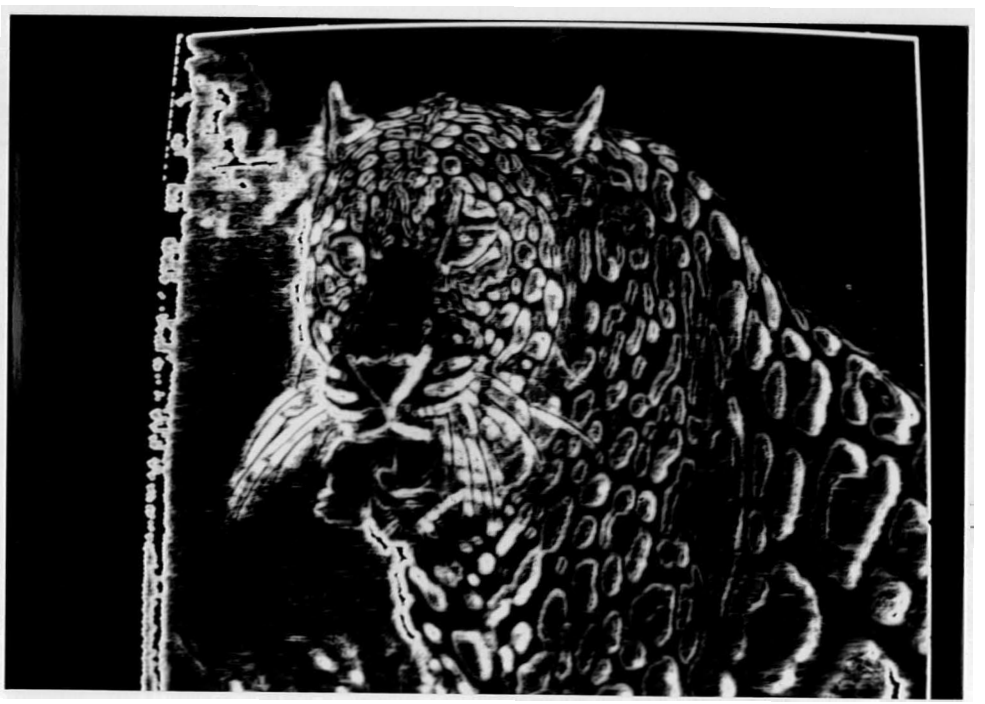


Figura V.21: A foto mostra o resultado da competição de orientação de *escala pequena*, cujos parâmetros são dados pelo teste 9

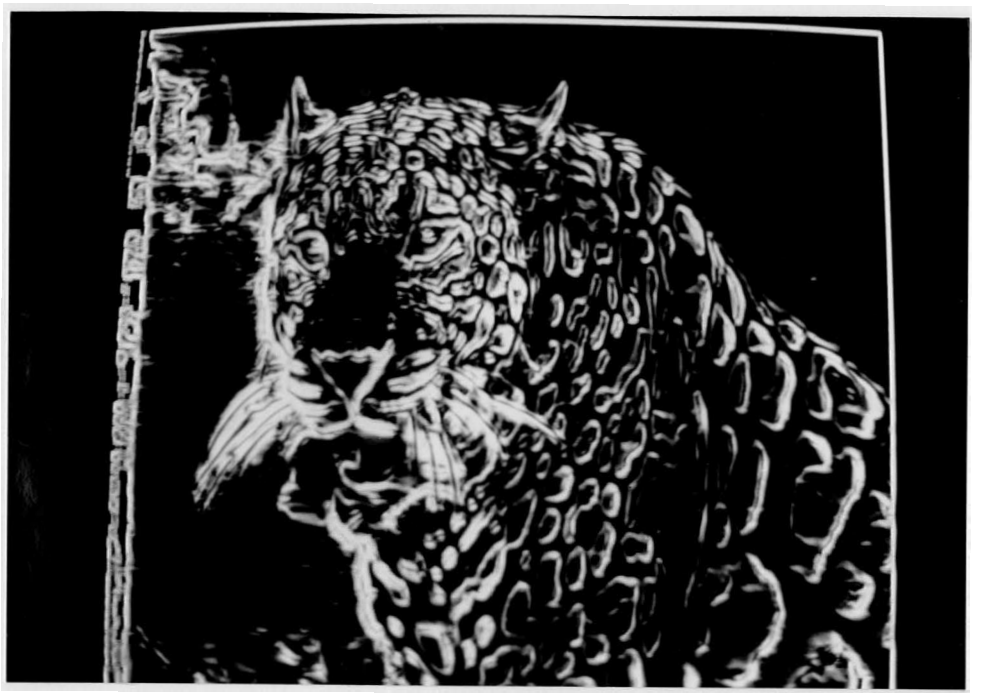


Figura V.22: A foto mostra o resultado da competição de orientação de *escala grande*, cujos parâmetros são dados pelo teste 9

escalas espaciais. A característica da escala pequena de detectar com mais precisão as curvas acentuadas nas bordas pode ser observada pela foto V.21 e a característica da escala grande de eliminar ruído com mais eficiência e completar borda com mais precisão é observada pela foto V.22. A interação das duas escalas é mostrada pela foto V.23. Percebe-se nas regiões marcadas na foto, conseguimos obter a detecção de curvas acentuadas e ao mesmo tempo eliminação de ruídos.

V.2.6 Cooperação Orientada

Nesssa seção, vamos testar a bipolaridade das células cooperativas para recuperar as bordas eliminadas pelos ruídos. A cooperação orientada deve explorar a escala grande devido à sua consistência das bordas detectadas. Os resultados obtidos são mostrados nas fotos V.24 V.25, V.26. Observamos que o parâmetro σ utilizado pelo teste 13 não obteve bons resultados, porque o valor limiar do parâmetro é muito grande em relação aos potenciais dos neurônios. No teste 11, diminuimos sensivelmente o valor do limiar para permitir que a pequena diferença entre dois polos possa recuperar as bordas. No entanto , esse limiar muito baixo pode recuperar ruído e

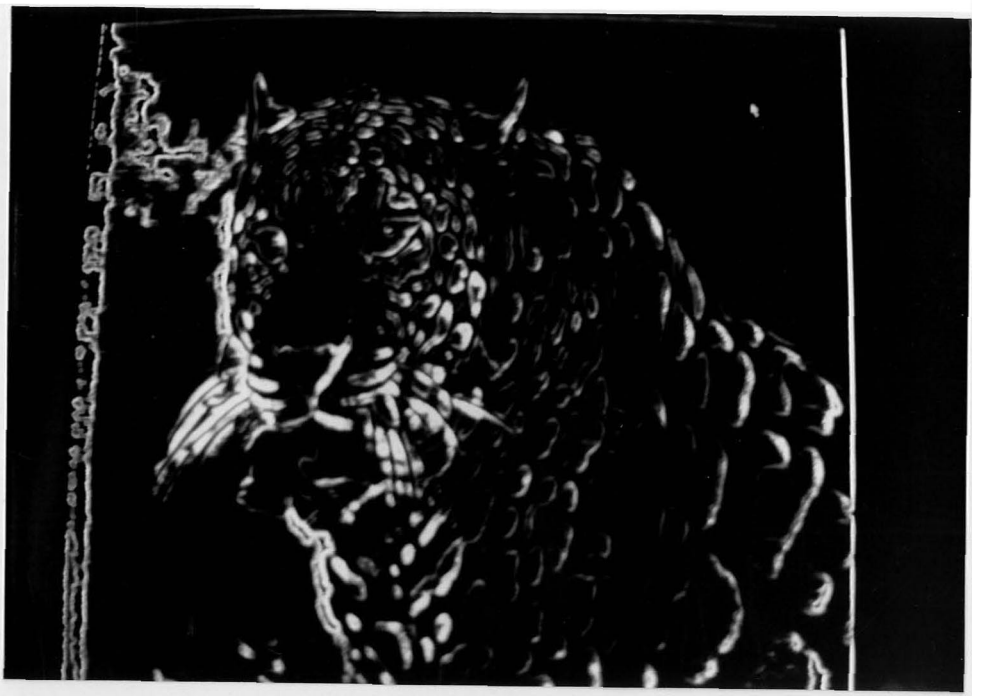


Figura V.23: A foto mostra o resultado da interação de múltiplas escalas espaciais

não bordas. Por isto o valor do parâmetro σ utilizado pelo teste 12, que é intermediário aos dois parâmetros anteriores, obteve resultado que assume o compromisso de recuperar bordas nítidas.

V.2.7 Resultado Final do Filtro CORT-X

Enfim, a foto V.27 mostra o resultado do filtro CORT-X de satifazer o compromisso entre localização de borda com curvaturas acentuadas, eliminação de ruídos perto e longe das bordas e completamentação das bordas eliminadas pelos ruídos.

Figura V.24: A foto mostra o resultado da cooperação orientada de acordo com parâmetros dados pelos teste 11

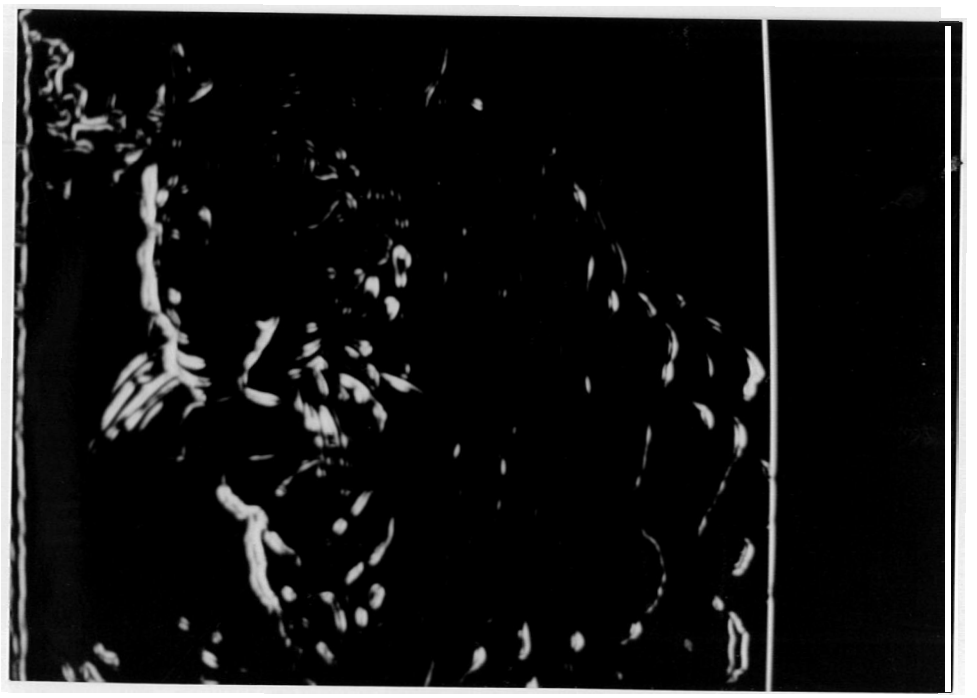


Figura V.25: A foto mostra o resultado da cooperação orientada de acordo com parâmetros dados pelos teste 12

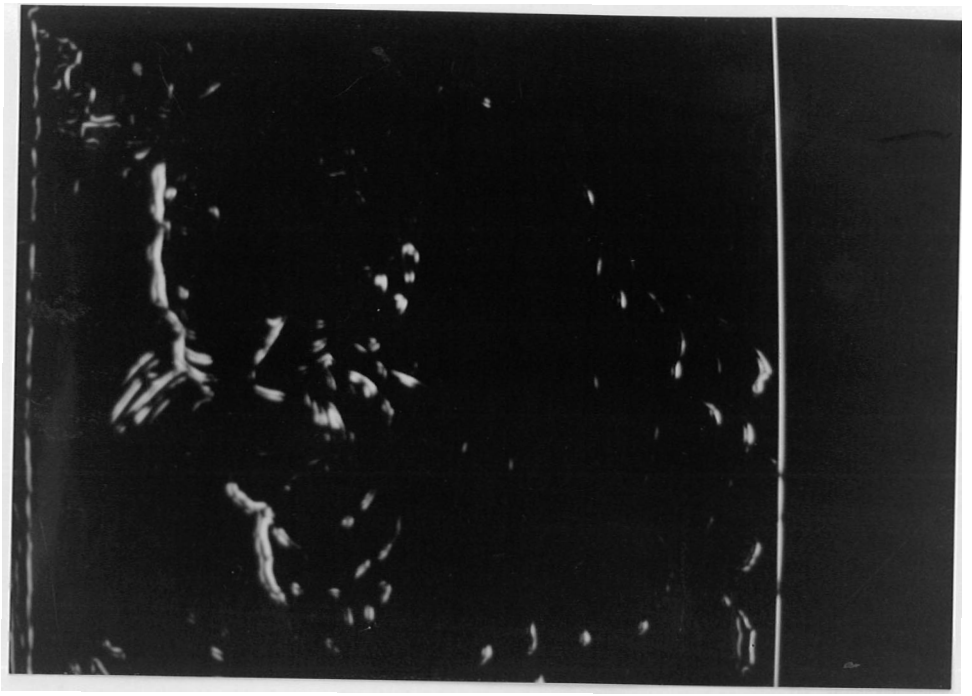


Figura V.26: A foto mostra o resultado da cooperação orientada de acordo com parâmetros dados pelos teste 13

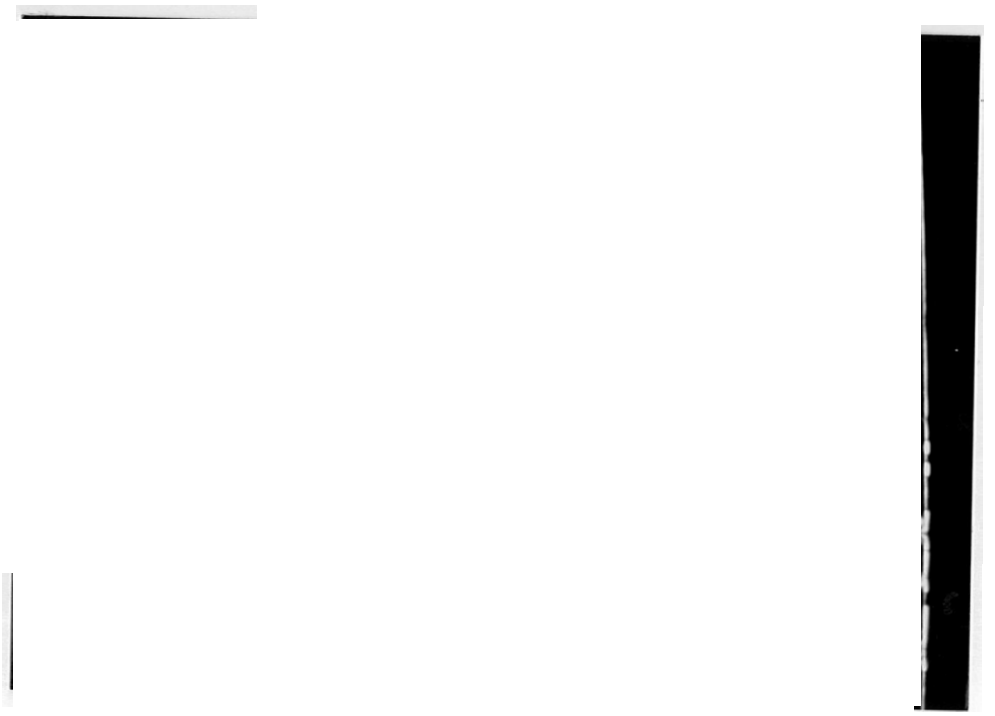


Figura V.27: A foto mostra o resultado final do filtro CORT-X

V.3 Resultados do Desempenho do Simulador Paralelo e Distribuído

Nesta seção, devemos analisar o desempenho da otimização da comunicação entre os processadores, da otimização da distância de comunicação entre os processadores, da topologia hipercubo retangular e a política de escalonamento de processo nos processadores.

Para isso, faremos a avaliação dos tempo obtidos pela simulação do filtro CORT-X em 2, 5 e 9 processadores, avaliação de comunicação entre os processadores através do desempenho dos neurônios situados na zona de fronteira, avaliação da influência de processamento dos neurônios situados na zona interna sobre o processamento dos neurônios situados na zona de fronteira e a avaliação do desempenho do processamento da união das duas zonas.

Depois disso faremos uma projeção do desempenho do simulador supondo um número razoável de *Transputers* disponíveis para simulação, e, finalmente, discutiremos a viabilidade de migração de processamento do T800 em conjunção com i860.

V.3.1 Avaliação dos Tempos Obtidos pela Simulação do Filtro CORT-X para 2,5 e 9 T800

O desempenho do simulador com variação de processadores simulados pode ser visto na tabela V.2, V.3 e V.5. A comparação da eficiência da simulação em 2 processadores e 5 processadores está na tabela V.7, percebemos que há sobrecarga de comunicação que varia de 35% a 44%. Essa sobrecarga já é esperado por causa da baixa velocidade de transmissão dos elos. Agora, a comparação da simulação de 5 processadores e 9 processadores não é muito animadora, praticamente não há ganho de desempenho. Tentaremos explicar alguns motivos que podem causar o baixo desempenho obtido, ver a tabela V.8. O primeiro motivo é devido à necessidade de enviar imagem de entrada a distâncias maiores. O segundo motivo pode ser visto pela tabela V.6, o leitor pode perceber que os números totais de neurônios situa-

Tempos medidos para simulação de 2 processadores com processamento de neurônios situados na zona interna			
n	ticks	segundos	ticks/neurônio
016x020x21	131.325	8,40	19,54
016x032x21	198.691	12,72	18,48
024x032x21	243.351	16,12	15,63
048x048x21	-	-	-
096x048x21	-	-	-
096x096x21	-	-	-
192x096x21	-	-	-
192x192x21	-	-	-
384x192x21	-	-	-
384x384x21	-	-	-
512x486x21	-	-	-

Tabela V.2: Tempos medidos pela simulação de 2 processadores com processamento de neurônios situados na zona interna

Tempos medidos para simulação de 5 processadores com processamento de neurônios situados na zona interna			
n	ticks	segundos	ticks/neurônio
016x020x21	48.040	3,07	7,14
016x032x21	67.178	4,29	6,24
024x032x21	87.603	5,60	5,43
048x048x21	180.230	11,53	3,72
096x048x21	304.510	19,48	3,14
096x096x21	533.130	34,12	2,75
192x096x21	-	-	-
192x192x21	-	-	-
384x192x21	-	-	-
384x384x21	-	-	-
512x486x21	-	-	-

Tabela V.3: Tempos medidos pela simulação de 5 processadores com processamento de neurônios situados na zona interna

Tempos medidos para simulação de 5 processadores sem processamento de neurônios situados na zona interna			
n	ticks	segundos	ticks/neurônio
016x020x21	47.914	3,06	7,13
016x032x21	74.100	4,74	6,89
024x032x21	91.659	5,86	5,68
048x048x21	178.452	11,42	3,68
096x048x21	280.703	17,96	2,90
096x096x21	428.459	24,42	2,21
192x096x21	-	-	-
192x192x21	-	-	-
384x192x21	-	-	-
384x384x21	-	-	-
512x486x21	-	-	-

Tabela V.4: Tempos medidos pela simulação de 5 processadores sem processamento de neurônios situados na zona interna

Tempos medidos para simulação de 9 processadores com processamento de neurônios situados na zona interna			
n	ticks	segundos	ticks/neurônio
048x048x21	195.760	12,51	4,04
096x048x21	289.997	18,56	2,99
096x096x21	523.355	33,49	2,70
192x096x21	809.225	51,79	2,09
192x192x21	-	-	-
384x192x21	-	-	-
384x384x21	-	-	-
512x486x21	-	-	-

Tabela V.5: Tempos medidos pela simulação de 9 processadores com processamento de neurônios situados na zona interna

Comparação de números de neurônios situados na zona de fronteira entre 5 e 9 processadores			
n	5 processadores	9 processadores	relaçao
$xxyxd$	n_{5R_f}	n_{9R_f}	n_{9R_f}/n_{5R_f}
048x048x21	1280	1792	1,40
096x048x21	2048	2560	1,12
096x096x21	2816	4096	1,45
192x096x21	4352	5632	1,29
192x192x21	5888	8704	1,47
384x192x21	8960	11776	1,31
384x384x21	12032	17920	1,48
512x486x21	15712	23232	1,47

Tabela V.6: Comparação de números de neurônios situados na zona de fronteira enter 5 e 9 processadores

dos na zona de fronteira na topologia retangular de 9 processadores são maiores do que os neurônios de 5 processadores situados na mesma zona, a taxa de aumento, que varia de 25% a 49% de *overhead* de comunicação e de sincronização. O terceiro motivo está no número dobrado de neurônios situados na zona de fronteira que requerem distânica maior para as mensagem percorrerem. O último motivo é a ordem pela qual o processo mestre envia mensagens de sincroniza, ao para processo servidor, o mestre deve procura sincronização os processos servidores vizinhos para evitar tempo de espera e memória para armazenar as mensagens de processamento no buffer. Achamos que os quatro motivos juntos são capazes de justificar os maus tempos obtido na simulação do filtro CORT-X em 9 processadores. Aproveitando o exemplo, queremos ressaltar que a baixa granularidade da topologia retangular não permite a avaliação do desempenho de otimização de distância de comunicação entre os processadores.

V.3.2 A Avaliação da Eficiência de Comunicação entre Os Processadores

A avaliação de eficiência de comunicação entre os processadores pode ser feita através dos tempos medidos pela simulação dos neurônios situados na zona de fronteira R_f e neurônios situados nas duas zonas, R_f e R_i(zona interna). As medidas são mostradas pelas tabelas V.4 e V.3. Cinco processadores são usados para simulação do filtro

Comparação de tempos medidos entre 2 e 5 processadores			
n	2 processador	5 processadores	t_5/t_4
$xxyxd$	ticks	ticks	
016x020x21	133.080	48.040	2,77
016x032x21	198.388	67.178	2,95
024x032x21	252.004	87.603	2,87
192x096x21	-	-	-
192x192x21	-	-	-
384x192x21	-	-	-
384x384x21	-	-	-
512x486x21	-	-	-

Tabela V.7: Comparação de tempos medidos entre simulação de 2 processador e simulação de 5 processadores

Comparação de tempos medidos entre 5 e 9 processadores			
n	5 processadores	9 processadores	t_9/t_5
$xxyxd$	ticks	ticks	
048x048x21	180.230	195.760	0,92
096x048x21	304.510	289.997	1,05
096x096x21	533.130	523.355	1,01
192x096x21	-	-	-
192x192x21	-	-	-
384x192x21	-	-	-
384x384x21	-	-	-
512x486x21	-	-	-

Tabela V.8: Comparação de tempos medidos entre simulação de 5 processadores e simulação de 9 processadores

n	nR_i	t_{R_i}	n_{R_f}	t_{R_f}	n_{R_i}/n_{R_f}	t_{R_i}
016x020x21	0	0,00	76	7,50	0,0	7,14
016x032x21	0	0,00	128	6,89	0,0	6,24
024x032x21	32	0,00	160	6,81	0,2	5,43
048x048x21	256	0,08	320	6,63	0,8	3,72
096x048x21	640	0,44	512	6,52	1,25	3,14
096x096x21	1600	0,77	704	7,24	2,27	2,75
192x096x21	3520	-	1088	-	3,23	-
192x192x21	7744	-	1472	-	5,23	-
384x192x21	16192	-	2240	-	7,22	-
384x384x21	33856	-	3008	-	11,25	-
512x486x21	58280	-	3928	-	14,83	-

Tabela V.9: Comparação de tempos medidos entre zona de fronteira e zona interna para 5 processadores

CORT-X, o processo mestre é mapeado no nó raiz e os quatros processos escravos são mapeados nos outros processadores. Para medir o desempenho de processamentos dos neurônios situados na zona de fronteira, nós medimos os tempos sem o processamento de neurônios situados na zona interna t_{R_f}, Em seguida, dividimos os tempos obtidos pelos números de neurônios situadados na zona de fronteira n_{R_f}. Assim, obtemos o tempo médio de processamento desses neurônios. A tabela V.9 mostra os tempos obtidos para diferentes números de neurônios simulados para o filtro. Nota-se, que só há pequenas variações dos tempos t_{R_f} que oscilam em torno de 6,5 a 7,5 *ticks*. As pequenas variações são indicadores de que os tempos obtidos seja a capacidade de comunicação dos *links* do *Transputer*, ou seja, o *Transputer* consegue dar suporte de comunicação a um neurônio em torna de 7.0 *ticks*.

V.3.3 Avaliação da Influência do Processamento de Neurônios Situados na Zona Interna

Na tabela V.9, temos os tempos aparentes de processamento de neurônios situados na zona interna R_i para números crescentes de neurônios. Os tempos são aparentes porque eles não medem o tempo real de processamento desses neurônios e sim, a influência que esses neurônios exercem sobre o tempo total da simulação.

O leitor deve estranhar os tempos obtidos nas linhas 2 e 3 da tabela V.3, devido ao fato de que os tempos medidos para simulação todos os neurônios do

filtro serem menores que os tempos medidos para a simulação de neurônios situados na zona de fronteira. Devemos lembrar que o *Transputer* faz simultaneamente a comunição pelos elos e o processamento interno. Se o tempo de processamento de neurônios situados na zona interna for menor que o tempo de comunicação nos elos, então o cálculo de tempo total da simulação do filtro não refletirá este tempo, por isto os tempos obtidos só dependem do processamento de neurônios situadas na zona de fronteira.

A relação entre número de neurônios situados na zona interna e o número de neurônios situados na zona de fronteira realça a afirmação feita, já que o número de neurônios situados na zona interna é muito menor nestes casos e o tempo de processamento destes neurônios sempre será menor porque eles não possuem sobrecarga de comunicação.

V.3.4 Avaliação do desempenho do Simulador

Outra questão que o leitor pode ter : 'Como os tempos médios medidos de processamento dos neurônios diminuem em função ao número crescentes de neurônios do filtro, se ambos os tempos t_{R_f} e t_{R_i} não diminuem?"

A resposta para esse questão pode ser vista na própria tabela. Nota-se que os tempos aparentes t_{R_i} são muito menores que os tempos t_{R_f}, e os números n_{R_i} vão tornando-se maiores do que os números n_{R_f}, de acordo com o aumento de número de neurônios simulados esta característica pode ser vista pela relação n_{R_i}/n_{R_f}. Com isso, o cálculo total de tempo médio de processamento de neurônios é a média ponderada dos tempos t_{R_i} e t_{R_f}. Esse tempo médio sempre estará no intervalo entre os dois tempos extremos,t_{R_i}(melhor tempo) e $_{R_f}$(pior tempo).

Esse resultado justifica a escolha da topologia hipercubo retangular que permite o agrupamento de neurônios na zona de fronteira e na zona interna. Possibilitando assim, uma eficiência de processamento de neurônios que pode ser superior a três vezes (ver a mesma tabela). O tempo médio de processamento dos neurônios vai de $7,149$ *ticks* a $2,755$ *ticks* com apenas $3,7\%$ de neurônios processados.

Esperamos que esse tempo atinja um tempo bom em torno de 1 *tick*, porque o valor alto da relação n_{R_i}/n_{R_f}, em torno de 14,837, indica que estámos mais próximo do t_{R_i} (melhor tempo). O tempo ótimo possibilita processamento de 15.625 neurônios por segundo, ou seja, o simulador leva 334.43 segundos para processar 5,225 milhões de neurônios do filtro CORT-X de uma imagem digitalizada de uma camâra CCD. Do ponto de vista de processamento em tempo real, ainda estamos muito longe de alcancá-lo com apenas cinco Transputers porque uma camâra CCD captura 30 quadros por segundo, então teremos que processar a imagem com desempenho 10032,9 vezes mais rápido.

V.3.5 Idealizando Número Ilimitado de T800 Disponível

De uma forma muito otimista, supondo que temos uma rede de ilimitada de *Transputers* disponíveis para a simulação, então o tempo de processamento de neurônios situados na zona interna torna-se insignificante em relação ao tempo de processamento de neurônios situados na zona de fronteira devido à alta granularidade. O simulador levaria em torno de 0,766 segundos para processar uma imagem em tempo real devido à sobrecarga de comunicação dos quatro elos do *Transputer*, mesmo com número ilimitado de *Transputers*. Mas se o *Transputer* tiver mais elos de comunicação (tecnicamente possível), então, nós podemos mapear outras dimensões do filtro CORT-X nestes elos. Com isso, achamos que podemos simular o filtro CORT-X em tempo real e até mesmo os sistemas BCS e FCS, apesar da necessidade de técnicas númericas como Runge-Kutta para resolver as equações diferenciais. É claro que nestes casos precisamos balancear a relação n_{R_i}/n_{R_f} (tempo de processamento e tempo de comunicação) para aproveitar melhor o desempenho dos *Transputers*.

V.3.6 Alguns Aspectos da Migração do Simulador para o i860

Sob aspecto da migração do simulador para o i860, devemos ressaltar a importância de explorar as capacidades de processamento paralelo e *pipeline*, porque a pastilha não é muito melhor do que i460 no modo escalar. Para isso, precisa-se trabalhar

em harmonia com os dois modos paralelos do i860, modo dual de operação e modo dual de instrução. Não vamos entrar em detalhes, por hora, esclarecemos como a migração pode ser feita.

Com a divisão de localizações de neurônios em zonas de fronteira e zona interna, então temos duas fontes de dados. Os neurônios situados na zona de fronteira podem comunicar-se com os nós vizinhos através de memórias bipolares de interface que fazem a comunicação entre *Transputer* e i860. Com auxílio da interface, quando T800 receber uma mensagem de nós vizinhos, ele a coloca na mémoria e avisa ao i860 a chegada de nova mensagem. Para explorar o paralelismo do i860, pode-se usar o modo dual de instrução para processar a fonte dual de dados, deixando assim, o controle dos fluxos de dados em explorar o modo dual de operação do i860. Consequentemente, o *Tranputer* só fica encarregado de prover comunicações entre os nós do NCPI.

Respondendo a pergunta feita na introdução da seção: "*Transputer* tem desempenho suficiente para dar suporte de comunicação entre os nós do NCPI?". Não temos condições de provar, mas damos indícios da eficiência de comunicação do *Transputer*. Se o *Transputer* só fica encarregado de receber mensagens dos elos e colocá-las no *buffer* de entrada e enviar mensagens armazenadas no *buffer* de saída então achamos que a resposta é afirmativa para simulação de filtro CORT-X e principalmente para os sistemas BCS e FCS, pois este requerem método número para resolver equações diferenciais.

Segundo nosssos cálculos o filtro CORT-X possui em torno de 80 milhões de operações de aritméticas a serem processadas, enquanto i860 possui performance de 80 *mflops*. Logo, i860 levaria 1s para processar o filtro. Por outro lado, o filtro possui 76 milhões de conexões, enquanto o *Transputer* conseque tranferir 5 megabytes por segundo. Portanto o T800 levaria 15 segundos para fazer as comunicações entre os neurônios, o que é 15 vezes mais lento que o i860. No entanto, com o mapeamento do filtro CORT-X na topologia hipercubo retangular de quatro nós, o *Transputer* só precisar fazer 1200 mil comunicações, baixando o tempo de comunicação para 0.2 segundos. Por outro lado com aumento de nós, os quatros i860 levaria 0.25 segundos. Enfim, os tempos obtidos permitem dar indícios

da viabilidade de suporte de comunicação por parte do *Transputer* ao i860.

Capítulo VI

Conclusões e Perspectivas Futuras

Nesta tese, procuramos mostrar a importância de alguns cuidados essenciais que devem ser considerados quando se deseja resolver problemas difíceis através de redes neuronais. A primeira precaucão é definir as regiões de vizinhança das células. Uma exemplo pode ser visto pelo nível da máscara orientada, as sensibilidades das orientações dos neurônios depende unicamente da definição do campo receptivo, ou seja, a região de vizinhança.

A segunda precaucão é definir as influências pelas quais as células se exercem mutuamente. No nível de cooperação orientada, vimos um exemplo de influência dos dois polos do campo receptivo do neurônio sobre ele mesmo para completar as bordas.

A terceira precausão é definir grupos de células com funções especializadas, cada um destes grupos só é capaz de resolver um subproblema em particular. No entanto as interações entre esses grupos permite a resolução global do problema. Vimos um exemplo deste tipo de filosofia de resolver problemas na interaçao dos sistemas BCS e FCS, onde só através da interação dos dois sistemas pode-se resolver as incertezas causadas dentro de cada sistema.

A quarta precaução é definir a equação diferencial que rege o comportamento de cada neurônio. O comportamento do neurônio é derivado das três primeiras precauções.

Queremos ressaltar que apesar das equações matemáticas do filtro

CORT-X serem simples, compostos de operações elementares tais como: soma, subtração, multiplicação e divisão, elas são capazes de resolver problemas complexos tais como : localização de borda, eliminação de ruído e complemento de borda. Este bons resultados foram obtidos graças às precaucões tomadas, caso contrários, teriamos que usar equação mais elaboradas.

Do ponto de vista mais global, o filtro CORT-X seria um preprocessador que envia informações de bordas de uma imagem para a etapa posterior. Essa etapa usa filtro *log-polar-Fourier* [8] [9] [10] [33] no sentido de auxiliar a próxima etapa em obter reconhecimento de objeto invariante à escala, à rotação e à translação. O processo de reconhecimento é feito pelas redes neuronais, ART2 [4] [5] e ART3 [2]. As arquiteturas da ART2 e ART3 são capazes de auto-organização estável de categorias de reconhecimento, ou seja, elas podem automaticamente aprender códigos de reconhecimento em resposta à ordem arbitrária dos padrões de entradas escolhido arbitriamente. As arquiteturas possuem parâmetro de vigilância que controla o grau de deformação aceitável na forma do objeto. Essa deformação é causado pelo filtro *log-polar-Fourier* no sentido de obter a invariância.

VI.1 Comentários sobre Resultados Obtidos pela Simulação do Filtro CORT-X

Sob aspectos dos resultados obtidos pela simulação do filtro CORT-X, podemos dizer que estamos satisfeitos. O compromisso de localização de borda, eliminação de ruído e complemento de borda é mostrado através de fotos. Os efeitos dos mecanismos de máscara orientada, competição espacial, competição de orientação, interação de múltiplas escalas espaciais e cooperação orientadas foram discutidos em detalhes e mostrados nas fotos também.

Os ajustes de parâmetros foram feitos para visualizar as influências destes parâmetros sobre as equações. Sugerimos auto-ajuste de parâmetros para trabalhos futuros. A idéia é baseada na interação de dois subsistemas de resolução de incertezas. O primeiro sistema possui a incerteza da qualidade dos valores usados nos parâmetros. Para compensar essa incerteza, o segundo subsistema avalia os

resultados obtidos pelo primeiro subsistema que dá indicativo de como o este sistema deve variar o paraâmetro na próxima iteração. Assim novos resultados será obtidos até que os valores desejados dos parâmetros. A outra maneira é deixar o próprio usuário fazer os ajustes através do ambiente de interface de *windows*.

Nós também sugerimos a auto-escalação do tamanho do campo receptivo. A vantagem da auto-escalação está na capacidade de variar as escalas de acordo com a imagem processada. Por exemplo, nas curvas acentuadas, a auto-escalação deve diminuir a escala (tamanho do campo receptivo) para detectar melhor esta curva. Mas como os neurônios podem saber a existência da curva acentuada? Um solução é usar potenciais vindos da céluas vizinhas, se as diferênças entre os potenciais dos polos do campo receptivo são elevadas, então é muito provável que haja uma curva. Com isto, as diferênças devem incentivar a diminuição de escala. Por outro lado, se as atividades das céluas vizinhas forem baixas, então elas devem incentivar o aumento da escala porque a probabilidade da região da imagem ser ruidosa é maior, sabendo que, os ruídos podem ser facilmente eliminados pelo filtro de escala grande.

VI.2 Aspectos Teóricos da Simulação Paralela e Distribuída

As idéias apresentadas para a simulação paralela e distribuída são válidas para uma séries de aplicações que usam redes neuronais do tipo *On-Center-Off-Surround*. Essas aplicações incluem processamento da fala, reconhecimento de padrão temporal, fazer decisão com risco, conexões neurobiológica, condicionamento clássico, diagnósticos, processamento de conhecimento, robótica, controle de ritmos cardiáco e outros.

A escolha de topologia hipercubo retangular permitiu uma otimização de comunicação entre os processadores e uma excelente otimização de distância de comunicação entre os processadores. De uma forma mais clara, a distância máxima é dois. A topologia também permite agrupamento de neurônios situados na zona de fronteira e na zona interna (não possui comunicação externa ao processador).

As otimizações e o agrupamento facilitam a migração da implementação do simulador em T800 para implementação de um simulador estendido que utiliza o recurso de comunicação do T800 e os modos paralelos do i860 ou seja, o modo dual de operação e o modo dual de instrução. Sem dúvida, um trabalho interessante para pesquisa futura é mostrar como o processamento paralelo e distribuído das redes neuronais do tipo *On-Center-Off-Surround* pode ser muito mais eficiente do que processamento serial ou paralelo. A eficiência mencionada é oriunde das trocas de informações que permitem interações entre os subsistemas. Essas trocas permitem resolver o problema com poucas iterações.

VI.3 Avaliação do Desempenho do Simulador Paralelo e Distribuído

Nós avaliamos o desempenho do simulador segundo os aspectos de números de processadores disponíveis e a topologia hipercubo retangular utilizada. A comparação da eficiência da simulação entre 2 processadores e 5 processadores obteve 35% a 44% de sobrecarga devido ao tempo de comunicações nos elos e ao tempo de sincronização de mudança de níveis do filtro CORT-X. Percebemos que a ordem pela qual o processo mestre envia mensagens de sincronização aos processos escravos é um fator determinante do desempenho do simulador. Os tempos medidos mostram que o melhor caso pode atingir 200% mais eficiencia do que o pior caso. Por isso devemos tomar muito cuidado com a ordem de envios de mensagens de sincronização do processo mestre ao processo escravo. Sugerimos para trabalhos futuros usar método de sincronização por difusão, ou seja, o processo mestre envia uma mensagem de controle ao processo escravo situado na região central da topologia, em seguida, o próprio processo escravo fica encarregado de difundir a mensagem de controle aos outros processos escravos da topologia. Desta forma, o processo mestre teria participação única na sincronização, já que ele só precisa enviar uma mensagem de sincronização. Geralmente, o processo mestre está situado na parte periférica da rede, a redução de número de envios de mensagem de sincronização otimiza a distância percorrida pelas mensagens de controle. Mais ainda, o processo de difusão permite que os processos escravos receber mensagens de controle muito rapidamente porque

quando os processos escravos receberem mensagens de controle, eles já sabem que os processos escravos que mandaram as mensagens de controle estão prontos para interagirem.

Com a adoção de uso de *buffers* internos, alguns cuidados devem ser tomadas para evitar possível bloqueio perpétuo. O bloqueio perpétuo é causada pelo não recebimento de mensagens de controle devido ao estouro do *buffer* de processamento. Por isso, um método eficiente de sincronização permite a redução do tamanho do buffer necessário para evitar bloqueio perpétuo, além de aumentar o desempenho do simulador. Outro uso de *buffer* de entrada para cada elo é permitir o escoamento mais rápido das mensagens em trânsito.

Na avaliação do desempenho da topologia hipercubo retangular, vimos que o agrupamento e neurônios em zona de fronteira e em zona interna possibilita uma eficiência de processamento de neurônios três vezes superior. De forma mais detalhada, vimos que o processamento dos neurônios situados na zona interna compensa o baixo desempenho do processamento dos neurônios situados na zona de fronteira. Mostramos também que o ganho de tempo comutacional depende exclusivamente da relação n_{R_i}/n_{R_f}. No entanto, não conseguimos avaliar o desempenho da topologia hipercubo retangular devido à baixa granularidade da rede de *Tranputers*. De uma maneira mais clara, não consequimos aproveitar a vantagem da teoria de otimização de distância de comunicação entre os processadores.

VI.4 Trabalhos Futuros

Iniciamos este capítulo dando importância às precauções que devem ser tomadas quando se deseja resolver um problema através de redes neuronais, e terminamos o capítulo dando sugestões para pesquisas futuras para entender melhor essas precauções. As seguintes sugestões permitem vizualizar a importância das precauções em cinco questões básicas: i)quem são os neurônios vizinhos?; ii)como eles se influenciam?; iii)quais são os grupos funcionais?; vi)como os grupos interagem para resolver globalmente o problema?; e v) qual é o comportamento dos neurônios?

Os trabalhos [34] [35] mostraram que mesmo em rede neuronais simples, ela podem ter comportamento caótico. Por isso, achamos essencial prover meios para visualização do comportamento das redes neuronais. Para isso estamos elaborando um sistema de visualização do comportamento das redes neuronais chamado de *Neural Networks Behavior Visualization.*

Nesse sistema procuraremos vizualizar vários escopos do comportamento das redes neuronais tais como: o comportamento do neurônio, topologia de vizinhança local, topologia do grupo, topologia da rede neuronal, a influência local, interação dos grupos, e principalmente difusão dos potenciais emitidos pelos neurônios. Cada potencial emitido é visualizado por uma partícula que se move entre os neurônios, em substituição da tradicional visualização das conexões fixas dos neurônios. Essa partícula possui atribuidos de cor, tamanho, duração, velocidade, aceleração e trajetoria que vão permitir flexibilidade de vizualizar alteração dinâmica da topologia da rede. O neurônio teria centro receptor de partículas para captar as partículas que chegam e centro emissor de partículas para emitir as partíulas.

Emfim, esperamos que o sistema de visualização possa nos ajudar a entender melhor o comportamento dinâmico ou caóticos das redes neuronais. Contribuindo assim, novos métodos de resolver problemas.

Referências Bibliográficas

[1] Alan BURNS. *Programming in Occam 2.* Addison-Wesley Publishing Company, 1988.

[2] G.A. CARPENTER and S. GROSSBERG. Art 3: hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Network*, 3(2):129–152, 1990.

[3] G.A. CARPENTER and S. GROSSBERG. The art of adaptive pattern recognition by a self-organization neural network. *IEEE Computer*, 21:77–88, March 1988.

[4] G.A. CARPENTER and S. GROSSBERG. Art2: self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, 26(23):4919–4930, December 1987.

[5] G.A. CARPENTER and S. GROSSBERG. Art2: stable self-organization of pattern recognition codes for analog input patterns. In Caudill M. and C. Butler, editors, *Proceedings First Internacional Conference on Neural Networks*, pages 727–736, IEEE San Diego, IEEE, New York, June 1987.

[6] G.A. CARPENTER and S. GROSSBERG. Invariant recogintion of cluttered scenes by a self-organizing art architecture: cort-x boundary segmentation. *Neural Network*, 2(3):169–181, 1989.

[7] G.A. CARPENTER, GROSSBERG S., and S. MEHANIAN. Self-organization of invariant recognition categories for noisy images by an adaptive resonance theory (art) architecture. *Neural Networks*, 1(Suppl. 1):15, 1988.

[8] D. Casasent and D. Psaltis. Position, rotation, and scale invariants optical correlations. *Applied Optics*, 15:1793–1799, 1976.

[9] P. Cavanagh. Image transforms in the visual system. In P.C. Dodwell and T. Caelli, editors, *Figural Synthesis*, pages 185–218, Erlbaum, 1984.

[10] P. Cavanagh. Size and position invariance in the visual system. *Perception*, 7:167–177, 1978.

[11] M.A. COHEN. The stability of sustained oscillations in a symmetric cooperative-competitive neural networks: a disproof of a conjecture about content addressable memory. *Neural Networks*, 1:217–221, 1988.

[12] M.A. COHEN. The stability of sustained oscillations in a symmetric cooperative-competitive neural networks. *Neural Networks*, 3(6):609–612, 1990.

[13] M.A. COHEN and S. GROSSBERG. Neural dynamics of brightness perception: features, boundaries diffusion, and resonance. *Perception and Psychophysics*, 36():428–456, 1984.

[14] M.A. COHEN and S. GROSSBERG. Sustained oscillations in low dimensional distance dependent on-center off-surround neural networks. *Neural Networks*, 1(1):, 1988.

[15] Anne DINNING. A survey of synchronization methods for parallel computer. *IEEE Computer*, 22(7):66–77, July 1989.

[16] Valmir C. Barbosa e Priscila M. V. Lima. *On The Distributed Parallel Simulation of Hopfield's Neural Networks.* Relatório Técnico do Programa de Engenharia de Sistemas e Computação ES-174/88, COPPE/UFRJ, Novembro 1988.

[17] Stephen S. Fried. Personal supercomputing with the intel i860. *BYTE*, 16(2):347–358, January 1991.

[18] Stephen GROSSBERG. *The Adaptive Brain, Vol II : Vision, Speech, Language, and Motor Control.* Volume of , Elsevier/North-Holland, Amsterdam, edition, 1987.

[19] Stephen GROSSBERG. Neural dynamics of brightness perception: features, boundaries, diffusion, and resonance. *Perception and Psychophysics*, 36(5):428–456, 1984.

[20] Stephen GROSSBERG. Neural dynamics of form perception: boundary completion, illusory figures, and neon color spreading. *Psychological Review*, 92(2):173–211, 1985.

[21] Stephen GROSSBERG. Neural dynamics of three-dimensional form, color, and brightness perception, i: monocular theory. *Perception and Psychophysics*, 41(2):87–116, 1987.

[22] Stephen GROSSBERG. Neural dynamics of three-dimensional form, color, and brightness perception, ii: binocular theory. *Perception and Psychophysics*, 41(2):117–158, 1987.

[23] Stephen GROSSBERG. Neural networks for visual perception in variable illumination. *Optics News*, 14(1):5–10, 1988.

[24] Stephen GROSSBERG and J.A. Marshall. A computational model of how cortical complex cells multiplex information about position, contrast, orientation, spatial frequency, and disparity. In Caudill M. and C. Butler, editors, *Proceedings First Internacional Conference on Neural Networks*, pages VI203–214, IEEE San Diego, IEEE, New York, June 1987.

[25] Stephen GROSSBERG and J.A. Marshall. A model cortical architecture for the preattentive perception of 3-d form. In E.L. Schwartz, editor, *Computational Neuroscience*, THe MIT Press, 1988.

[26] Stephen GROSSBERG and J.A. Marshall. Stereo boundary fusion by cortical complex cells: a system of maps, filters, and feedback networks for multiplexing distributed data. *Neural Networks*, 2(1):29–51, 1989.

[27] Stephen GROSSBERG and Ennio Mingolla. Neural dynamics of perceptual grouping: textures, boundary, and emergent segmentations. *Perception and Psychophysics*, 38(2):141–171, 1985.

[28] Stephen GROSSBERG and Ennio Mingolla. Neural dynamics of surface perception: bounday webs, illuminants, and shape-from-shading. *Computer Vision, Graphics, and Image Processing*, 37:116–165, 1987.

[29] Stephen GROSSBERG and Ennio. Mingolla. A neural network architecture for preattentive vision: multiple scale segmentation and regularization. In Caudill M. and C. Butler, editors, *Proceedings First Internacional Conference on Neural Networks*, pages VI177–184, IEEE San Diego, IEEE, New York, June 1987.

[30] Stephen GROSSBERG and Dejan Todorovic. Neural dynamics of 1-d e 2-d brightness perception: a unified model of classical and recent phenomena. *Perception and Psychophysics*, 43:241–277, 1988.

[31] Frank HAYES. Intel's cray-on-a-chip. *BYTE*, 14(6):113–114, May 1989.

[32] W. Daniel Hillis. *The Connection Machine.* The Mit Press, Cambridge, Massachusetts, 1985.

[33] H. Szu. Three layers of vector output product neural networks for optical pattern recognition. In H. Szu, editor, *Optical and Hybrid Computing*, pages 312–330, SPIE, 1986.

[34] Han L. J. van der Maas, Verschure P.M.J., and P. C. M. Molenaar. A note on chaotic behavior in simple neural networks. *Neural Networks*, 3(1):119–120, 1990.

[35] Yong Yao and Walter J. Freeman. Model of biological pattern recognition with spatially chaotic dynamics. *Neural Networks*, 3(2):153–171, 1990.