Projeto de Operadores Morfológicos para Imagens e Sinais

Abordagem de reticulados finitos discretos

Roberto Hirata Jr.

TESE APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO

PARA OBTENÇÃO DO GRAU DE DOUTOR
EM
CIÊNCIA DA COMPUTAÇÃO

Área de Concentração : **Ciência da Computação**
Orientador : **Prof. Dr. Junior Barrera**

O autor recebeu apoio financeiro da OLIVETTI do Brasil,
da CAPES, da Texas A & M University e do CNPq.

- São Paulo, novembro de 2001 -

Agradecimentos

Primeiramente, quero agradecer a todos que direta ou indiretamente participaram deste trabalho. Em especial, agradeço:

ao Junior Barrera, pela incansável orientação durante todos estes anos; pelas diversas discussões científicas pois delas aprendi não apenas do assunto que discutíamos, mas também a ter uma postura científica mais séria do que já tinha; pelas oportunidades que ele me deu ao longo destes anos; por ter sido além de orientador, um amigo.

ao Edward R. Dougherty, pela orientação; por tudo que dele aprendi; pelas oportunidades que ele me deu durante o período que estive na Texas A&M University

ao Eduardo Jordão Neves, Gerald F. Banon, Nelson D. d'Avila Mascarenhas e Sérgio S. Furuie, por terem participado e colaborado com diversas sugestões, desenvolvimentos práticos e teóricos deste trabalho.

ao André F. Kohn, pelo excelente curso de "Reconhecimento de Padrões", no qual, entre outras coisas, aprendemos árvores de decisão.

ao Flávio Soares Corrêa da Silva, João E. Ferreira, Hugo A. Armelin, Roberto M. César Jr., Roberto de Alencar Lotufo e Routo Terada pelo que aprendi com eles em diversas oportunidades em que trabalhamos juntos.

à Nina S. T. Hirata por tudo que temos desenvolvido tanto em termos profissionais, quanto pessoais. Agradeço também pelas diversas figuras que ela fez para esta tese; pela revisão do capítulo do ISI; pelo "fine-tuning" de latex em algumas partes do trabalho.

ao Marcel Brun, pelos diversos trabalhos que desenvolvemos com as abordagens multire-solução e projeto híbrido de operadores com ajuda humana, os envelopes.

ao Daniel O. Dantas, Gustavo P. Esteves e ao Ronaldo F. Hashimoto, pelos trabalhos que desenvolvemos com a segmentação de imagens de microarray.

ao Franklin C. Flores, pelos diversos trabalhos que fizemos em segmentação de seqüências de imagens de vídeo.

ao Heraldo Madeira, pelo trabalho que fizemos com os Diagramas de Decisão Binários.

ao Caetano J. Carezzatto, Luciano V. de Araujo e Thiago T. Santos pela ajuda na administração da rede do Bioinfo.

à Barbara Fike por todo o apoio nos dado durante nossa estada no Texas.

aos funcionários e funcionárias da biblioteca, da seção de cópias, da secretaria do departamento de computação e da secretaria de pós-graduação por todo o apoio que deles recebi.

aos demais pesquisadores do Bioinfo, por diversos trabalhos que fizemos juntos.

ao João E. Kogler Jr. e ao Luiz G. de Barros pelo apoio e boa acolhida ao LIVES e ao SENAC.

aos demais pesquisadores do LIVES e do SENAC, pela boa acolhida.

aos pesquisadores do CAMDI-LAB da Texas A&M University com quem trabalhei diretamente: Artyom M. Grigoryan, Vishnu Kamat, Seungchan Kim, Sinan Batman, Siva Kumar, Yoganand Balagurunatan.

ao CNPq, à Olivetti do Brasil e ao Departamento de Engenharia Elétrica da Texas A$M University pelo apoio financeiro recebido durante a elaboração deste trabalho.

à compreensão de todos os amigos e parentes pela continuada ausência durante todos estes anos.

à minha família por todo o apoio e incentivo.

à compreensão de todos que esqueci de citar aqui :-)

À todos vocês, o meu Muito Obrigado!

Resumo

O projeto automático de operadores (i.e., mapeamentos) definidos entre imagens em níveis de cinza é um problema muito difícil em termos de complexidade computacional e precisão estatística. Restringir a classe de operadores é uma forma de tornar o problema tratável. O tamanho da classe dos operadores binários (i.e., que mapeiam imagens binárias em imagens binárias) invariantes por translação (i.e., cujas representações não dependem da posição em que são aplicados) e localmente definidos por uma janela W (i.e., cujas representações não dependem de informações que estão fora de um subconjunto do domínio definido por W) é exponencial no tamanho da janela (i.e., $2^{(2^{|W|})}$). Neste caso, limitar o tamanho de W é a restrição mais evidente para abordar o problema, seja do ponto de vista estatístico, quanto computacional. No caso das imagens em níveis de cinza, cujo tamanho da classe de operadores é $256^{(256^{|W|})}$, para imagens de 256 níveis, esta limitação não é suficiente. Neste caso temos que encontrar outras formas de restringir o espaço dos operadores. Uma forma de fazer isso é limitar a quantidade de níveis de cinza que podem ser vistos pela janela W a um intervalo K e, mas não necessariamente, limitar a quantidade de níveis de cinza do contra-domínio da função característica do operador em M. Sob estas limitações, um operador pode ser automaticamente projetado se ele pertencer à classe dos operadores invariantes por translação no espaço (domínio da imagem), e localmente definido dentro dos limites impostos por W e por K. Fazendo estas restrições, o espaço das possíveis funções características reduz-se para $M^{(K^{|W|})}$, que pode ser tratável se K e M não forem muito grandes. Neste trabalho estudamos os aspectos estatísticos e computacionais do projeto de operadores de imagens com estas restrições. Do ponto de vista estatístico, investigamos qual é o erro de estimação causado pela restrição nos níveis de cinza e pelo posicionamento vertical da restrição (i.e., janela da escala). Também abordamos o problema da escolha do método de representação e sua influência na indução, ou "generalização", do operador. Nesse sentido, uma das contribuições originais deste trabalho é a extensão do algoritmo ISI para achar a representação de operadores de imagens em níveis de cinza em termos da sua base. Esse método foi implementado e comparado com um outro método: a representação por árvores de decisão. Outra contribuição original foi estender o projeto estatístico de operadores em níveis de cinza para operadores de imagens coloridas e, também, estender o projeto para tratar restrições na resolução do espaço. Este é um outro tipo de restrição possível e que dá bons resultados. Finalmente, aplicamos o método em problemas reais, inclusive sobre imagens coloridas, com bons resultados.

Abstract

The automatic design of gray-level image operators (i.e., mappings between gray-level images) is a difficult problem in terms of computational complexity and statistical precision. To restric the class of operators is a way to treat the problem. The size of binary operators (i.e., mappings between binary images) class that are translation invariant (i.e., their representation do not depend on the position they are applied) and locally defined by a window W (i.e., their representation do not depend on the information outside a window W) is exponential on the size of the window W (i.e., $2^{(2^{|W|})}$). In that case, limiting the size of W is the most evident way to treat the problem either statistically or computationaly. In the case of gray-level images, the size of the class of operators is $256^{(256^{|W|})}$, for images of 256 levels, and limiting the size of the window may not be enough. In this case, one has to find other forms to restrict the space of operators. One way to do that is to limit the number of gray-leves that can be seen by the window W to an interval K and, but not necessarily, to limit the quantity of gray-levels of the characteristic function of the operator to M. Under these limitations, an operator may be automatically designed if it belongs to the class of operators that are translation invariant in the domain of the image and locally defined by the limits imposed by W and K. Besides that, if the operators are also range translation invariant, then their project is still more interesting statitically and computationally. Under those restrictions, the space of characteristic functions shrinks to $M^{(K^{|W|})}$, and the design is feasible if K and M are not too large. In this work we focused on the statistical and computational aspects of designing operators under these restrictions. On the statistical side, we investigate the estimation error due to the gray-level restriction and its vertical positioning (i.e., the range window). We have also approached the problem of the computational representation of those operators and its influence on the induction, or generalization, of the operator. In this sense, one of the original contributions of this work is the extension of the ISI algorithm to find the representation of the gray-level operator by its base. This method is implemented and has been compared to another method: the representation by decision trees. Other original contributions are: extension of the automatic design to color image operators and multiresolution design of operators. This is another possible restriction that gives good results. Finally, we have applied the method to several real problems, including to color image applications, with good results.

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A área de processamento de sinais e imagens [1, 2, 3] é uma das áreas de aplicação e pesquisa que mais tem crescido nos últimos anos. Várias são as razões para isso: o aumento da capacidade de processamento e armazenamento dos computadores conjugado com a diminuição dos custos desses equipamentos; os grandes avanços nos equipamentos ópticos e óptico-computacionais; diversos avanços significativos devidos a pesquisas básicas da área.

É fato que as aplicações que usam sinais (principalmente voz) e imagens são cada vez mais comuns; a variedade e a quantidade dos problemas resolvidos, parcialmente resolvidos e por serem resolvidos é muito grande. Assim, é razoável que essa área venha sendo dividida em diversas subáreas de estudo, que algumas delas estejam sendo mescladas com outras áreas do conhecimento e que novas áreas de pesquisa estejam sendo criadas. Por exemplo, Reconhecimento de Faces [4], "Human Computer Interation" [5], Processamento de Documentos [6], etc.

Neste trabalho estamos interessados num importante ponto em comum a todas essas áreas: o projeto de mapeamentos (operadores) entre imagens ou sinais. Afinal, normalmente precisamos aplicar diversas transformações para extrair informações úteis e resolver um dado problema que envolve processamento de sinais ou imagens.

A abordagem heurística é muito utilizada para resolver problemas práticos nessas áreas. Nesta abordagem um especialista planeja, de acordo com seus conhecimentos teóricos e práticos, que tipo de transformações e parâmetros (quando necessário) usará na solução do problema. Desta forma, a experiência do especialista conta muito para uma solução adequada e eficiente.

Uma outra forma é usar sistemas automáticos ou semi-automáticos que auxiliam o projeto dessas soluções. Sistemas automáticos são normalmente baseados em técnicas de aprendizado computacional [7] ou estimação estatística [8]. Sistemas semi-automáticos mesclam técnicas dos métodos heurísticos e não heurísticos [9, 10, 11].

O projeto automático de operadores entre imagens binárias é um assunto bastante estudado (mas não esgotado) [11]. A classe dos operadores binários de interesse é a dos W-operadores binários, i.e., operadores que mapeiam imagens binárias em imagens binárias, invariantes por translação (i.e., cujas representações não dependem da posição na imagem em que são aplicados) e localmente definidos por uma janela W (i.e., cujas representações não dependem de informações que estão fora de um subconjunto do domínio definido por W). O tamanho desta classe, i.e., a quantidade possível desses mapeamentos é exponencial no tamanho da janela (i.e., $2^{(2^{|W|})}$).

Uma outra classe útil, que também tem-se estudado no contexto de projeto estatístico, é a dos W-operadores para classificação (i.e., operadores que mapeiam imagens binárias em imagens em níveis de cinza, onde cada nível de cinza representa uma classe). Estes operadores são bastante úteis em aplicações de reconhecimento de caracteres (ou OCR [12]) e o tamanho desta classe varia de acordo com W e com a quantidade de níveis de cinza. Por exemplo, se l é o número de classes ou níveis de cinza, então o tamanho da classe é $l^{(2^{|W|})}$

A classe dos W-operadores entre imagens em níveis de cinza (i.e., que mapeiam imagens em níveis de cinza em imagens em níveis de cinza e são localmente definidos por uma janela W) tem tamanho $256^{(256^{|W|})}$ se tanto as imagens do domínio, quanto as do contra-domínio do operador possuírem 256 níveis de cinza. O projeto estatístico destes operadores esbarra na dificuldade de se fazer uma boa estimação das probabilidades condicionais de cada configuração de janela e dos níveis de cinza a ela condicionados. No caso binário essa dificuldade vai aumentando com o tamanho de W, apenas. Assim, limitar o tamanho da janela é uma forma de tornar o problema computacional e estatisticamente tratável. Porém, no caso das imagens em níveis de cinza, temos que encontrar outras maneiras de restringir o espaço dos operadores. Uma forma de fazer isso é limitar a quantidade de níveis de cinza que podem ser vistos pela janela W a um intervalo K. Sob estas limitações, um operador pode ser automaticamente projetado se ele pertencer tanto à classe dos operadores invariantes por translação no espaço (domínio da imagem), como à classe dos operadores localmente definidos dentro dos limites impostos por W e por K. Como há duas limitações a serem consideradas, estes operadores são chamados *WK-operadores*, ou operadores *"aperture"* [1]. Eles são algebricamente bem definidos e este trabalho concentra-se nos aspectos estatísticos e computacionais do projeto automático deles.

Em nosso trabalho, estudamos, especificamos e implementamos um sistema para projetar estes mapeamentos automaticamente a partir de uma especificação de alto nível do problema que é dada pelo usuário. Essa especificação é feita via pares de imagens (uma imagem de entrada associada a uma saída desejada) que expressam a intenção do usuário em obter um certo resultado semelhante quando este apresentar ao sistema outras imagens de entrada com características semelhantes. Os pares de imagens dados pelo usuário para o sistema projetar o operador são denominados *dados de treinamento*. A figura 1.1 ilustra o sistema através de um fluxograma.

O sistema projeta, via técnicas de estimação estatística ou aprendizado computacional, o melhor operador possível segundo um critério de erro escolhido pelo usuário e segundo as imagens de treinamento. Esse operador raramente é o operador ótimo, mas aproxima-se dele na medida em que vão sendo fornecidos mais e mais dados (imagens) de treinamento. Uma vez projetado, o usuário pode aplicar o operador em tantas imagens (com características estatísticas semelhantes às imagens de treinamento) quantas desejar.

O sistema foi especificado para ser flexível e, entre outras coisas, isso nos possibilitou experimentar três formas diferentes de projetar o operador: via árvores de decisão, via intervalos do núcleo do operador e via multiresolução. Como é de se esperar, o erro do operador depende dessa escolha (que é feita pelo usuário via os parâmetros de projeto do operador) e algumas comparações experimentais foram feitas para o melhor entendimento dessas três diferentes abordagens.

[1] Grafaremos entre aspas por não termos uma tradução adequada para o português, uma vez que o nome abertura [13] já havia sido adotado para designar o operador "opening" [14]. Esse nome foi adotado pois o janelamento $W \times K$ assemelha-se à abertura de uma máquina fotográfica.

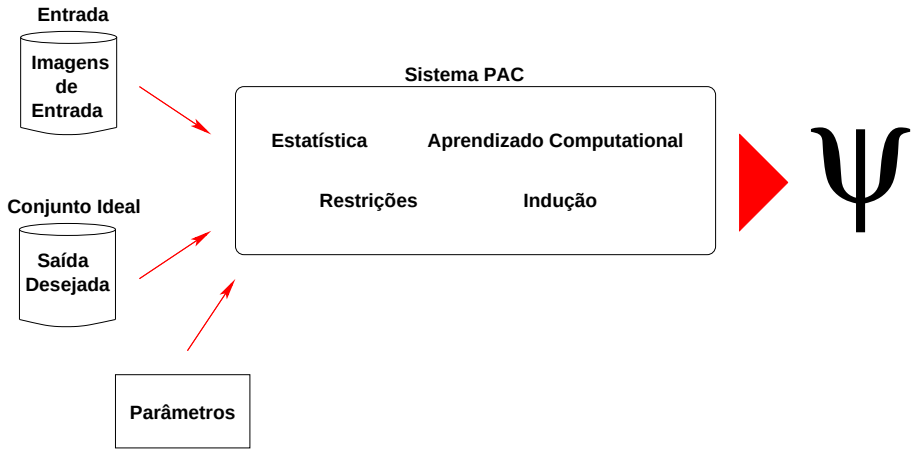


Figura 1.1: Sistema de aprendizado PAC

1.1 Fundamentação matemática, estatística e computacional

A base matemática deste trabalho é a Morfologia Matemática cujos estudos foram iniciados por Matheron, Serra e associados na década de sessenta na França [15, 16, 17, 14].

O principal objeto de estudo da morfologia matemática (MM) são os operadores definidos entre reticulados completos e suas propriedades de decomposição. Já que podemos modelar as imagens como elementos de um reticulado completo e as transformações de imagens como operadores definidos entre os reticulados [17], a MM nos dá uma estrutura algébrica adequada para o estudo da área de processamento de imagens digitais. Um exemplo desta utilidade é o uso e a classificação dos operadores de acordo com o estudo de suas propriedades algébricas. Este estudo (de propriedades algébricas) permite definir subclasses (subfamílias) de operadores e, dessa forma, utilizar uma classe adequada para resolver um determinado problema. Um outro exemplo é o estudo das possíveis formas de representação de um operador. Neste caso, pode-se definir que representações são mais adequadas para uma determinada solução de software e hardware.

A primeira obra em português sobre MM [13] é de Banon e Barrera. Ali, além de diversos resultados teóricos, eles apresentam de forma didática o conceito de Máquina Morfológica (MMACH) que havia sido introduzida por eles em [18, 13] e que tem permeado nosso estudo de diversas formas.

Fora os livros clássicos da área, há uma grande variedade de artigos sobre MM que são importantes para o nosso trabalho. Alguns de cunho algébrico: [19, 20, 18, 21], outros de cunho genérico: [22, 23]. Há ainda os de cunho algorítmico: [24, 25], que são úteis na implementação de operadores morfológicos. Sob certo sentido, nos é útil os trabalhos de cunho probabilístico (por exemplo, sob a ótica de conjuntos aleatórios) temos: [26, 27, 28, 29, 30, 31].

A base estatística do trabalho está na teoria de Estimação (ou Inferência) Estatística [32]. Ela insere-se no projeto estatístico de operadores de duas formas diferentes: o uso de dados para fazer estimações sobre conjuntos ou coleções de conjuntos, o que nos leva aos modelos de ruídos ou imagens; e na estimação dos parâmetros que definem um operador morfológico. Em outras palavras, o conhecimento destes parâmetros pode auxiliar no projeto automático de operadores morfológicos de diversas formas. Por exemplo, a precisão de um operador (i.e., o quanto ele se aproxima do operador ótimo) projetado pelo sistema implementado depende de uma boa estimação dos parâmetros que o

definem. Estes, por sua vez, dependem da quantidade de dados de treinamento. Assim, se conhecermos os parâmetros que regem um certo fenômeno de degradação de imagens, podemos usá-los para gerar quantas imagens de treinamento forem necessárias para chegar a uma certa precisão desejada.

Da teoria de Reconhecimento de Padrões [33, 29, 34] e Aprendizado Computacional [35, 36, 37, 38, 39, 7, 40] herdamos diversos métodos e técnicas que são úteis para o projeto automático de operadores. Essa utilidade pode se dar, por exemplo, em termos de comparação da qualidade dos operadores, ou métodos para estimar o número mínimo de exemplos para que o operador tenha um erro não superior a um valor estipulado.

Além disso, é de grande utilidade o conhecimento de técnicas eficientes de Análise e Construção de Algoritmos [41, 42, 43, 44] e conhecimentos de Engenharia de Software [45].

A área de processamento de imagens envolve grandes quantidades de dados e de processamento. O projeto de operadores via técnicas de estimação estatística é ainda pior pois envolve problemas cuja solução é combinatória. Estas duas áreas exigem um conhecimento profundo de construção e análise de algoritmos e de engenharia de software para a implementação eficiente e correta dos programas.

1.2 Resumo histórico

Houve um grande progresso no projeto automático de operadores não lineares para imagens binárias na década de 90 em contraposição com o projeto para imagens em níveis de cinza. Isso se deve à várias razões: (1) a relação entre imagens binárias e a geometria estocástica [46] colocou o assunto no contexto de operadores sobre conjuntos aleatórios [15]; (2) o sucesso da morfologia matemática em prover algoritmos práticos e intuitivos para problemas de visão computacional motivou a pesquisa em desenvolvimento automático de operadores, já que daí não há necessidade de um treinamento árduo e longo de um profissional em processamento de imagens [16, 47]; (3) a relação entre filtros binários definidos por uma janela, autômatos celulares e operadores morfológicos básicos motivou o descobrimento do poder da representação morfológica [48, 49, 50, 18]; (4) a complexidade computacional do problema no âmbito dos operadores em níveis de cinza é um incentivo forte para a busca de soluções binárias, sempre que possível. (5) O crescimento combinatório do número de possíveis observações com o crescimento do tamanho da janela e da escala (dos níveis de cinza) torna a estimação estatística praticamente impossível para níveis de cinza.

Os primeiros trabalhos sobre representação foram feitos para filtros crescentes [16, 19, 20], e baseado neles e nos trabalhos de Wiener e Pugachev, Dougherty fez os primeiros trabalhos de projeto estatístico ótimo de operadores binários [51]. Esse panorama estendeu-se após o artigo seminal de Banon e Barrera que mostra que operadores entre imagens binárias podem ser decompostos morfologicamente independentemente de serem crescentes, ou não [18]. O caráter lógico dessas representações foi apresentado [52], o projeto ótimo foi tratado para operadores genéricos [53] e o assunto foi colocado no contexto da teoria de aprendizado computacional [54]. Vários problemas foram considerados: precisão da estimação quando o operador é estimado por realizações de imagens [55]; filtros ótimos de conversão de resolução [6]; uso de informação a priori [56, 57, 58]; projeto de filtros parcialmente restritos [59] e projeto multiestágio de filtros [60].

O primeiro trabalho de projeto de otimização de filtros para imagens em níveis de cinza no contexto da morfologia matemática [61] foi baseado na representação morfológica de operadores crescentes entre imagens em níveis de cinza [49, 50]. A otimização foi tratada como um problema de

busca e o espaço minimal dos elementos estruturantes foi achado. Entretanto, esse espaço mínimo é extremamente grande, assim como o número de filtros possíveis, mesmo no caso crescente. A teoria de representação lá desenvolvida também tinha problemas estruturais pois era algebricamente baseada na representação de uma imagem em níveis de cinza por umbras (o espaço dos pontos que estão abaixo do gráfico da função que gera a imagem) [14]. Porém, a motivação da abordagem vem do fato que a morfologia binária é algebricamente embutida na morfologia em níveis de cinza e do fato da teoria para otimização binária servir para níveis de cinza [61, 62]. Uma abordagem adaptativa foi também proposta para projeto de filtros para níveis de cinza [63]. Ela provou ser útil para alguns tipos de restauração, mas a convergência não foi caracterizada (embora o algoritmo possa convergir para um filtro não ótimo [64]) e os problemas mencionados permanecem.

Um grande progresso matemático aconteceu com o segundo artigo seminal de Banon e Barrera [21]. Neste artigo, apresenta-se a teoria de representação morfológica para operadores entre reticulados completos quaisquer. Embora o artigo não tenha a devida difusão por ser bastante abstrato, ele dá todo o arcabouço necessário para a implementação do método no âmbito do filtros restritos a uma janela [65], o que deu origem à teoria da representação da morfologia computacional [66, 67].

Do ponto de vista de otimização, não basta termos uma teoria de representação, a teoria tem de ser adequada a otimização, ao modelamento do espaço de imagens e adequada a colocação de restrições algébricas, características presentes na MM. Esta é a principal razão do sucesso obtido no desenvolvimento dos métodos de projeto de operadores morfológicos binários nos últimos anos [68].

A teoria da morfologia computacional também satisfaz aqueles requisitos porque ela é uma generalização direta da representação binária em que as representações via núcleo, redução e representação via base para operadores crescentes são extensões da teoria de representação binária. Porém, como o espaço de operadores é muito grande, pouco progresso havia sido feito nesse âmbito.

Isso não impediu que se trabalhassem com as imagens níveis de cinza. Uma forma de usar a metodologia binária para imagens em níveis de cinza dá-se através dos filtros "stack" [69]. Estes filtros são construídos a partir de filtros binários crescentes que operam individual e igualmente em cada conjunto resultante de cada possível limiar de um sinal. Um exemplo de um filtro "stack" é o filtro da mediana, que é aplicado com sucesso em sinais ruidosos, principalmente ruído branco. Um ponto favorável ao projeto de filtros "stack" é que um número pequeno de sinais produz uma quantidade enorme de dados de treinamento [70, 71, 72]. Porém, o sistema tem que ser projetado com cuidado pois há maneiras de implementar o operador de forma que ele seja mais eficiente que se aplicássemos um operador binário para cada nível do sinal [73]. Do ponto de vista de projeto, embora os conjuntos formados pelos "thresholds" não serem igualmente distribuídos, i.e., não obedecerem ao mesmo processo aleatório binário, é possível provar que a otimização não é afetada [73, 74].

O conceito de filtro "stack" pode ainda ser estendido via uma função booleana sobre as variáveis representando o sinal de uma banda de níveis de cinza em torno de um particular nível. Tais filtros são chamados filtros "stack" generalizados e também foram otimizados [75]. A classe desses filtros é maior que a anterior, mais ainda formam uma subclasse dos filtros crescentes [49, 61]. A representação dessa classe de filtros é um caso especial da representação geral de filtros crescentes e portanto a sua otimização é restrita ao caso da classe dos filtros crescentes [61]. Enquanto isso ajuda na precisão do operador projetado [55], às vezes é necessário lidar com a inaceitável subotimalidade dos filtros crescentes [6, 54].

Uma outra razão para não se ter abordado o problema do projeto de operadores para imagens em níveis de cinza via otimização estatística baseada na representação morfológica é a grande facilidade

de se projetar filtros lineares ótimos. Além de serem pouco custosos para se projetar, eles também são simples de serem representados (tão simples que são conhecidos como "filtros de memória zero"). Esta talvez seja a razão principal do porquê até hoje eles serem a primeira opção no projeto de uma solução em processamento de imagens ou visão computacional. Não podemos esquecer que até poucos anos atrás, memórias para computador eram caras e poucos sonhavam com o poder computacional atual.

Uma grande desvantagem desses filtros é que eles não dão bons resultados quando as imagens (ou sinais) estão corrompidas com ruídos não aditivos, quando o ruído não é independente da imagem (ou sinal), quando as distribuições que governam o ruído não são gaussianas e, principalmente, quando aplicados sobre sinais digitais de mais de uma dimensão [76].

Um grande passo no sentido de tornar viável o projeto de operadores em níveis de cinza via estimação estatística foi dado por Barrera e Dougherty [62]. Neste trabalho eles introduzem os operadores WK, que são operadores invariantes por translação e localmente definidos tanto no espaço como nos níveis de cinza. Estes operadores são o assunto desta tese.

1.3 Contribuições da tese

Nesta seção resumimos as contribuições deste trabalho. A descrição detalhada das contribuições originais encontram-se nos capítulos de 4 a 8.

- Dentre as possíveis representações que pesquisamos para os WK-operadores (árvores de decisão paralelas, árvores de decisão oblíquas, intervalos num reticulado, redes neurais), escolhemos duas: a representação por árvores de decisão e a representação por intervalos. Neste sentido, uma das contribuições originais foi estender o algoritmo ISI para níveis de cinza [77] e a outra foi fazer um estudo extensivo da aplicação das árvores de decisão oblíquas [78] para a representação do núcleo dos operadores níveis de cinza. A importância destas duas contribuições vem do fato que da representação depende a eficiência (em termos de memória e ciclos de CPU) para projetar e aplicar o operador. Além disso, certas representações podem ser mais adequadas para embutir conhecimentos algébricos a priori, ou para melhorar o poder de predição (generalização) do operador.

- Comparamos os resultados dos WK-operadores com os filtros lineares ótimos restritos a uma janela W e verificamos, através de centenas de experimentos, a vantagem dos primeiros em relação aos últimos [79, 78]. Dentre os experimentos realizados com sinais, fizemos: filtragem de ruído gaussiano em sinais unidimensionais; desembaçamento de sinais unidimensionais; restauração de sinais corrompidos ao mesmo tempo por ruído gaussiano e embaçamento em sinais unidimensionais. Dentre os experimentos com imagens, fizemos: filtragem de ruído gaussiano e desembaçamento. O problema de desembaçamento neste contexto (imagens 2D) é de grande importância já que não existem bons filtros para resolvê-lo [76]. Em todos estes experimentos pudemos verificar como o erro dos operadores varia com o posicionamento da janela nos níveis de cinza, o tamanho da janela espacial e da janela nos níveis de cinza e com a quantidade de exemplos de treinamento.

- Verificamos a subotimalidade dos WK-operadores quando variam-se o tamanho da janela ($|W|$), a restrição nos níveis de cinza ($|K|$) e o número de exemplos. Esta parte necessita de uma explicação mais detalhada: já que não conhecemos as distribuições que regem o operador, o projeto

é baseado em observações e portanto, se o tamanho da amostra (número de exemplos observados) não é suficientemente grande, a precisão do operador projetado não será necessariamente ótima. Na verdade, é subotima na maioria dos casos quando usamos janelas grandes. Nesta parte, a contribuição foi de expandir o conhecimento que havia sido obtido para o projeto de filtros binários [54, 80].

- Estendemos o projeto multiresolução de operadores para os operadores níveis de cinza e obtivemos resultados melhores que os filtros lineares ótimos de janela restrita, além de resultados comparáveis com os WK-operadores com representação por árvores de decisão [81, 82]. Neste estudo, investigamos o problema da suboptimalidade do operador multiresolução, implementamos um sistema para o projeto desses operadores e fizemos dezenas de experimentos com imagens e sinais unidimensionais para investigar o comportamento desses operadores quando variamos o tamanho da amostra de treinamento e o tamanho das janelas espacial e níveis de cinza.

- Um dos anseios ao final de nosso trabalho de mestrado [83] era um dia poder usar as técnicas de projeto automático de operadores para encontrar marcadores para segmentação e, assim, eliminar a parte heurística do paradigma de Beucher-Meyer. Durante o doutorado pudemos aplicar o projeto automático de WK-operadores para encontrar marcadores [84, 85] e os resultados mostraram que era possível. A parte heurística não é totalmente eliminada pois o classificador não é perfeito. Porém, essa parte limita-se a uma filtragem de componentes conexas de área pequena. Em cooperação com F. C. Flores, aplicamos estas idéias para a segmentação de seqüências de vídeo monocromáticas.

- Motivados pelos resultados de segmentação de seqüências de vídeo monocromáticas, implementamos a generalização dos WK-operadores para imagens coloridas e os resultados foram igualmente bons [86, 87]. Este trabalho envolveu, entre outras coisas, uma pesquisa bibliográfica sobre imagens coloridas e sua segmentação. Dessa pesquisa, pudemos propor e implementar diversas medidas de diferenciação entre pixels coloridos (mais especificamente gradientes) que transformam imagens coloridas em imagens monocromáticas e assim aplicar o paradigma de Beucher-Meyer em sua forma original. Ainda em cooperação com F. C. Flores, aplicamos estas idéias para a segmentação de seqüências de vídeo coloridas.

Parte do trabalho aqui apresentado foi realizado em cooperação com o prof. Dr. E.R. Dougherty entre setembro de 1997 e fevereiro de 1999 na Texas A&M University.

A avaliação das técnicas empregadas neste trabalho foram feitas via simulações em computador de modelos de imagens e sinais visando a solução de algum problema de processamento de imagens. Uma biblioteca de rotinas em linguagem C tem sido desenvolvida, em conjunto com Nina S. T. Hirata, usando-se a plataforma Khoros [88, 47], com as rotinas necessárias para o projeto, aplicação, análise e avaliação das técnicas estudadas (já possui aproximadamente 50000 linhas de código e mais de 100 páginas de documentação). Outras técnicas também têm sido implementadas e estudadas para comparação e têm sido integradas à biblioteca.

O trabalho de implementação tem envolvido especificação e estruturação cuidadosa do software, tomando-se o cuidado de produzir uma biblioteca de rotinas independentes de plataforma, objetivando, assim, a maior portabilidade, o desenvolvimento de um sistema integrado, e a simplificação de desenvolvimento e de manutenções futuras. Desta maneira, este trabalho não possui as mesmas

características dos conhecidos "gradwares" que são feitos única e exclusivamente para o uso do estudante de pós-graduação que o implementou. A biblioteca em questão tem sido usada por diversos pesquisadores do grupo e em aplicações profissionais ligadas a reconhecimento de caracteres.

1.4 Estrutura da tese

- No capítulo 2 (**Fundamentos**), apresentamos diversos conceitos básicos ligados ao projeto estatístico de filtros lineares e não-lineares.

  O projeto de filtros lineares envolve o encontro dos parâmetros (ou pesos) do núcleo de um operador linear chamado *Convolução* via estimação estatística. Este capítulo justifica-se por duas razões: (i) em várias aplicações fizemos comparações com filtros lineares; (ii) a idéia inicial do projeto estatístico de filtros não lineares foi inspirada no projeto estatístico destes filtros.

  Apresentamos também os conceitos de reticulado, reticulado completo e operações em reticulados. Finalmente, introduzimos os operadores básicos da morfologia matemática.

- No capítulo 3 (**W-operadores e seu projeto**), apresentamos as idéias básicas do projeto de operadores morfológicos.

  Apresentamos os operadores de janela, ou W-operadores, sua caracterização em termos de sup-geradoras e em termos de árvores de decisão.

  Apresentamos o projeto automático de operadores de imagens, alguns conceitos de aprendizado computacional e uma discussão sobre indução, ou como também é conhecido, generalização.

  Ao final, mostramos uma análise do erro cometido quando restringe-se a janela do operador.

- No capítulo 4 (**Operadores "Aperture" e seu projeto**), apresentamos a classe dos operadores morfológicos que são o objeto de estudo deste trabalho. Estes operadores de imagem são restritos tanto no espaço, quanto nos níveis de cinza.

  Este capítulo apresenta a definição dos operadores "aperture", ou WK-operadores, na sua versão mais geral e a sua decomposição algébrica em operadores sup-geradores.

  Apresentamos o projeto estatístico dos operadores "aperture" e uma análise de erro do operador projetado em relação ao operador ótimo. Ao final, apresentamos algumas simulações feitas para verificar qual a probabilidade de um sinal estar dentro do janelamento WK.

- No capítulo 5 (**Representação e indução via árvores de decisão**), relembramos o algoritmo OC1 que implementamos no sistema para a representação do operador "aperture".

  Apresentamos parte de um estudo extenso da aplicação dos operadores "aperture" em problemas de filtragem de ruído gaussiano em sinais e desembaçamento em sinais e imagens 2D.

- No capítulo 6 (**Representação e indução via intervalos**), apresentamos o algoritmo ISI de indução da base do operador a partir do kernel, ou de uma parte do kernel.

  Recordamos o algoritmo ISI utilizado para projetar operadores binários e apresentaremos a extensão do algoritmo para W-operadores binários para classificação (mapeamentos entre imagens binárias e imagens em níveis de cinza, úteis para, por exemplo, OCR). Apresentamos também a extensão do algoritmo para os operadores "aperture" e uma comparação experimental com as árvores de decisão.

- No capítulo 7 (**Representação e indução via multiresolução**), recordamos conceitos de multiresolução de operadores e apresentaremos o projeto de operadores "aperture" por esta metodologia.

  O conceito de pirâmide de operadores é apresentado, assim como o projeto e a análise de erro do operador multiresolução projetado. Apresentamos o "software" implementado para testar a idéia e, ao final, apresentamos uma parte dos experimentos que fizemos para analisar experimentalmente o erro dos operadores projetados por essa abordagem em relação aos operadores lineares e "apertures".

- O capítulo 8 (**Aplicações**) tem três subseções: na primeira apresentamos um a aplicação dos operadores "aperture" para restauração de embaçamento ("deblurring") em imagens reais.

  Na segunda mostramos uma aplicação dos operadores "aperture" para melhoramento de resolução.

  Na terceira mostramos a aplicação dos operadores "aperture" como ferramentas auxiliares para encontro de marcadores. Apresentamos também a extensão desses operadores para espaço das imagens coloridas e alguns exemplos de aplicação nesse contexto.

- No capítulo 9 (**Conclusão**) apresentamos algumas conclusões obtidas neste trabalho e possíveis estudos a serem realizados no futuro.

- **Apêndice A** apresentamos uma recordação sobre o paradigma de segmentação de imagens e os operadores morfológicos importantes para a segmentação.

- **Apêndice B** apresentamos uma lista das nossas publicações relacionadas com o tema da tese.

- **Apêndice C** apresentamos uma lista das nossas publicações pouco ou não relacionadas com o tema da tese, mas que foram desenvolvidas ao longo deste trabalho.

Capítulo 2

Fundamentos

Neste capítulo apresentamos alguns conceitos básicos que facilitam o entendimento das idéias nas quais se baseiam o projeto estatístico de filtros lineares de janela restrita e do projeto estatístico de operadores morfológicos.

Iniciamos com uma pequena recordação do conceito de imagens e das medidas mais importantes da diferença entre dois sinais ou imagens. Por causa da aleatoriedade dos processos de formação e aquisição de sinais ou imagens, apresentamos também a relação delas com os processos aleatórios discretos.

Recordamos o conceito de operadores lineares ótimos e a teoria na qual eles se fundamentam: estatística multivariada e álgebra linear. A estatística multivariada estuda o comportamento simultâneo de muitas variáveis aleatórias, daí o nome estatística, ou análise multivariada; a álgebra linear estuda transformações entre espaços lineares. Embora os operadores lineares sejam simples de implementar, eles possuem algumas limitações conhecidas, como, por exemplo, não darem bons resultados para imagens de duas ou mais dimensões. Daí a motivação para achar filtros não-lineares para imagens.

Para terminar, recordamos conceitos da morfologia matemática e de operadores morfológicos. Começamos pelo conceito de reticulado completo e as operações de intersecção, união e complemento. Depois, recordamos os operadores básicos da morfologia.

Os operadores morfológicos têm despertado cada vez mais interesse na comunidade de processamento de sinais e imagens. Este interesse vem, principalmente, do sucesso apresentado pelas soluções morfológicas e do fato que os computadores estão cada vez mais velozes e os circuitos de memórias mais baratos.

2.1 Imagens e transformações de imagens

Neste trabalho estamos interessados na representação numérica de um instante de uma cena. Mais especificamente, numa representação sobre o domínio $Z \times Z$ em que cada ponto ou, pelo menos, cada vizinhança de pontos da cena é representado por um número. A forma de se chegar a essa representação é vastamente explorada na literatura básica de processamento de imagens [1, 2, 89, 3, 90, 91].

2.1.1 Sinais e imagens digitais

Quando a representação numérica é feita por números pertencentes a um subconjunto finito dos números reais, dizemos que ela é uma **imagem digital**. Como exemplos, podemos citar: o conjunto $[0, 255]$, que é um intervalo dos números inteiros positivos, Z^+; ou um subconjunto finitos dos números racionais. Quando apenas dois valores forem suficientes para representar a imagem, por exemplo $\{0, 1\}$ ou $\{0, 255\}$, a imagem digital será chamada de **imagem binária**, caso contrário ela será chamada de **imagem em níveis de cinza** e cada valor será um **nível de cinza**.

Sinais ou imagens digitais podem ser formalmente definidas e representadas por mapeamentos que vão de um conjunto não vazio E, que é um grupo abeliano com respeito a uma operação binária $+$, em um reticulado ordenado completo (ou *cadeia*) L [14, 13].

Em nosso trabalho estamos interessados em imagens cujo domínio, E, é o conjunto $Z \cup \{-\infty, \infty\}$ (para o caso dos sinais), ou o conjunto $(Z \cup \{-\infty, \infty\}) \times (Z \cup \{-\infty, \infty\})$ (para o caso das imagens bidimensionais), e o contra domínio é o conjunto $Z \cup \{-\infty, \infty\}$. Daqui para frente usaremos a letra L para denotar o conjunto $Z \cup \{-\infty, \infty\}$. Imagens binárias podem tanto ser representadas como elementos da coleção das partes de E (i.e., coleção de todos os subconjuntos de E), denotada $\mathcal{P}(E)$, ou como uma função de E em $[0, 1]$ (ou $\{0, k\}$, $k \in Z$).

O conjunto de todas as possíveis imagens (ou sinais[1]) digitais de E em L será denotado por L^E. Por exemplo, o conjunto de todas as imagens binárias de E em $\{0, 1\}$ será denotado por $\{0, 1\}^E$, e o conjunto de todas as imagens com até 255 níveis de cinza será denotado por $[0, 255]^E$.

2.1.2 Imagens e sinais como processos aleatórios discretos

A hipótese de que sistemas que processam imagens discretas reais (i.e., imagens que foram digitalizadas e não sintetizadas) são melhor projetados se levamos em conta características "médias" de uma classe de imagens é um princípio de nosso trabalho.

Este princípio baseia-se em diversos argumentos que nos indicam que imagens reais não são determinísticas, isto é, elas não podem ser expressas em uma fórmula matemática explícita[2]. Porém, sabemos que há vozes contra essa abordagem e que essa discussão beira os campos da Filosofia [92].

2.1.3 Vetores aleatórios discretos

Vetores são uma representação adequada para sinais ou seqüências aleatórias finitas, principalmente no que concerne à sua análise. Nessa representação, cada posição do vetor representa uma variável, ou a realização de uma variável aleatória e, por isso, os vetores deste tipo são conhecidos como *vetores de variáveis aleatórias*, ou *vetores aleatórios discretos*. A palavra discreto neste contexto refere-se à discretização no domínio espacial do sinal ou da imagem.

Denotaremos variáveis aleatórias com letras maiúsculas e vetores aleatórios por letras maiúsculas

[1] A partir de agora usaremos apenas o termo imagens, exceto quando estivermos tratando apenas de sinais.

[2] Note que, uma vez digitalizadas, elas podem!

em negrito. Por exemplo,

$$\mathbf{X} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix}.$$

denota um vetor aleatório formado pelas variáveis aleatórias $X_1, X_2, \ldots, X_n$. Em contraste, o transposto de $\mathbf{X}$, i.e., $\mathbf{X}'$, será denotado por $(X_1, X_2, \ldots, X_n)$. Desta maneira, como $(\mathbf{X}')' = \mathbf{X}$, então ele também pode ser denotado por $\mathbf{X} = (X_1, X_2, \ldots, X_n)'$.

Denotaremos a realização de variáveis aleatórias por letras minúsculas e a realização de vetores aleatórios por letras minúsculas em negrito. Por exemplo, $\mathbf{x} = (x_1, x_2, \ldots, x_n)'$ denota uma realização de $\mathbf{X} = (X_1, X_2, \ldots, X_n)'$.

Um vetor aleatório é completamente caracterizado pela função de distribuição conjunta de suas variáveis aleatórias [93, 94, 32, 8], i.e.,

$$F(x_1, x_2, \ldots, x_n) = \Pr(X_1 \leq x_1, X_2 \leq x_2, \ldots, X_n \leq x_n) = \Pr(\mathbf{X} \leq \mathbf{x})$$

Denotaremos por $p(\mathbf{x})$ a função de probabilidade conjunta de $\mathbf{X}$, i.e., $p(\mathbf{x}) = p(x_1, x_2, \ldots, x_n) = Pr(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$, onde Pr denota probabilidade.

No projeto de filtros ótimos é comum o interesse no estudo de um vetor aleatório $\mathbf{X}$ em conjunto com uma variável aleatória Y. Da mesma forma que no caso de vetores aleatórios, essa conjunção é completamente caracterizada pela função de distribuição conjunta do vetor $\mathbf{X}$ com a variável Y, i.e.,

$$F(\mathbf{X}, Y) = \Pr(X_1 \leq x_1, X_2 \leq x_2, \ldots, X_n \leq x_n, Y \leq y) = \Pr(\mathbf{X} \leq \mathbf{x}, Y \leq y)$$

2.1.4 Análise de momentos

No caso em que as distribuições de probabilidade que governam o comportamento dos processos aleatórios sejam desconhecidas, é comum analisar os momentos da distribuição (que é possível mesmo desconhecendo a distribuição, já que é possível obter estimativas dos momentos). Em geral, faz-se a análise do primeiro e segundo momentos.

Definição 2.1 *Seja* $\mathbf{X}$ *um vetor aleatório. O primeiro momento, ou esperança, de* $\mathbf{X}$ *é dada por,*

$$\overline{\mathbf{X}} = E[\mathbf{X}] = (E[X_1], \ldots, E[X_n])$$

Definição 2.2 *Seja* $\mathbf{X}$ *um vetor aleatório e* $\mathbf{X}'$ *o seu transposto. O segundo momento, ou matriz de correlação, de* $\mathbf{X}$ *é dado por,*

$$\mathbf{R_X} = E[\mathbf{X}\mathbf{X}']$$

Por exemplo, a matriz de correlação de $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ é dada por:

$$\mathbf{R_X} = E[\begin{pmatrix} X_1 \\ X_2 \\ \cdots \\ X_n \end{pmatrix} (X_1 X_2 \ldots X_n)] = \begin{pmatrix} E[X_1^2] & E[X_1X_2] & \cdots & E[X_1X_n] \\ E[X_2X_1] & E[X_2^2] & \cdots & E[X_2X_n] \\ \vdots & \vdots & \ddots & \vdots \\ E[X_nX_1] & E[X_nX_2] & \cdots & E[X_n^2] \end{pmatrix}.$$

Definição 2.3 *Seja* $\mathbf{X}$ *um vetor aleatório e* $\overline{\mathbf{X}}$ *sua esperança. O segundo momento central (momento ao redor da média), ou matriz de covariância, de* $\mathbf{X}$ *é dado por,*

$$\mathbf{C}_x = E[(\mathbf{X} - \overline{\mathbf{X}})(\mathbf{X} - \overline{\mathbf{X}})']$$

Por exemplo, a matriz de covariância de $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ é dada por:

$$\mathbf{C}_x = E[\begin{pmatrix} X_1 - \overline{\mathbf{X}} \\ X_2 - \overline{\mathbf{X}} \\ \cdots \\ X_n - \overline{\mathbf{X}} \end{pmatrix} (X_1 - \overline{\mathbf{X}} \;\; X_2 - \overline{\mathbf{X}} \ldots X_n - \overline{\mathbf{X}})] =$$

$$= \begin{pmatrix} E[(X_1 - \overline{X_1})^2] & E[(X_1 - \overline{X_1})(X_2 - \overline{X_2})] & \cdots & E[(X_1 - \overline{X_1})(X_n - \overline{X_n})] \\ E[(X_2 - \overline{X_2})(X_1 - \overline{X_1})] & E[(X_2 - \overline{X_2})^2] & \cdots & E[(X_2 - \overline{X_2})(X_n - \overline{X_n})] \\ \vdots & \vdots & \ddots & \vdots \\ E[(X_n - \overline{X_n})(X_1 - \overline{X_1})] & E[(X_n - \overline{X_n})(X_2 - \overline{X_2})] & \cdots & E[(X_n - \overline{X_n})^2] \end{pmatrix}.$$

Definição 2.4 *Sejam* $\mathbf{X}$ *um vetor aleatório e* Y *uma variável aleatória. A matriz de correlação cruzada de* $\mathbf{X}$ *e* Y *é dada por,*

$$\mathbf{R}_{xy} = E[\mathbf{X}Y']$$

Por exemplo, a matriz de correlação cruzada de $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ e Y é dada por:

$$\mathbf{R}_{xy} = E[\begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix} Y] = \begin{pmatrix} E[X_1 Y] \\ E[X_2 Y] \\ \vdots \\ E[X_n Y] \end{pmatrix}.$$

Definição 2.5 *Sejam* $\mathbf{X}$ *e* Y *como anteriormente,* $\overline{\mathbf{X}}$ *e* $\overline{Y}$ *suas respectivas esperanças. A matriz de covariância cruzada de* $\mathbf{X}$ *e* Y *é dada por,*

$$\mathbf{C}_{xy} = E[(\mathbf{X} - \overline{\mathbf{X}})(Y - \overline{Y})]$$

Por exemplo, a matriz de covariância cruzada de $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ e Y é dada por:

$$\mathbf{C}_{xy} = E[\begin{pmatrix} (X_1 - \overline{\mathbf{X}})(Y - \overline{Y}) \\ (X_2 - \overline{\mathbf{X}})(Y - \overline{Y}) \\ \vdots \\ (X_n - \overline{\mathbf{X}})(Y - \overline{Y}) \end{pmatrix}] = \begin{pmatrix} E[(X_1 - \overline{\mathbf{X}})(Y - \overline{Y})] \\ E[(X_2 - \overline{\mathbf{X}})(Y - \overline{Y})] \\ \vdots \\ E[(X_n - \overline{\mathbf{X}})(Y - \overline{Y})] \end{pmatrix}.$$

2.1.5 Processos aleatórios discretos

Um *processo aleatório* é uma família de variáveis aleatórias $\{\chi(\omega, t)\}$ dependentes de um parâmetro t que assume valores em um *conjunto de referência* T. Assim, para cada $t \in T$ fixo, $\chi(\omega, t)$ é uma

variável aleatória $\chi(\omega, t)$ definida num espaço amostral S ($\omega \in S$). Por outro lado, se ω é fixado, então $\chi(\omega, t)$ define uma função sobre T denominada *realização* do processo aleatório.

Segundo Dougherty [8] e Gikhman [95], se T é arbitrário ou mesmo a reta real R, o termo *função aleatória* é mais conveniente que processo aleatório, que é melhor empregado quando t refere-se a tempo. No caso em que t é uma variável espacial, então a função também é chamada de *campo aleatório* ou *imagem aleatória*. Dizemos que o processo é *discreto* se T é um conjunto discreto. Dizemos que o processo é *contínuo* se T é um conjunto contínuo. A figura 2.1 ilustra o conceito. O processo aleatório discreto apresentado é uma família de cosenos cuja amplitude e freqüência são fixados, mas o ângulo de fase é aleatório, isto é,

$$\chi(\omega, t) = cos(\frac{t.\pi}{10} + \theta) \tag{2.1}$$

onde θ é o valor de uma variável aleatória Θ.

A figura 2.2 ilustra um processo aleatório contínuo. Neste caso, o processo é família de funções coseno com ruído, isto é,

$$\chi(\omega, t) = cos(\frac{t.\pi}{10} + c) + \theta \tag{2.2}$$

onde θ é o valor de uma variável aleatória Θ entre $[-0.5, 0.5]$.

2.1.6 Operadores de imagens

Em termos tecnológicos, imagens per si não servem para muita coisa se não podemos analisá-las e delas extrair informações. Para isso, elas precisam ser transformadas pelos operadores de imagens [1, 2, 89, 3, 90, 91].

Definição 2.6 *Dado um conjunto de imagens L^E, onde E e L são definidos como anteriormente, uma transformação Ψ de L^E em L^E é chamada de operador de imagem ou filtro.*

Se g e h são duas imagens de L^E relacionadas por Ψ, então escrevemos

$$g = \Psi(h), \tag{2.3}$$

isto é, g é a imagem obtida de h por Ψ.

2.1.7 Medidas pontuais entre duas imagens

Em várias partes deste trabalho iremos mensurar o quanto uma imagem difere de outra, mais especificamente, se Ψ é um operador definido sobre L^E, e $g \in L^E$ é a imagem ideal que gostaríamos de ter obtido com a aplicação de Ψ sobre $h \in L^E$, então é natural que queiramos saber quanto $\Psi(h)$ difere de g, em média.

Para isso, recorreremos a duas medidas pontuais muito utilizadas, o "*Mean Absolute Error*", ou MAE, e o "*Mean Square Error*", ou MSE, de Ψ, $\Psi : L^E \to L^E$.

O MAE de um operador Ψ, MAE(Ψ), é dado pela equação

$$\text{MAE}(\Psi) = E[|g - \Psi(h)|], \tag{2.4}$$

onde $E[\bullet]$ é a esperança de $\bullet$. O MSE é definido de forma semelhante pela equação

$$\mathrm{MSE}(\Psi) = E[|g - \Psi(h)|^2]. \tag{2.5}$$

A diferença básica entre estas medidas está no peso que se dá a cada ponto errado. Isto é, no MAE toma-se a diferença absoluta simples e no MSE toma-se o quadrado da diferença. Portanto, quanto maior o erro, maior é a importância dele na medida MSE. Por causa disso, quando o objetivo é tirar ruído, o MSE pode ser uma medida mais interessante que o MAE, principalmente para sinais. No caso das imagens, há que se levar em conta outros fatores, por exemplo, o embaçamento das arestas, mesmo quando tratamos filtragem de ruído.

Em geral, não temos como calcular exatamente o MAE ou o MSE pois não temos todas as imagens ideais g_i correspondentes à aplicação de Ψ em h_i, para todos as imagens h_i. Porém, podemos estimá-los através de amostras,

As figuras 2.3a e b mostram a imagem do "camera man" e seu embaçamento por uma matriz 5×5 gaussiano (Fig. 2.4a). O MAE e o MSE estimados a partir dessas imagens é, respectivamente, 8.8474 e 319.4225. A figura 2.4b mostra o resultado de um operador linear de desembaçamento (um filtro de Wiener) sobre a imagem embaçada. O MAE e o MSE da imagem restaurada é 6.2192 e 63.5656, respectivamente.

É importante lembrar que não existe uma regra geral sobre qual das duas medidas é melhor e, portanto, onde e quando usá-las. Também não existe uma regra geral que relacione qualquer uma delas com o operador que dá a melhor qualidade visual para uma imagem. A única coisa que é válida, e é óbvia, é que quanto mais próximos estamos da imagem ideal, menor o MAE, ou o MSE. Porém, é fácil encontrar exemplos em que tanto o MAE, quanto o MSE, são pequenos e a imagem é visualmente ruim em relação a imagem ideal (caso comum em restauração de imagens embaçadas).

Estas são as medidas mais usadas neste trabalho, embora existam outras medidas conhecidas [96, 97, 98]. Quando falarmos de segmentação de imagens, introduziremos uma outra medida mais apropriada para essa aplicação.

2.2 Espaços e transformações lineares

Nesta seção recordamos conceitos básicos de Álgebra Linear. Recordaremos a definição de corpo, de espaço vetorial e, finalmente, de uma transformação linear. As principais fontes de consulta para esta seção foram os livros de Hoffman & Kunze [99] e de Larry Smith [100].

Seja F um conjunto de objetos $F = \{x, y, z, w, \ldots\}$ e sejam $+$ e $.$ duas operações definidas sobre F tais que: para todos $x, y \in F$, $x + y \in F$ e $x.y \in F$.

Definição 2.7 *Dizemos que $(F, +, .)$ é um corpo comutativo se as seguintes propriedades são satisfeitas.*

1. A operação $+$ é *comutativa*, i.e., $x + y = y + x, \forall x, y \in F$.
2. A operação $+$ é *associativa*, i.e., $x + (y + z) = (x + y) + z, \forall x, y, z \in F$.
3. Existe um único *elemento neutro* 0 em F, tal que $x + 0 = x, \forall x \in F$.

4. Para cada $x \in F$, existe um único *elemento inverso* $-x \in F$, tal que $x + (-x) = 0$.
5. A operação . é *comutativa*, i.e., $x.y = y.x, \forall x, y \in F$.
6. A operação . é *associativa*, i.e., $x.(y.z) = (x.y).z, \forall x, y, z \in F$.
7. Existe um único elemento não nulo 1 em F tal que $x.1 = x, \forall x \in F$.
8. Para cada $x \in F$, não nulo, existe um único elemento $x^{-1} \in F$, tal que $x.x^{-1} = 1$.
9. A operação . é distributiva em relação à operação +, i.e., $x.(y + z) = x.y + x.z, \forall x, y, z \in F$.

Seja F o conjunto dos números reais ou dos números complexos. Sejam $+$ e $.$ as operações usuais de adição e multiplicação, então F é um corpo comutativo. É comum a nomenclatura *escalar* em lugar de *número* quando tratamos de espaços lineares.

Precisamos da noção de corpo comutativo para definir combinações lineares dos elementos do corpo. A definição a seguir trata do espaço dessas combinações lineares.

Definição 2.8 *Um espaço vetorial constitui-se dos seguintes ítens:*

1. *Um corpo F de escalares.*
2. *Um corpo V de objetos, denominados vetores.*
3. *Uma operação $+$ em V, denominada adição de vetores, tal que $\forall \mathbf{u}, \mathbf{v} \in V$, $\mathbf{u} + \mathbf{v} \in V$. Além disso, a operação $+$ tem que ser comutativa, associativa, existir um único elemento neutro e um único elemento inverso.*
4. *Uma operação $.$, denominada multiplicação escalar, que associa a cada $c \in F$ e a cada vetor $\mathbf{v} \in V$, um vetor $c\mathbf{v}$ em V. Esta operação satisfaz as seguintes propriedades:*
   - *(a)* $1.\mathbf{v} = \mathbf{v}$.
   - *(b)* $(c_1 c_2).\mathbf{v} = c_1(c_2.\mathbf{v})$.
   - *(c)* $c(\mathbf{v} + \mathbf{w}) = c\mathbf{v} + c\mathbf{w}$.
   - *(d)* $(c_1 + c_2)\mathbf{v} = c_1\mathbf{v} + c_2\mathbf{v}$.

Embora a maioria dos fenômenos reais não sejam lineares, usualmente a primeira abordagem em sua modelagem é a linear. Por muitos anos a física, a biologia, a economia e diversas outras ciências têm aplicado a modelagem linear para obter uma aproximação dos fenômenos reais. No caso dos filtros, ou transformações entre imagens ou sinais, a história não foi diferente. Assim, apresentamos a definição de transformação linear.

Definição 2.9 *Sejam V e V' espaços vetoriais sobre o corpo F. Uma transformação linear de V em V' é uma função T que satisfaz:*

$$T(a.\mathbf{u} + b.\mathbf{v}) = a.T(\mathbf{u}) + b.T(\mathbf{v})$$

para todos os escalares $a, b \in F$ e todos os vetores $\mathbf{u}, \mathbf{v} \in V$.

A seguinte proposição será apresentada sem prova por ser trivial.

Proposição 2.10 *Seja A uma matriz $n \times n$ com valores em F, seja $\mathbf{v}$ um vetor $n \times 1$ de V. A transformação definida pela fórmula abaixo é uma transformação linear.*

$$\Psi(\mathbf{v}) = A.\mathbf{v} \tag{2.6}$$

Definição 2.11 *Seja F o corpo dos números reais[3] e seja V um espaço vetorial sobre F. Um produto interno sobre V é uma função que associa a cada par de vetores $\mathbf{u}$ e $\mathbf{v}$ de V, um escalar $\langle \mathbf{u}, \mathbf{v} \rangle$ de F tal que, para todos $\mathbf{u}, \mathbf{v}, \mathbf{w}$ de V e todos os escalares c de F,*

1. $\langle \mathbf{u} + \mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{u}, \mathbf{v} \rangle + \langle \mathbf{v}, \mathbf{w} \rangle$
2. $\langle c.\mathbf{u}, \mathbf{v} \rangle = c.\langle \mathbf{u}, \mathbf{v} \rangle$
3. $\langle \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{v}, \mathbf{u} \rangle$
4. $\langle \mathbf{u}, \mathbf{u} \rangle > 0$ *se* $\mathbf{u} > 0$

Definição 2.12 *Um espaço vetorial munido de um produto interno é chamado de espaço de Hilbert.*

2.2.1 Operador linear sobre imagens

Para nossa próxima definição, vamos ampliar o espaço das imagens para as funções de E no corpo F. Esse artifício é necessário pois os operadores lineares são definidos sobre um espaço vetorial e L^E não forma um espaço vetorial.

Definição 2.13 *Sejam g e h duas imagens de F^E e c_1 e c_2 dois elementos quaisquer do corpo F. Dizemos que Ψ é um operador linear se ele satisfaz:*

$$\Psi(c_1.g + c_2.h) = c_1.\Psi(g) + c_2.\Psi(h) \tag{2.7}$$

Note que, a rigor, **não é possível** definir transformações lineares em L^E[4]. Porém, é comum chamar a melhor aproximação de um operador linear Ψ no espaço dos operadores sobre L^E de operador linear. Se um operador não satisfaz a propriedade de ser linear, então ele é dito ser *não-linear*.

2.2.2 Convolução

A convolução é um dos operadores lineares mais importantes. Seu nome significa[5]: "ato de enrolar junto", que é justamente o que o operador faz; isto é, ele produz uma imagem de saída em que cada ponto é o resultado de uma multiplicação de função chamada *kernel*, ou *suporte*, por uma vizinhança

[3] No caso de F ser o corpo dos números complexos, então a propriedade 3 envolve o complexo conjugado dos escalares.

[4] Pois nenhum subconjunto finito de F pode ser um subcorpo, i.e., um subconjunto de F que também satisfaz as propriedades de ser um corpo.

[5] Em português, ou inglês.

daquele ponto na imagem de entrada, como se ele estivesse "passeando", ou "enrolando", o kernel pela imagem. Formalmente, dada uma imagem $f \in F^E$ e uma função $h \in F^W$, $W \subset E$, a convolução de h por f é dada por:

$$\Psi(f)(x) = \sum_{p \in W} h(p).f(x-p) \tag{2.8}$$

2.2.3 Projeto estatístico de operadores lineares

A idéia de utilizar métodos estatísticos para projetar operadores lineares não é nova e houve um avanço muito grande desde sua origem, tanto teórico quanto prático[101, 102, 103, 92, 104]. Nesta seção recordamos alguns fundamentos da teoria de projeto estatístico de operadores lineares e damos uma interpretação geométrica para a idéia.

2.2.4 Filtros ótimos

O projeto de filtros a partir de medidas estatísticas sobre um conjunto de imagens, e o estudo de como é o erro desses filtros quando aplicados a um conjunto de imagens com características estatisticamente similares ao primeiro, é o assunto que trata a filtragem ótima. Embora seja um assunto muito amplo, nesta seção vamos abordar a essência do projeto de filtros lineares ótimos e mais tarde vamos ver como ele se estende naturalmente para o projeto de operadores morfológicos ótimos.

Essencialmente, o projeto de filtros lineares ótimos está baseado no principio da ortogonalidade, que por sua vez depende da existência de um espaço linear com produto interno, ou espaço de Hilbert. Nos parágrafos que seguem, apresentaremos os filtros lineares discretos ótimos de tamanho restrito. Estes filtros, no limite, isto é, quando a janela é igual ao tamanho do domínio E, são equivalentes ao famoso filtro de Wiener [92, 8].

2.2.5 Filtros lineares ótimos de tamanho restrito

Seja $\mathbf{W} = (W_1, \ldots, W_m)$ um vetor aleatório observado e Y uma variável aleatória a ser estimada.

O problema de projetar um filtro linear ótimo de tamanho restrito Ψ é equivalente a achar uma função ψ que pode ser escrita como combinação linear das variáveis aleatórias que melhor estimam Y relativo ao erro quadrático médio, ou **MSE** (**M**ean **S**quare **E**rror). Em outras palavras, nós queremos achar constantes $a_1, \ldots, a_m \in R$ tais que,

$$MSE(\Psi) = E\left[|Y - \psi(\mathbf{W})|^2\right] = E\left[|Y - \sum_{k=1}^{m} a_k W_k|^2\right],$$

é mínimo. Como ψ (e conseqüentemente Ψ) depende da escolha de $A = (a_1, a_2, \ldots, a_m)$ denotaremos ψ por ψ_A de agora em diante. Como A será estimado, vamos denotá-lo por $\hat{A}$.

Dizemos que $\hat{Y} = \psi_{\hat{A}}(\mathbf{W}) = \sum_{k=1}^{m} \hat{a}_k W_k$ é o filtro linear ótimo se e só se,

$$E[|Y - \hat{Y}|^2] \leq E\left[|Y - \sum_{k=1}^{m} a_k W_k|^2\right],$$

sobre todas as escolhas possíveis de $a_1, a_2, \ldots, a_m$.

Teorema 2.14 *(***Princípio da Ortogonalidade***) Para um vetor aleatório* $\mathbf{W} = (W_1, W_2, \ldots, W_m)'$, *existe um conjunto de constantes que minimizam o MSE como um estimador de* Y *baseado em* $W_1, W_2, \ldots, W_m$ *sobre todas as possíveis escolhas de constantes. Além disso,* $\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_m$ *formam um conjunto mínimo se e só se, para quaisquer constantes* $a_1, a_2, \ldots, a_m$,

$$E\left[\left(Y - \sum_{k=1}^{m} \hat{a}_k W_k\right)\left(\sum_{l=1}^{m} a_l W_l)\right)\right] = 0$$

significando que $Y - \hat{A}.W$ *é ortogonal a* $A.W$.

Pode-se provar que, se $W_1, \ldots, W_m$ são linearmente independentes, então a coleção de constantes de otimização é única.

De acordo com o princípio de ortogonalidade, $\hat{Y}$ é o melhor estimador linear MSE de Y baseado em $W_1, \ldots, W_m$ se e só se,

$$E\left[\left(Y - \hat{Y}\right)\left(\sum_{l=1}^{m} a_l W_l\right)\right] = \sum_{l=1}^{m} a_l E\left[(Y - \hat{Y})W_l\right] = 0$$

para quaisquer constantes $a_1, \ldots, a_m$.

Esta igualdade pode ser resolvida se $E[(Y - \hat{Y})W_l] = 0$ para todo l, $1 \leq l \leq m$, i.e., temos que resolver,

$$E\left[\left(Y - \hat{Y}\right)W_l\right] = E\left[\left(Y - \sum_{k=1}^{m} \hat{a}_k W_k\right)W_l\right] = E[Y W_l] - \sum_{k=1}^{m} \hat{a}_k E[W_k W_l] = 0$$

para todo l, $1 \leq l \leq m$.

A condição acima conduz a um sistema de equações da forma,

$$\begin{array}{l} R_{11}\hat{a}_1 + R_{12}\hat{a}_2 + \cdots + R_{1m}\hat{a}_m = R_1 \\ R_{21}\hat{a}_1 + R_{22}\hat{a}_2 + \cdots + R_{2m}\hat{a}_m = R_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ R_{m1}\hat{a}_1 + R_{m2}\hat{a}_2 + \cdots + R_{mm}\hat{a}_m = R_m \end{array}$$

onde $R_{ij} = E[W_i W_j] = E[W_j W_i] = Rji$ e $R_k = E[Y W_k]$.

Este sistema de equações pode ser escrito em forma matricial $R\hat{A} = C$, se nós fizermos $\hat{A} = (\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_m)'$, $C = (R_1, R_2, \ldots, R_m)'$, e

$$R = \begin{pmatrix} R_{11} & R_{12} & \cdots & R_{1m} \\ R_{21} & R_{22} & \cdots & R_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ R_{m1} & R_{m2} & \cdots & R_{mm} \end{pmatrix}.$$

C é o *vetor de correlação cruzada* e R a *matriz de auto-correlação* de $W_1, W_2, \ldots, W_m$.

Se $det[R] \neq 0$ (e é pois W_i são independentes), então a solução do sistema é $\hat{A} = R^{-1}C$, e o filtro linear pode ser escrito como,

$$\Psi = \mathbf{W}.\hat{A}$$

Chamamos este filtro de: *Filtro Linear Ótimo Restrito* pois ele é um filtro linear ótimo de tamanho finito dado pela dimensão de A, $|A|$.

Os filtros ótimos de tamanho restrito são uma aproximação dos filtros de Wiener [8]. Quanto maior a janela, melhor a aproximação. Usamos os primeiros ao invés do filtro de Wiener pois podíamos integrar mais facilmente com o projeto de operadores "aperture".

2.2.6 Interpretação geométrica

A teoria dos filtros lineares MSE ótimos tem uma interpretação geométrica didática. Como vimos, o conjunto de variáveis aleatórias $W_1, W_2, \ldots, W_n, Y$, e também as combinações lineares entre elas, formam um espaço vetorial.

A esperança é um operador linear definido neste espaço e a esperança do produto de dois vetores, $E[W_i W_j]$, é um produto interno. Assim, dois vetores W_i e W_j são ortogonais se $E[W_i W_j] = 0$. Isso só acontece se $i \neq j$ (pois assumimos que as variáveis W_i $1 \leq i \leq n+1$ são independentes). Desta última relação vem o nome do princípio da ortogonalidade. A Fig. 2.5 ilustra estes conceitos para três variáveis independentes, W_1, W_2 e Y.

Como Y não é uma combinação linear de W_1, W_2, então Y não pertence ao subespaço vetorial gerado por esses vetores. Por outro lado, um vetor Y' definido por $Y' = a_1.W_1 + a_2.W_2$ pertence ao subespaço. Assim, a questão que se coloca é: quando o vetor $(Y - Y')$ é mínimo? Isto é, para que valores de a_1 e a_2 o vetor Y' mais se aproxima de Y? A resposta é simples, quando $(Y - Y')$ é um vetor ortogonal a Y' (veja Fig. 2.5).

Finalmente, note que o produto interno de $(Y - Y')$ por ele mesmo pode ser escrito como (compare com o teorema da ortogonalidade)

$$E[(Y - Y')(Y - Y')] = E[Y(Y - Y')]$$

pois Y' e $(Y - Y')$ são componentes ortogonais de Y.

2.3 Fundamentos de morfologia matemática

Nesta seção, apresentamos alguns conceitos básicos que irão ser necessários ao longo do trabalho. Primeiramente recordamos os conceitos de reticulado completo e sua representação diagramática.

2.3.1 Reticulados e representação em diagramas

Sejam L e E definidos como anteriormente. Seja $\leq$ uma relação de ordem [105] entre os elementos de L^E, dada por:

$$f \leq g \Longleftrightarrow f(x) \leq g(x), \forall x \in E$$

É fácil mostrar que ela é uma relação de ordem parcial, i.e., que satisfaz, para todo f, g e h em L^E as propriedades: $f \leq f$; $(f \leq g \text{ e } g \leq h \Longrightarrow f \leq h)$; $(f \leq g \text{ e } g \leq f) \Longrightarrow f = g$.

A definição a seguir será muito importante num algoritmo que apresentaremos no capítulo 4.

Definição 2.15 *Seja $\mathcal{L}$ um "poset" [105], i.e., um conjunto com uma relação de ordem parcial denotada por $\leq$. Dados a e b em $\mathcal{L}$, dizemos que b cobre a se $a < b$ e não existe $x \in \mathcal{L}$ tal que $a < x < b$.*

A noção acima permite-nos construir uma representação para os conjuntos parcialmente ordenados conhecida como "Diagrama de Hasse", que será usado em diversos lugares ao longo do texto. O diagrama é construído em duas etapas. Primeiro desenha-se um círculo para cada elemento do "poset", posicionando b mais alto que a sempre que $a < b$. Em seguida, desenha-se uma linha ligando todos os elementos a e b, se e somente se b cobre a. A figura 2.6 ilustra estes conceitos.

O *limite superior* de um subconjunto $\mathcal{X} \subset \mathcal{L}$ é um elemento de $\mathcal{L}$ que cobre todos os elementos $x \subset \mathcal{X}$. O *menor limite superior* é um limite superior coberto por todos os outros limites superiores. De forma dual defini-se o *maior limite inferior*.

Definição 2.16 *Seja $\mathcal{L}$ um "poset". Dizemos que $\mathcal{L}$ é um reticulado se, para quaisquer dois elementos de $\mathcal{L}$ existe um maior limite inferior e um menor limite superior. Um reticulado é completo quando cada um dos seus subconjuntos têm um maior limite inferior e um menor limite superior em $\mathcal{L}$.*

Proposição 2.17 *Se $\mathcal{L}$ é um reticulado finito, por exemplo $\mathcal{L} = L^E$, onde $L = Z \cup \{-\infty, \infty\}$ e $E = Z \cup \{-\infty, \infty\}$ para sinais, ou $E = (Z \cup \{-\infty, \infty\}) \times (Z \cup \{-\infty, \infty\})$ para imagens, então $\mathcal{L}$ é um reticulado completo.*

A proposição acima foi provada por Banon e Barrera [21, 13]. Assim, o conjunto das imagens digitais forma um reticulado completo, e uma imagem $f \in L^E$ é um elemento do reticulado. A figura 2.7a ilustra o reticulado das imagens binárias de três pontos, isto é, o reticulado $\{0,1\}^{\{e_1,e_2,e_3\}}$. Cada ponto do reticulado representa uma imagem binária de três pontos e todas as possíveis imagens binárias de três pontos estão representadas no reticulado, isto é, 2^3 imagens. A figura 2.7b ilustra o reticulado das imagens em níveis de cinza com dois pontos e 5 níveis de cinza, isto é, o reticulado $[-2,2]^{\{e_1,e_2\}}$.

2.3.2 Operações no espaço das imagens

Duas operações importantes no espaço de imagens são a translação espacial e a translação nos níveis de cinza da imagem.

Definição 2.18 *Seja $f \in L^E$ uma imagem definida sobre um espaço E que é um grupo abeliano em relação a operação de adição em E, $+$, e seja $t \in E$. A translação espacial de f por t é definida por:*

$$f_t(x) = f(x - t), \forall x \in E. \tag{2.9}$$

Definição 2.19 *A translação vertical de $f \in L^E$ por z, onde $z \in L$, é definida por:*

$$(f + z)(x) = f(x) + z, \forall x \in E. \tag{2.10}$$

Note que f está definida como uma função de E em Z e que a operação + agora é a soma de números inteiros.

Definição 2.20 *A translação de um subconjunto de $W \subset E$ por $t \in E$, denotada por W_t e definida por:*

$$W_t = \{x \in E, x + t \in W\} \tag{2.11}$$

Duas operações básicas da morfologia matemática são a intersecção e a união de dois elementos do reticulado.

Definição 2.21 *Sejam $f, g \in L^E$. As operações de intersecção, denotada por $\wedge$, e união, denotada por $\vee$, de f e g são definidas, respectivamente, por:*

$$(f \wedge g)(x) = \min\{f(x), g(x)\}, \forall x \in E. \tag{2.12}$$

e

$$(f \vee g)(x) = \max\{f(x), g(x)\}, \forall x \in E. \tag{2.13}$$

Os resultados dessas operações são chamados, respectivamente, de *ínfimo* e *supremo*.

Outra operação básica da morfologia matemática é o complemento.

Definição 2.22 *Seja $f \in L^E$, e k o maior elemento de L. A operação complemento é definida por,*

$$\nu(f(x)) = k - f(x), \forall x \in E \tag{2.14}$$

O resultado desta operação é chamado de *negação*.

2.3.3 Operadores morfológicos

Nesta seção recordamos a dilatação e a erosão morfológicas por um conjunto B de E denominado *elemento estruturante*.

Definição 2.23 *Seja f uma imagem de L^E e $B \subset E$. A dilação de f por B, denotada por $\delta_B(f)$, e dada por:*

$$\delta_B(f)(x) = \bigvee_{y \in B^t + x} f(x + y) \tag{2.15}$$

Onde $\bigvee$ é o símbolo para indicar que estamos tomando o valor máximo do conjunto.

Definição 2.24 *Seja f uma imagem de L^E e $B \subset E$. A erosão de f por B, denotada por $\delta_B(f)$, e dada por:*

$$\varepsilon_B(f)(x) = \bigwedge_{y \in B + x} f(x + y) \tag{2.16}$$

Onde $\bigwedge$ é o símbolo para indicar que estamos tomando o valor mínimo do conjunto.

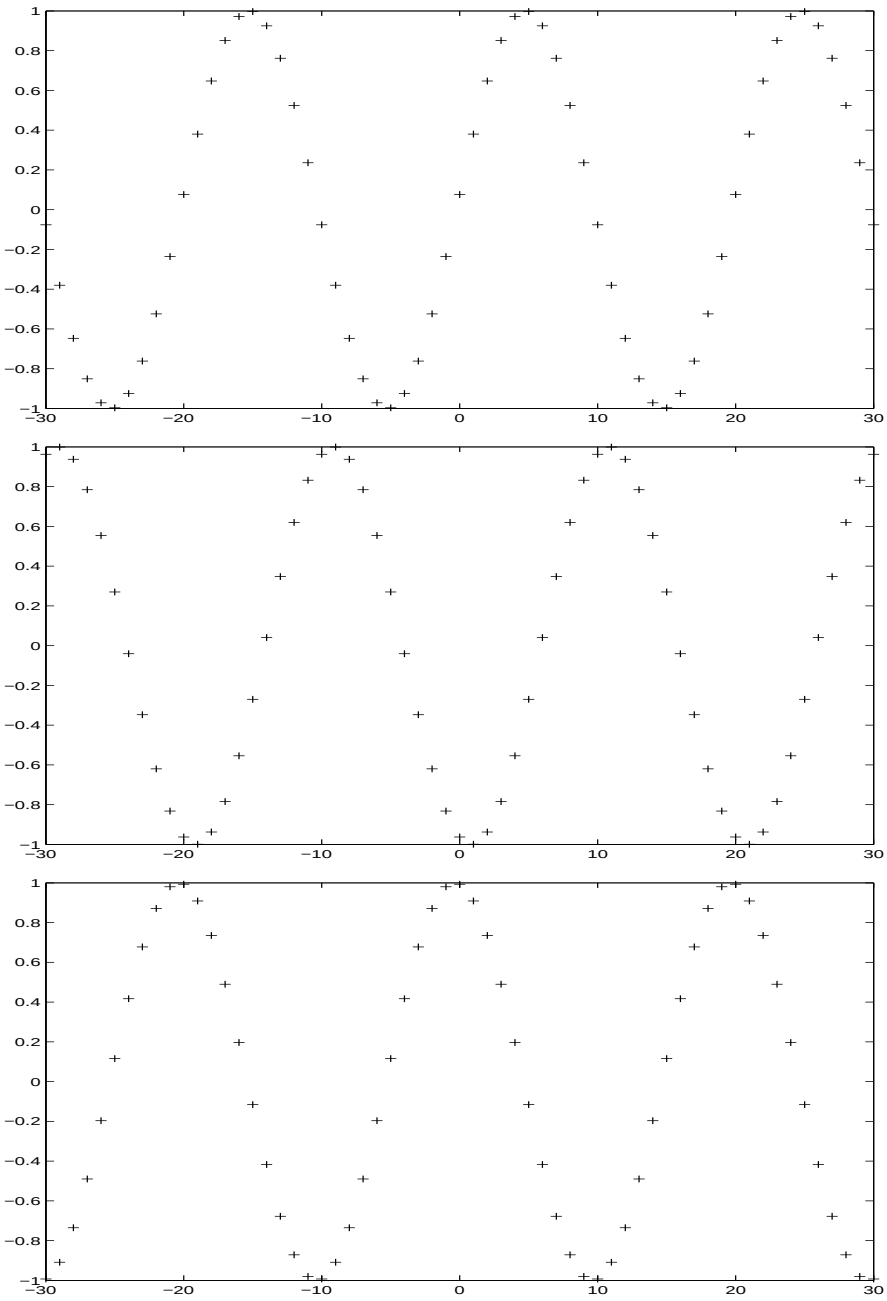


Figura 2.1: Realizações de um processo aleatório discreto

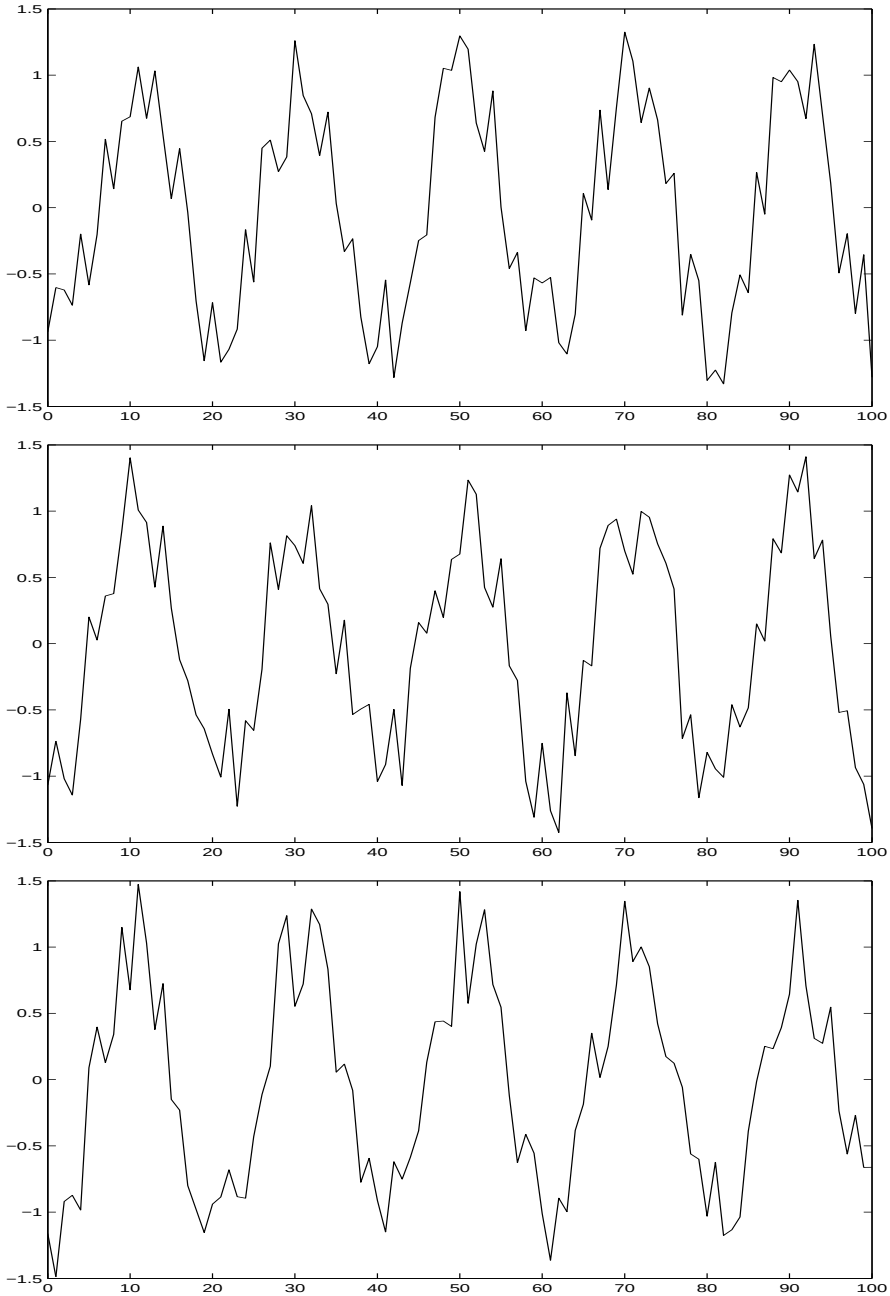
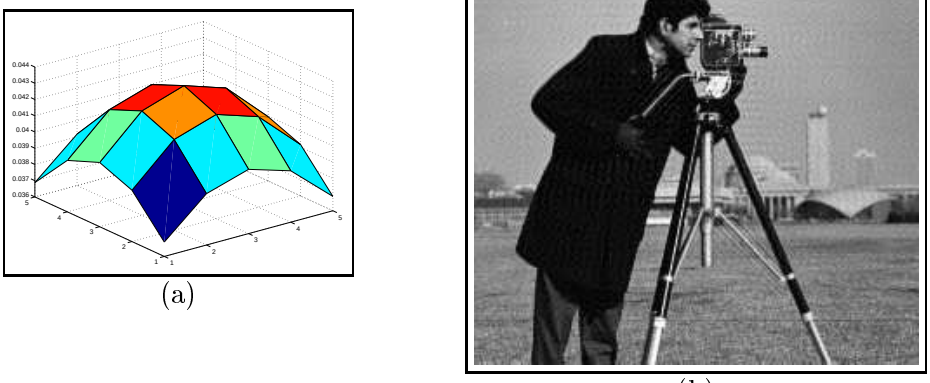


Figura 2.2: Realizações de um processo com ruído aleatório

(a) (b)

Figura 2.3: “Camera man” (a) e seu embaçamento (b)



(a) (b)

Figura 2.4: Matriz de convolução (a) e a restauração (b)

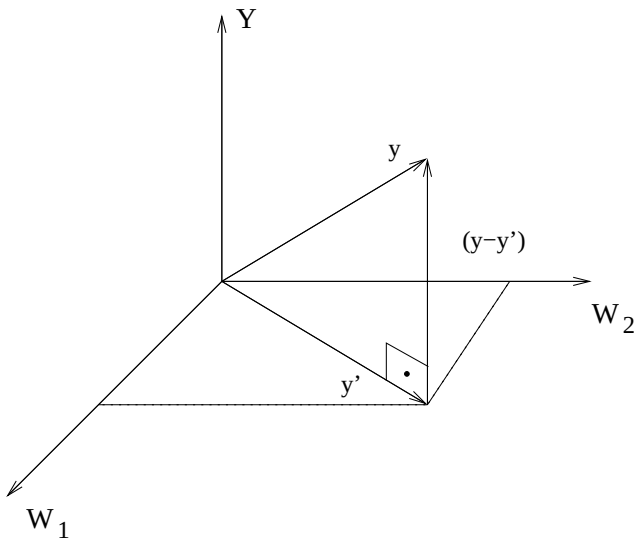


Figura 2.5: Interpretação geométrica

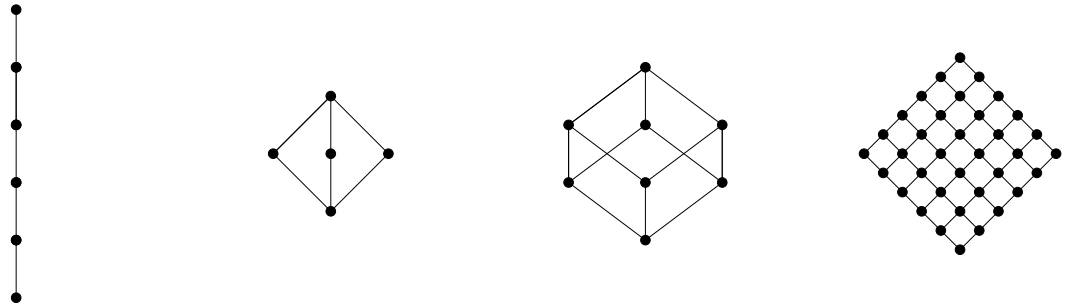


Figura 2.6: Diagramas de Hasse

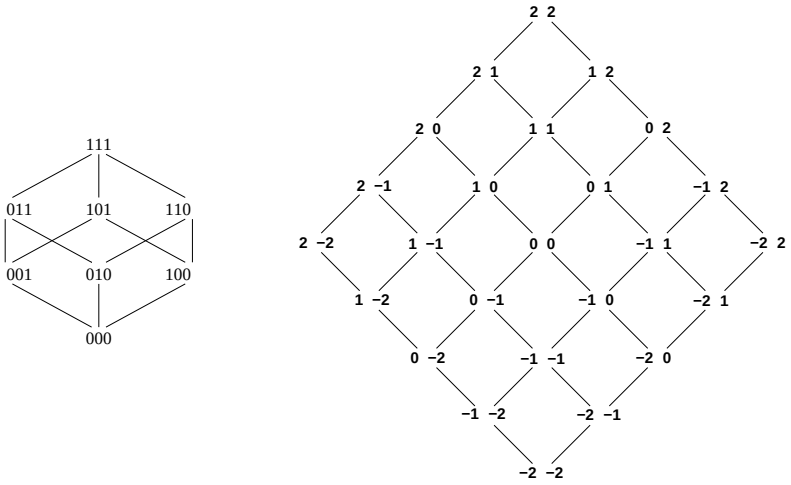


Figura 2.7: Dois reticulados de imagens.

Capítulo 3

W-operadores e seu projeto

O projeto heurístico de operadores entre imagens é uma tarefa difícil e que requer muita experiência e conhecimento do usuário. Assim, não é qualquer usuário que está capacitado para resolver seus problemas práticos com técnicas de processamento de imagens, muito embora talvez fosse a maneira mais cômoda de fazê-lo. Por exemplo, um biólogo que queira contar células marcadas radioativamente, pode beneficiar-se tanto em qualidade de seu trabalho, quanto em precisão, se puder usar técnicas de processamento de imagens que segmentam e contam as células de interesse; principalmente se a quantidade de imagens for grande. Como este profissional não necessariamente possui conhecimentos para combinar os operadores de imagens corretamente, o desejável seria que ele dispusesse de um sistema em que: ele pudesse editar (i.e., segmentar, filtrar, rotular, etc.) manualmente, via uma interface gráfica, os objetos de interesse em uma, ou mais, imagens; ele pudesse submeter essas imagens segmentadas para o sistema projetar um operador de imagens que imitasse o trabalho que ele fez manualmente; ele pudesse processar tantas imagens quanto quisesse rapidamente; finalmente, ele pudesse reforçar o treinamento dando mais exemplos segmentados manualmente, ou consertando eventuais erros que o sistema cometeu.

Um sistema deste tipo não é simples. Ele é composto por várias partes e este trabalho produziu duas delas, quais sejam, o subsistema de *projeto* e o subsistema de *aplicação* de operadores de imagens. O subsistema de projeto serve para estimar o operador a partir de um par, ou mais, de imagens (por exemplo, imagem das células, e a respectiva imagem das células marcadas) e, também, obter uma representação computacional compacta para o operador. O subsistema de aplicação serve para transformar imagens a partir da representação computacional do operador projetado.

O projeto estatístico de operadores de janela requer a estimação das probabilidades condicionais (probabilidade de ocorrer um certo nível de cinza, y, dado que foi observada uma certa configuração de janela $\mathbf{x}$) para cada uma das possíveis configurações, quando restringimos adequadamente o tamanho da janela W. Essa estimação se dá a partir de exemplos de treinamento fornecidos pelo usuário via pares de imagens, ou via tabelas de exemplos de treinamento. Como exemplificado acima, em geral, a forma mais cômoda para o usuário é fornecer pares de imagens em que ele produz a saída desejada para cada imagem de treinamento observada.

Neste capítulo trataremos do projeto de W-operadores usando conceitos de estimação estatística [32] e de teoria de decisão [106]. A idéia, do lado estatístico, tem raízes nos trabalhos de Wiener [101, 102] e Pugachev [103], portanto tem semelhanças com o projeto estatístico de filtros lineares. Do lado morfológico, a idéia tem raízes nos trabalhos de decomposição de Matheron [15] e de Banon e Bar-

rera [18, 21]. Dougherty foi quem concebeu a idéia e mostrou como juntar as duas abordagens para fazer o projeto de W-operadores [51, 61]. Não podemos deixar de lembrar que assumimos por hipótese que os processos envolvidos na formação dos sinais, ou imagens, de interesse são aleatórios e estacionários. Partindo dela, podemos estimar o núcleo de um operador invariante por translação no espaço.

3.1 Operadores de janela

Quando um operador de imagens é localmente definido por uma janela W, isto é, quando basta olharmos para os pontos dentro de W para decidir o valor da saída do operador, dizemos que ele é um operador de janela. Nesta seção, apresentaremos a noção de janelamento espacial e a definição de W-operadores.

3.1.1 Configurações de janela

Um subconjunto finito W de E será chamado de *janela* e a sua *dimensão*, isto é, o número de pontos de W, será denotada por $|W|$. O símbolo L será usado para designar uma cadeia, como no capítulo 2.

Uma *configuração de janela*, ou simplesmente *configuração*, é uma função de W em L. O espaço de todas as possíveis configurações de W em L será denotado por L^W, e a quantidade de possíveis configurações é dada por $|L|^{|W|}$ (pois cada ponto da janela pode assumir $|L|$ valores) onde $|L|$ é a quantidade de níveis de cinza em L.

Configurações de janela não precisam necessariamente ter relação com imagens. Quando elas não têm, são geralmente chamadas de *padrões*. Normalmente as configurações de janela são obtidas transladando uma imagem sob uma janela e observando as configurações que vão "passando" sob a janela. Uma outra maneira de obtê-las é transladando a janela sobre a imagem e observando as configurações que vão "passando" sob a janela. Essas duas formas de se obter configurações a partir de uma imagem são equivalentes.

Seja $f \in L^E$ uma imagem em níveis de cinza e $W \subset E$ uma janela tal que $W = \{x_1, x_2, \ldots, x_n\}$, $x_i \in E$ e $n = |W|$. Denominamos por *restrição* de f a W no ponto t, $t \in E$, a configuração $f_{-t}|W : W \to L$ definida por (o sinal negativo em t deve-se ao fato que estamos transladando a função para a origem):

$$f_{-t}|W = (f(x_1 + t), f(x_2 + t), \ldots, f(x_n + t)). \tag{3.1}$$

Assim, variando t para todos os pontos de E, podemos obter todas as configurações vistas por uma janela W.

A segunda maneira usa a definição do translado de W por t, $t \in E$. Seja $W = \{x_1, x_2, \ldots, x_n\}$, $x_i \in E$ e $n = |W|$, então $W_t = (x_1 + t, x_2 + t, \ldots, x_n + t)$ é o translado de W por t. Uma configuração $f|W_t : W \to L$ é descrita por:

$$f|W_t = (f(x_1 + t), f(x_2 + t), \ldots, f(x_n + t)). \tag{3.2}$$

Novamente, variando t para todos os pontos de E obtemos todas as configurações vistas pela janela sobre f. Como podemos ver, as duas formas $f_{-t}|W$ e $f|W_t$ são equivalentes.

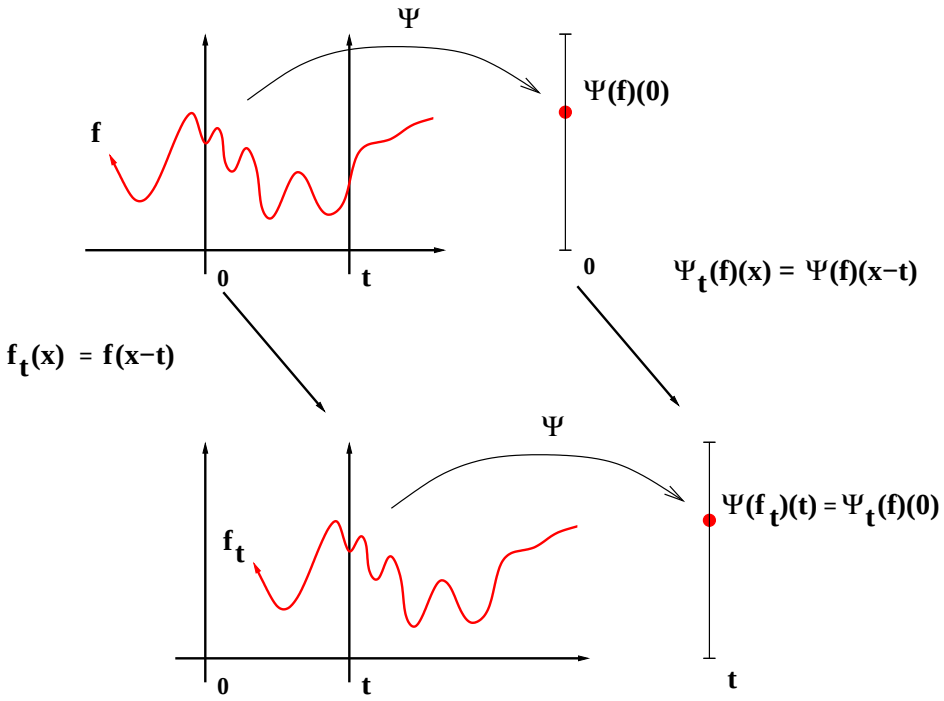


Figura 3.1: Invariância por translação.

Exemplo 3.1 *Vejamos um exemplo de como essas configurações são obtidas. Seja* $W = \{(-1,0),(0,0),(1,0)\}$*, onde* $(0,0)$ *é a origem de* E*. Seja* f *uma função de* L^E *e seja* $t = (t_x, t_y)$ *um ponto de* E*. Pela equação 3.1, temos:* $f_{-t}|W = (f(-1+t_x, 0+t_y), f(0+t_x, 0+t_y), f(1+t_x, 0+t_y)) = (f(-1+t_x, t_y), f(t_x, t_y), f(1+t_x, t_y))$*. Pela equação 3.2, temos:* $W_t = \{(-1+t_x, t_y), (t_x, t_y), (1+t_x, t_y)\}$*, e* $f|W_t = (f(-1+t_x, t_y), f(t_x, t_y), f(1+t_x, t_y))$*.*

3.1.2 Operadores invariantes por translação espacial

Os operadores invariantes por translação espacial, também denominados H-operadores [14], formam uma classe importante dentro da classe dos operadores lineares[92]. No caso não linear, eles também são muito importantes pois, para a maioria dos problemas de filtragem, basta que os operadores sejam invariantes por translação espacial. Dentre os casos clássicos em que esses operadores não são adequados (embora isso dependa também do problema), está o caso em que as imagens têm uma perspectiva muito pronunciada, por exemplo, cenas em que objetos de tamanhos iguais parecem ter tamanhos diferentes por causa da perspectiva. Outro caso é o de imagens que têm características muito diferentes dependendo do lugar para o qual se olha a imagem.

Um operador $\Psi : L^E \to L^E$ é chamado *invariante por translação* (i.t.) no espaço se satisfaz a seguinte relação:

$$\Psi(f_t)(x) = \Psi(f)(x-t) \quad \forall t \in E, \ \forall f \in L^E. \tag{3.3}$$

Em outras palavras, aplicar o operador sobre a função transladada é equivalente a aplicar o operador sobre a função e depois transladar o resultado. A figura 3.1 ilustra graficamente um exemplo deste conceito.

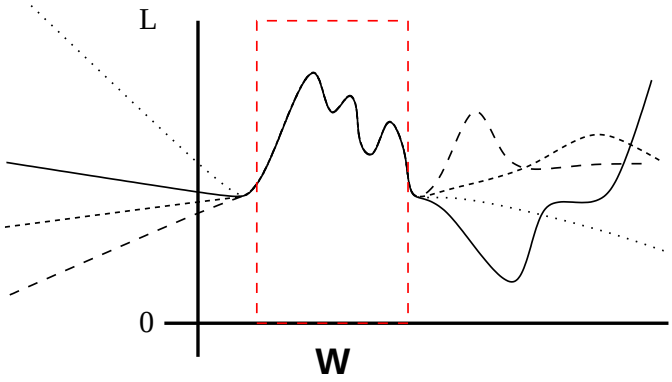


Figura 3.2: Classe de restrição.

3.1.3 Operadores localmente definidos

O projeto estatístico de operadores de imagens é de interesse se os operadores puderem ser definidos localmente por uma janela W não muito grande [1]. Nesta seção definimos melhor estes operadores.

Dizemos que um operador binário $\Psi : \mathcal{P}(E) \to \mathcal{P}(E)$, onde $\mathcal{P}(E)$ é o conjunto das partes de E [14, 13], é *localmente definido* (l.d.) por uma janela W se e somente se $x \in \Psi(X) \iff x \in \Psi(X \cap W_{-x})$ ou, equivalentemente, se $x \in \Psi(X) \iff o \in \Psi(X_{-x} \cap W)$, onde o é a origem de E.

Desta forma, no caso binário, a propriedade de um operador ser l.d. é definida baseada em intersecções. No caso dos operadores em níveis de cinza é preciso generalizar esta idéia.

Definição 3.2 *A classe de restrição de f sobre W, denotada $\mathcal{F}_{f|W}$, é a família de funções cuja restrição a W resulta em $f|W$, isto é:*

$$\mathcal{F}_{f|W} = \{g \in L^E : f|W = g|W\}. \tag{3.4}$$

A figura 3.2 ilustra o conceito mostrando quatro curvas (sólida, tracejada, pontilhada e tracejada com traço curto) diferentes que são iguais quando restritas a W.

Definição 3.3 *Seja L uma cadeia. Um operador de imagens $\Psi : L^E \to L^E$ é chamado localmente definido (l.d.) em W se e só se, para quaisquer $f \in L^E$ e $x \in E$,*

$$\Psi(f)(x) = \Psi(g)(x), \forall g \in \mathcal{F}_{f_{-x}|W}. \tag{3.5}$$

3.1.4 W-operadores

Um operador que é invariante por translação no espaço e localmente definido por uma janela W é chamado de *W-operador*.

[1] O conceito de grande depende de vários fatores, por exemplo: número de imagens disponíveis para fazer a estimação, tamanho da memória disponível, velocidade do computador.

Uma propriedade importante dos W-operadores é que eles podem ser definidos canonicamente por funções de L^W em L chamadas de *medidas* ou *funções características* [2] [21, 62, 107, 68, 108]. Mais ainda, é possível provar que existe uma bijeção entre o conjunto de todas as funções características de L^W em L e o conjunto dos W-operadores [108].

Em outras palavras, dado um W-operador $\Psi : L^E \to L^E$, existe uma, e apenas uma, função característica $\psi : L^W \to L$ tal que:

$$\Psi(f)(x) = \psi(f_{-x}|W), \forall x \in E. \tag{3.6}$$

Usar uma função característica para definir Ψ é uma forma equivalente de definir um W-operador muito usada por Dougherty [109, 66, 67, 107].

O número de operadores dentro da classe dos W-operadores é $|L|^{(|L|^{|W|})}$, pois a cada configuração (dentre as $|L|^{|W|}$ possíveis configurações) pode-se associar um valor de saída diferente (dentre os $|L|$ possíveis níveis de cinza).

3.2 Caracterização de W-operadores

Nesta seção, apresentamos a caracterização dos W-operadores via intervalos do reticulado. Sejam $a, b \in L^W$. Dizemos que a é menor ou igual a b, $a \leq b$, sse $a(x) \leq b(x)$, $\forall x \in W$.

Definição 3.4 *Sejam $a, b \in L^W$ tais que $a \leq b$. O subconjunto de L^W dado por:*

$$[a, b] = \{u \in L^W : a \leq u \leq b\} \tag{3.7}$$

é chamado de intervalo de extremidades a e b.

Definição 3.5 *Sejam $a, b, c, d \in L^W$ tais que $a \leq b$ e $c \leq d$. Dizemos que o intervalo $[c, d]$ está contido no intervalo $[a, b]$, $[c, d] \subset [a, b]$, sse $a \leq c \leq d \leq b$.*

Definição 3.6 *Uma função $\lambda_{a,b}$ de L^W em $\{0, 1\}$ dada por:*

$$\lambda_{a,b}(u) = \begin{cases} 1 & \text{se } u \in [a, b], \\ 0 & \text{caso contrário,} \end{cases} \tag{3.8}$$

para qualquer $u \in L^W$, é chamada de função característica sup-geradora.

Definição 3.7 *Seja uma função característica ψ de L^W em L, e um valor $y \in L$ qualquer. O núcleo de uma função característica é dado por:*

$$\mathcal{K}(\psi)(y) = \{u \in L^W : y \leq \psi(u)\} \tag{3.9}$$

Exemplo 3.8 *Seja L um intervalo de Z de seis níveis, $L = [0, 5]$, e seja W uma janela de dois pontos de E, $W = \{w_1, w_2\}$. Seja Ψ o operador média [3] de L^W em L. A figura 3.3 mostra o resultado do operador (em caracteres claros) aplicado a todos os pontos do reticulado L^W. O núcleo de Ψ é dado por:*

[2] Quando L é finito, elas também são chamadas de *funções computacionais*.

[3] Como L é uma cadeia discreta, usamos o valor da média é truncada.

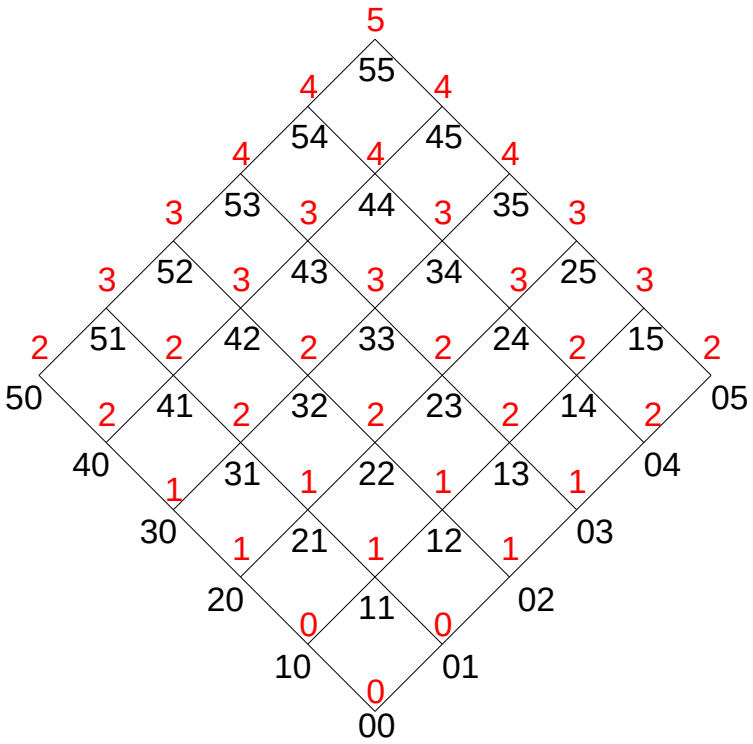


Figura 3.3: Resultado do operador média.

- $\mathcal{K}(\psi)(0) = \left\{ \begin{array}{l} (0,0),(0,1),(0,2),(0,3),(0,4),(0,5), \\ (1,0),(1,1),(1,2),(1,3),(1,4),(1,5), \\ (2,0),(2,1),(2,2),(2,3),(2,4),(2,5), \\ (3,0),(3,1),(3,2),(3,3),(3,4),(3,5), \\ (4,0),(4,1),(4,2),(4,3),(4,4),(4,5), \\ (5,0),(5,1),(5,2),(5,3),(5,4),(5,5) \end{array} \right\}$

- $\mathcal{K}(\psi)(1) = \left\{ \begin{array}{l} (0,2),(0,3),(0,4),(0,5), \\ (1,1),(1,2),(1,3),(1,4),(1,5), \\ (2,0),(2,1),(2,2),(2,3),(2,4),(2,5), \\ (3,0),(3,1),(3,2),(3,3),(3,4),(3,5), \\ (4,0),(4,1),(4,2),(4,3),(4,4),(4,5), \\ (5,0),(5,1),(5,2),(5,3),(5,4),(5,5) \end{array} \right\}$

- $\mathcal{K}(\psi)(2) = \left\{ \begin{array}{l} (0,4),(0,5), \\ (1,3),(1,4),(1,5), \\ (2,2),(2,3),(2,4),(2,5), \\ (3,1),(3,2),(3,3),(3,4),(3,5), \\ (4,0),(4,1),(4,2),(4,3),(4,4),(4,5), \\ (5,0),(5,1),(5,2),(5,3),(5,4),(5,5) \end{array} \right\}$

- $\mathcal{K}(\psi)(3) = \left\{ \begin{array}{l} (1,5), \\ (2,4),(2,5), \\ (3,3),(3,4),(3,5), \\ (4,2),(4,3),(4,4),(4,5), \\ (5,1),(5,2),(5,3),(5,4),(5,5) \end{array} \right\}$
- $\mathcal{K}(\psi)(4) = \left\{ \begin{array}{l} (3,5), \\ (4,4),(4,5), \\ (5,3),(5,4),(5,5) \end{array} \right\}$
- $\mathcal{K}(\psi)(5) = \{\ (5,5)\ \}$

O conceito de núcleo de uma função é importante pois existe uma bijeção entre funções características e núcleos, isto é, basta dar o núcleo da função característica para definí-la completamente. Note que o núcleo de uma função característica tem tantas camadas quantos forem os níveis de cinza do seu contra-domínio. O teorema a seguir apresenta a forma de representar a função característica via seu núcleo [21, 62].

Teorema 3.9 *Se ψ é uma função característica de L^W em L, então ela pode ser representada por:*

$$\psi(u) = \bigvee \{y \in L : \bigvee \{\lambda_{a,b}(u) : [a,b] \subseteq \mathcal{K}(\psi)(y)\} = 1\} \tag{3.10}$$

para qualquer configuração $u \in L^W$.

A representação pelo núcleo do operador é, no entanto, pouco econômica e o recomendável é fazer a representação por intervalos do núcleo. Em geral, usa-se a representação por intervalos maximais do núcleo do operador. Esta maneira de representar o operador é, em geral, mais econômica.

Definição 3.10 *Seja $[a,b]$ um intervalo de $\mathbf{I}$, $\mathbf{I} \subseteq L^W$. Dizemos que $[a,b]$ é maximal em $\mathbf{I}$ sse não existe $[a',b'] \subseteq \mathbf{I}$ tal que $[a,b] \subseteq [a',b']$.*

Definição 3.11 *Seja ψ uma função característica definida de L^W em L, e $y \in L$. A base de uma função característica ψ no nível y, $\mathbf{B}(\psi)(y)$, é dada por:*

$$\mathbf{B}(\psi)(y) = Max(\{[a,b] \subseteq L^W : [a,b] \subseteq \mathcal{K}(\psi)(y)\}), \tag{3.11}$$

onde Max($\bullet$) é o conjunto de todos os intervalos maximais de $\bullet$.

A representação da função característica via sua base é definida pelo teorema abaixo [21, 62].

Teorema 3.12 *Se ψ é uma função característica de L^W em L, então ela pode ser representada por:*

$$\psi(u) = \bigvee \{y \in L : \bigvee \{\lambda_{a,b}(u) : [a,b] \in \mathbf{B}(\psi)(y)\} = 1\} \tag{3.12}$$

para qualquer configuração $u \in L^W$.

Exemplo 3.13 *Seja L um intervalo de Z de seis níveis, $L = [0,5]$, e seja W uma janela de dois pontos de E, $W = \{w_1, w_2\}$. Seja Ψ o operador média de L^W em L. A base de Ψ é dada por:*

- $\mathbf{B}(\psi)(0) = \{[(0,0),(5,5)]\}$
- $\mathbf{B}(\psi)(1) = \{[(0,2),(5,5)],[(2,0),(5,5)],[(1,1),(5,5)]\}$
- $\mathbf{B}(\psi)(2) = \{[(0,4),(5,5)],[(4,0),(5,5)],[(1,3),(5,5)],[(3,1),(5,5)],[(2,2),(5,5)]\}$
- $\mathbf{B}(\psi)(3) = \{[(1,5),(5,5)],[(5,1),(5,5)],[(2,4),(5,5)],[(4,2),(5,5)],[(3,3),(5,5)]\}$
- $\mathbf{B}(\psi)(4) = \{[(3,5),(5,5)],[(5,3),(5,5)],[(4,4),(5,5)]\}$
- $\mathbf{B}(\psi)(5) = \{[(5,5),(5,5)]\}$

A figura 3.4 mostra uma representação da base do operador média onde, de baixo para cima, estão representados todos os intervalos maximais para cada nível da base (de 0 a 5).

3.2.1 Usando a representação para aplicar ψ

A aplicação do operador representado por ψ é uma conseqüência imediata da fórmula 3.12. Dada uma configuração, o algoritmo percorre a lista de intervalos da base de cada nível de cinza, até encontrar um intervalo que a contenha. As bases são percorridas do maior nível de cinza para o menor pois queremos o maior nível de cinza cuja base contenha a configuração.

3.3 Representação por árvores de decisão

Nesta seção iremos caracterizar a representação de uma função característica por uma árvore de decisão (**DT**, do inglês "decision tree").

Uma *árvore de decisão* (DT) é uma árvore enraizada [44] onde cada nó interno, inclusive a raiz, é associado a um *teste*, ou *decisão*, e cada folha (nó externo) é associada a um valor (inteiro ou real) ou a um *conceito*. O teste, ou decisão, é uma função das variáveis ou do conceito associado àquele nó e dependendo do resultado do teste, um dos ramos da árvore é escolhido (para a esquerda se o teste é verdadeiro e para a direita se o teste é falso). Uma *árvore de decisão discreta* é uma árvore de decisão onde cada folha é associada a um número inteiro v, $v \in L, L \subset Z$, finito. De agora em diante, quando falarmos de árvores de decisão, estaremos falando de árvores de decisão discretas. A figura 3.5 ilustra uma árvore de decisão que representa a função média truncada sobre uma janela de dois pontos, onde cada ponto pode assumir os valores 0, 1, 2, ou 3.

A proposição básica deste capítulo diz que uma função característica $\psi : L^W \to L$ qualquer pode ser representado por uma árvore de decisão discreta. Neste sentido, uma árvore de decisão é uma representação de uma função discreta. Porém, esta representação não é canônica, isto é, existe mais de uma árvore de decisão para cada função característica discreta. A figura 3.6 ilustra uma outra árvore de decisão que representa a mesma função que a árvore da figura 3.5.

Teorema 3.14 *Qualquer função característica ψ, $\psi : L^W \to L$, definida entre conjuntos discretos pode ser representado como uma árvore de decisão, e vice-versa.*

A prova deste teorema pode ser encontrada em [110].

As árvores de decisão podem ser classificadas em dois tipos: *DT paralela* e *DT oblíqua*.

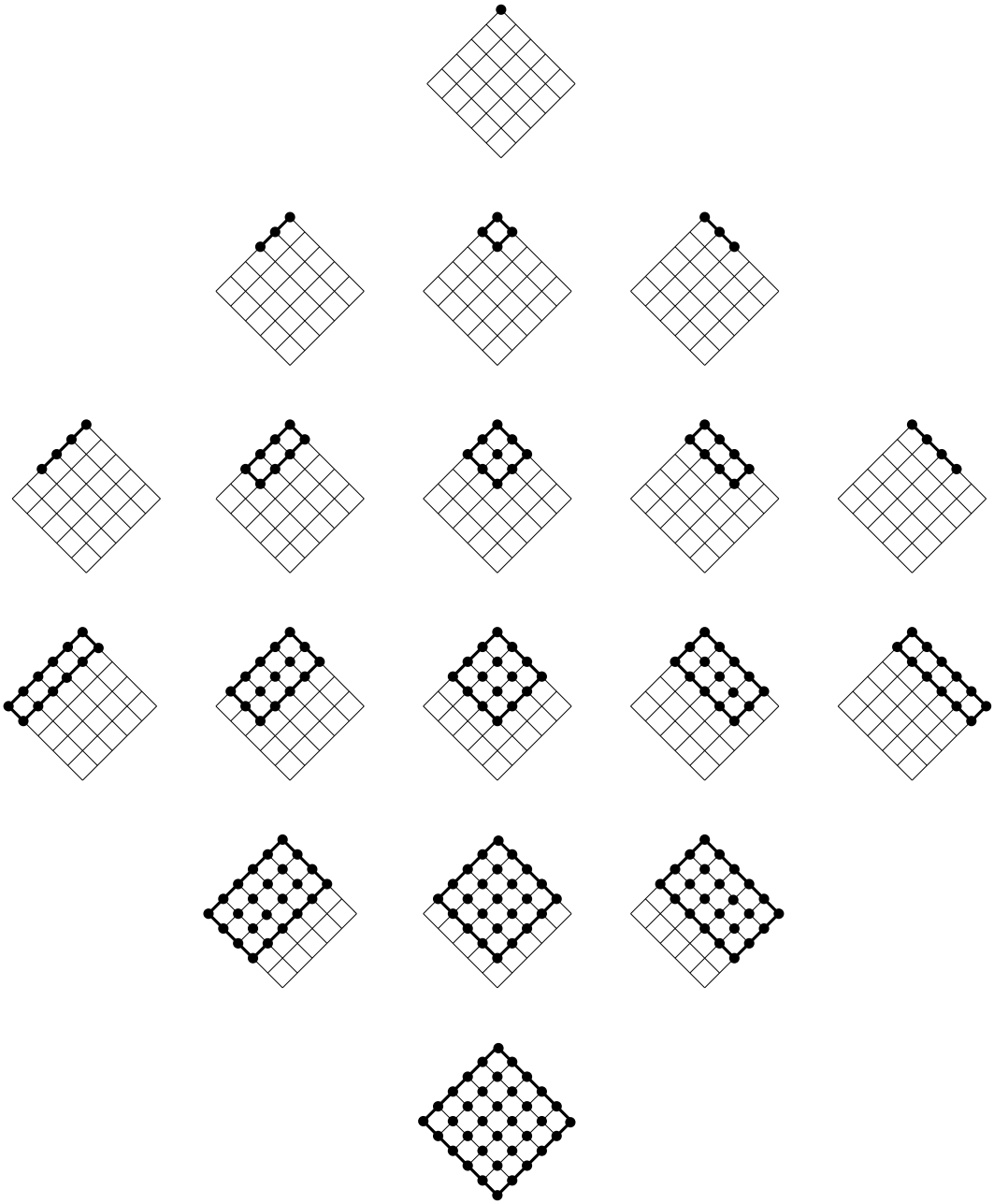


Figura 3.4: Representação do operador média pela base.

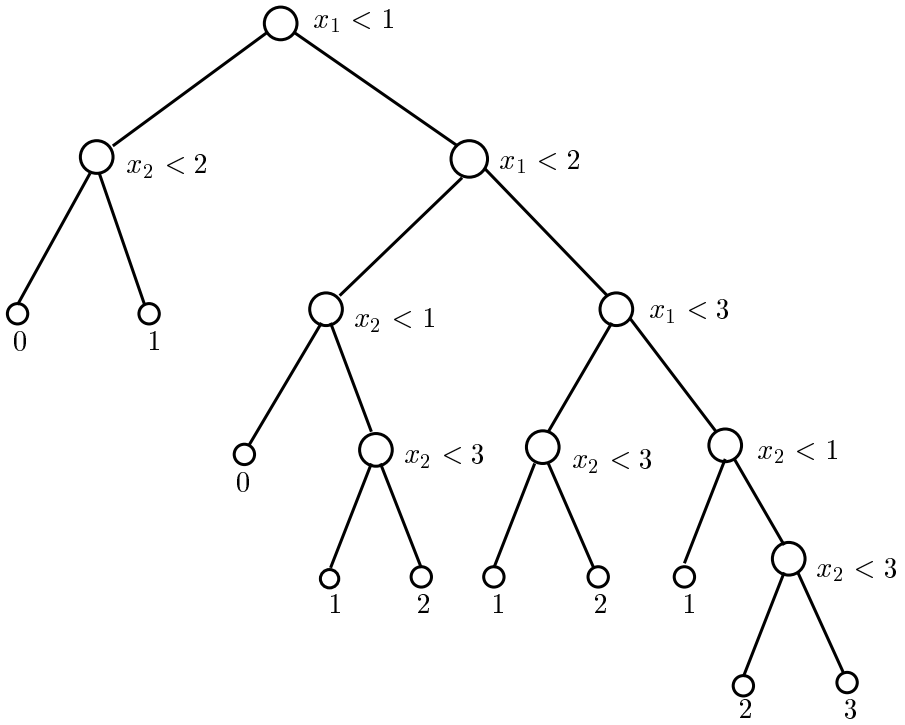


Figura 3.5: Uma possível representação do operador média.

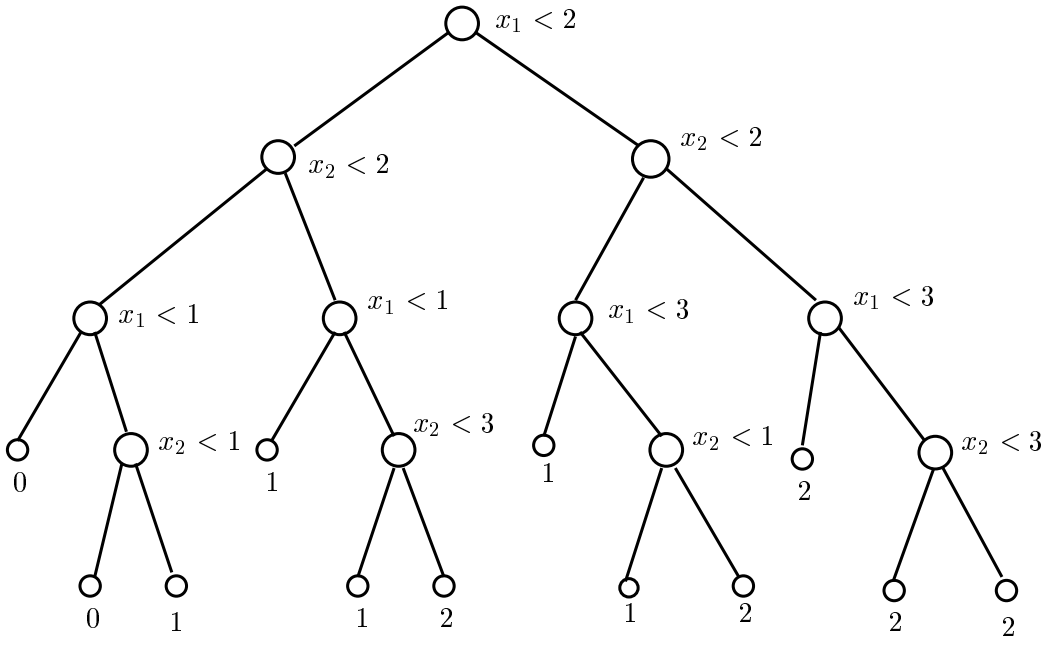


Figura 3.6: Uma outra possível representação do operador média.

Definição 3.15 *DT paralela - uma árvore de decisão é dita paralela se cada vértice interno está associado a um teste em uma única variável x_i de* $\mathbf{x}$.

Geometricamente, as fronteiras resultantes da partição do reticulado são *hiperplanos paralelos* aos eixos de um sistema cartesiano definido por $\mathbf{x}$, onde cada variável x_i de $\mathbf{x}$ é um eixo desse espaço. A figura 5.5 ilustra esse conceito.

Definição 3.16 *DT oblíqua - uma árvore de decisão é dita oblíqua (ODT) se a decisão associada a cada nó é baseada numa combinação linear das variáveis x_i de* $\mathbf{x}$.

Desta vez, se olharmos para as fronteiras resultantes da partição do reticulado, elas serão *hiperplanos oblíquos* aos eixos do sistema cartesiano definido por $\mathbf{x}$. A figura 5.7 ilustra esse conceito.

A vantagem de usar árvores oblíquas em relação às árvores paralelas é que a altura das árvores oblíquas tende a ser mais baixa que a altura das árvores paralelas, o que diminui o tempo médio de busca.

3.3.1 Usando a representação via árvores de decisão para aplicar ψ

Uma das vantagens de usar a estrutura de árvore é a rapidez de aplicação do operador, ou mais especificamente, a rapidez para achar o valor de uma dada configuração. Para isto, basta percorrer a árvore, começando pela raiz, de acordo com o resultado do teste associado ao nó corrente. O processo pára quando alcançamos uma folha, que contém o valor a ser associado à configuração.

3.4 Projeto de operadores de janela

Seja S uma família de pares de imagens, $S = \{h_i, g_i\}$, $i \in \mathbf{N}$ (o conjunto dos números naturais), definidas em E com valores em L, onde $h_i \in L^E$ é denominada *imagem observada* e $g_i \in L^E$, *imagem ideal*. Vamos supor que os pares $\{h_i, g_i\}$ são tais que existe um operador de imagens Ψ tal que $\Psi(h_i) = g_i$, para qualquer par de imagens de S. Apesar de supormos que Ψ existe, não o conhecemos e, por isto, queremos estimá-lo. Isto posto, vamos dizer que, dada uma família S de pares de imagens, projetar estatísticamente um operador de imagens sobre S é achar um operador $\hat{\Psi}$ que minimiza uma medida de erro entre $\hat{\Psi}(h_i)(t)$ e $g_i(t)$, para todo $t \in E$ e todos os pares (h_i, g_i) de S [8]. A medida de erro a ser minimizada é usualmente o erro absoluto médio ("*Mean Absolute Error*" ou MAE) ou o erro quadrático médio ("*Mean Square Error*" ou MSE).

O erro absoluto médio, ou MAE, é dado por:

$$\mathrm{MAE}(\Psi) = E[|g - \Psi(h)|], \tag{3.13}$$

e o erro quadrático médio, ou MSE, é dado por:

$$\mathrm{MSE}(\Psi) = E[(g - \Psi(h))^2]. \tag{3.14}$$

Note que não temos Ψ e, por isso, queremos estimá-lo de modo que o erro do operador seja mínimo. No contexto da teoria de decisão, essas medidas são chamadas de *funções de risco*, e as funções $|g - \Psi(h)|$ e $(g - \Psi(h))^2$ são chamadas de *funções de perda*. O problema principal em teoria de decisão é, aplicando o critério *minimax* [106, 94], encontrar o estimador $\hat{\Psi}$ que leve ao menor risco.

Teorema 3.17 *Seja S uma família de pares de imagens, $S = \{h_i, g_i\}$, $i \in \mathbf{N}$, definidas em E com valores em L. Dado um ponto $t \in E$ e um $W \subset E$, seja $P(g(t), h_{-t}|W)$ a probabilidade da configuração $h_{-t}|W$ estar associada ao nível de cinza $g(t)$. Por hipótese, vamos supor que $P(g(t), h_{-t}|W)$ não depende de t, isto é, o processo é estacionário. Se Ψ é um W-operador, então $MSE(\Psi) = MSE(\psi)$, onde ψ é a função característica do operador Ψ.*

Dem.:

$$\begin{aligned}
MSE(\Psi) &\overset{(1)}{=} E[(g(t) - \Psi(h)(t))^2] \\
&\overset{(2)}{=} \sum_{g,h \in S} \tfrac{1}{|E|} \sum_{t \in E} (g(t) - \Psi(h)(t))^2 P(g,h) \\
&\overset{(3)}{=} \tfrac{1}{|E|} \sum_{t \in E} \sum_{g(t) \in L} \sum_{h_{-t}|W \in L^W} (g(t) - \psi(h_{-t}|W))^2 P(g(t), h_{-t}|W) \\
&\overset{(4)}{=} \tfrac{1}{|E|} \sum_{t \in E} \sum_{y \in L} \sum_{\mathbf{x} \in L^W} (y - \psi(\mathbf{x}))^2 P(y, \mathbf{x}) \qquad (3.15) \\
&\overset{(5)}{=} \sum_{y \in L} \sum_{\mathbf{x} \in L^W} (y - \psi(\mathbf{x}))^2 P(y, \mathbf{x}) \\
&\overset{(6)}{=} E[(Y - \psi(\mathbf{X}))^2] \\
&\overset{(7)}{=} MSE(\psi)
\end{aligned}$$

■

A passagem da equação 1 para a 2 deve-se à definição de esperança. A passagem da equação 2 para a 3 deve-se ao fato de Ψ ser um W-operador, e portanto definido localmente por uma função característica ψ. A probabilidade conjunta $P(g(t), h_{-t}|W)$ aparece na equação pois agora os somatórios são sobre todos os níveis de cinza e sobre todas as possíveis configurações. A passagem da equação 3 para a 4 deve-se à hipótese de estacionariedade. A passagem da equação 4 para a 5 vale pois a somatória sobre t não depende de t, e portanto os fatores $|E|$ se cancelam. A passagem da equação 5 para a 6 deve-se novamente à definição de esperança. Finalmente, a passagem da equação 6 para a 7 deve-se à definição de MSE.

Portanto, como o MSE(Ψ) é igual ao MSE(ψ), então minimizar o MSE de um W-operador Ψ é equivalente a minimizar o MSE de sua função característica ψ [8].

Teorema 3.18 *O MSE mínimo de um operador Ψ i.t. e l.d. por ψ é dado por $\psi(\mathbf{x}) = E[y|\mathbf{x}]$, onde $E[\bullet]$ é a esperança de $\bullet$.*

Dem.: *Reproduzimos aqui a prova dada por Dougherty em [8]. Esta prova tem duas partes. Ele mostra primeiramente quem é o estimador que minimiza o MSE de uma variável aleatória Y e, depois, quem é o estimador que minimiza o MSE de uma variável aleatória condicionada $Y|\mathbf{X}$.*

a) $E[(Y - \hat{Y})^2] = E[Y^2 - 2Y\hat{Y} + \hat{Y}^2] = E[Y^2] - 2\hat{Y}E[Y] + \hat{Y}^2$

Derivando a equação acima em função de $\hat{Y}$ e igualando a zero, temos $\hat{Y} = E[Y]$. Derivando em relação a $\hat{Y}$ uma vez mais, temos que a segunda derivada é positiva, ou seja, $\hat{Y} = E[Y]$ é um valor que minimiza $E[|Y - \hat{Y}|^2]$. Na verdade, usamos o valor inteiro mais próximo de $E[Y]$ pois Y está definido no domínio discreto.

b)

$$\begin{aligned}
&E[(Y - \psi(\mathbf{X}))^2] \overset{(1)}{=} \\
&\quad \overset{(1)}{=} E[(Y - \psi(\mathbf{X})^2|\mathbf{X}]
\end{aligned}$$

$$
\begin{aligned}
&\overset{(2)}{=} E[(Y - E[Y|\mathbf{X}] + E[Y|\mathbf{X}] - \psi(\mathbf{X}))^2|\mathbf{X}] \\
&\overset{(3)}{=} E[((Y - E[Y|\mathbf{X}])^2 - 2(Y - E[Y|\mathbf{X}])(E[Y|\mathbf{X}] - \psi(\mathbf{X})) + (E[Y|\mathbf{X}] - \psi(\mathbf{X}))^2)|\mathbf{X}] \\
&\overset{(4)}{=} E[(Y - E[Y|\mathbf{X}])^2|\mathbf{X}] - 2E[(Y - E[Y|\mathbf{X}])(E[Y|\mathbf{X}] - \psi(\mathbf{X}))|\mathbf{X}] + E[(E[Y|\mathbf{X}] - \psi(\mathbf{X}))^2|\mathbf{X}] \\
&\overset{(5)}{=} E[(Y - E[Y|\mathbf{X}])^2|\mathbf{X}] + E[(E[Y|\mathbf{X}] - \psi(\mathbf{X}))^2|\mathbf{X}] \\
&\overset{(6)}{\geq} E[(Y - E[Y|\mathbf{X}])^2|\mathbf{X}] \qquad (3.16)
\end{aligned}
$$

A passagem da igualdade 1 para a 2 vem da soma e subtração de $E[Y|\mathbf{X}]$. A passagem da igualdade 2 para a 3 e, em seguida, 3 para 4, vem da expansão do soma de quadrados. A passagem da igualdade 4 para a 5 vem do fato que o termo $E[(Y - E[Y|\mathbf{X}])(E[Y|\mathbf{X}] - \psi(\mathbf{X}))|\mathbf{X}]$ é nulo pois $E[(Y - E[Y|\mathbf{X}])(E[Y|\mathbf{X}] - \psi(\mathbf{X}))|\mathbf{X}] = E[Y - E[Y|\mathbf{X}]|\mathbf{X}]E[E[Y|\mathbf{X}] - \psi(\mathbf{X})|\mathbf{X}] = (E[Y|\mathbf{X}] - E[E[Y|\mathbf{X}]|\mathbf{X}])E[E[Y|\mathbf{X}] - \psi(\mathbf{X})|\mathbf{X}] = (E[Y|\mathbf{X}] - E[Y|\mathbf{X}])E[E[Y|\mathbf{X}] - \psi(\mathbf{X})|\mathbf{X}] = 0$. A passagem da igualdade 5 para a desigualdade 6 é do fato que ambos termos são positivos, assim sua soma é maior que a de um deles. Portanto, o estimador de $Y|\mathbf{X}$ que minimiza $E[(Y - \psi(\mathbf{X})^2]$ é a esperança de Y condicionada a $\mathbf{X}$, ou $E[Y|\mathbf{X}]$, como queríamos mostrar. ■

Como não conhecemos a distribuição conjunta de $Y|\mathbf{x}$, então temos que usar como estimador de $E[y|\mathbf{x}]$ a média amostral e atribuímos a $\psi_{W,N}$($\mathbf{x}$ o valor inteiro mais próximo da média amostral, isto é,

$$\psi_{W,N}(\mathbf{x}) = \lfloor \frac{1}{N(\mathbf{x})} \sum_{i=1}^{N(\mathbf{x})} y_i|\mathbf{x} + 0.5 \rfloor \qquad (3.17)$$

onde $y|\mathbf{x}$ é o valor de Y dada a observação $\mathbf{X} = \mathbf{x}$, $N(\mathbf{x})$ é o número de vezes que $\mathbf{x}$ é observado na amostra e $\lfloor \bullet \rfloor$ é o menor inteiro mais próximo de $\bullet$.

De forma semelhante, pode-se mostrar que o estimador que minimiza o MAE é a mediana [93, 111].

3.5 Análise do erro de projeto

Nesta seção vamos estudar o incremento de erro causado por usarmos um operador de janela projetado em lugar do operador de janela ótimo.

Se $\psi_W : L^W \to L$ é a função característica do operador MSE ótimo sobre uma janela W (significando que $\psi_W(\mathbf{X}) = E[Y|\mathbf{X}], \forall \mathbf{X} \in L^W$) e $\psi : L^W \to L$ é uma outra função característica qualquer sobre W, vamos chamar de *incremento de erro*, o erro resultante por usarmos ψ ao invés de ψ_W e o denotaremos por $\triangle(\psi, \psi_W)$. Seja $\psi_{W,N}$ uma função característica estimada de ψ_W baseada em N pares de amostras $(\mathbf{X}^1, Y^1), \ldots, (\mathbf{X}^N, Y^N)$, então o incremento de erro será igual a $\triangle(\psi_{W,N}, \psi_W)$. A figura 3.7 ilustra o incremento de erro $\triangle(\psi_{W,N}, \psi_W)$ para o operador $\psi_{W,N}$ em relação ao operador ótimo sobre W.

O MSE do operador projetado é dado por

$$\text{MSE}\langle\psi_{W,N}\rangle = \text{MSE}\,\langle\psi_W\rangle + \triangle(\psi_{W,N}, \psi_W). \qquad (3.18)$$

Como $\psi_{W,N}$ depende das amostras de treinamento, ele é aleatório e sua *imprecisão*, ou *erro de estimação*, é definido como a esperança do custo, isto é, $E[\triangle(\psi_{W,N}, \psi_W)]$[55]. O MSE esperado do filtro projetado é obtido tomando a esperança da Eq. 3.18, lembrando que $\text{MSE}\,\langle\psi_W\rangle$ é constante.

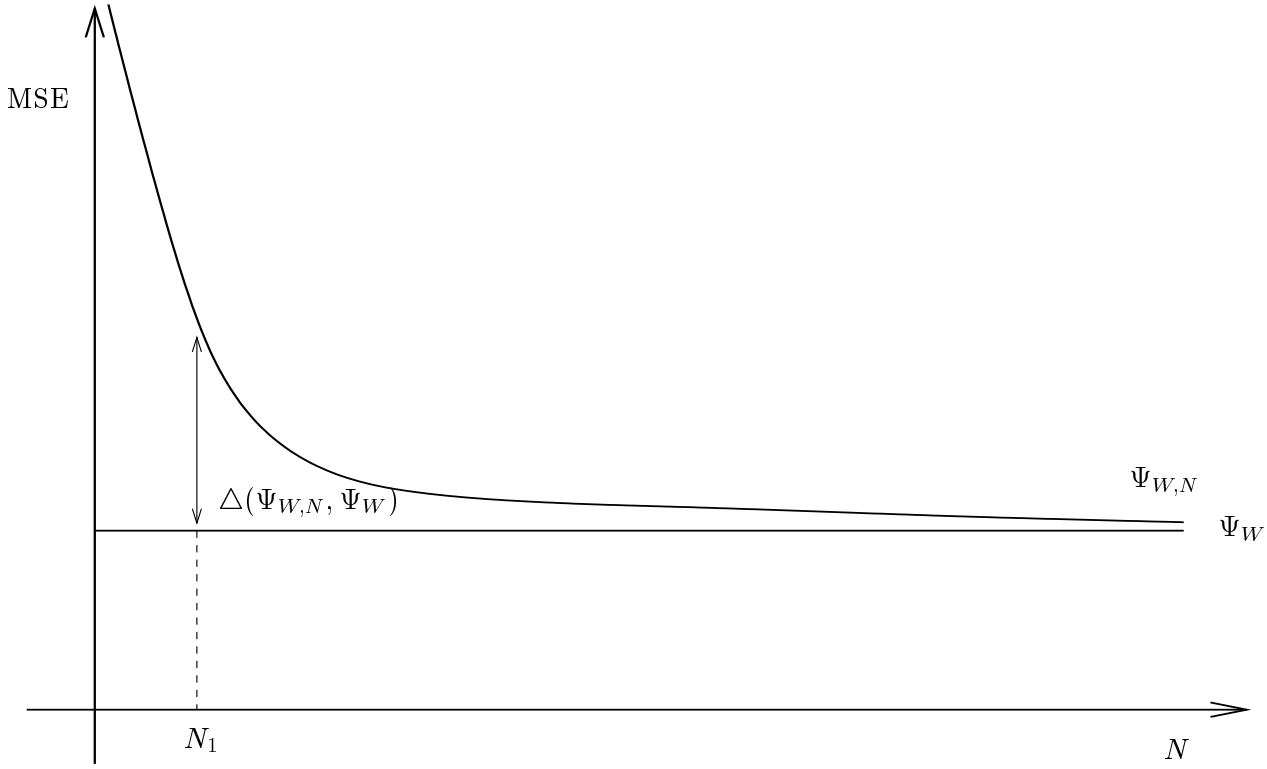


Figura 3.7: Incremento de erro

Não só $E[\triangle(\psi_{W,N}, \psi_W)]$ depende da estimação, como também, tende a zero quando N tende a infinito ($\lim_{N\to\infty} E[\triangle(\psi_{W,N}, \psi_W)] = 0$), que é geralmente o caso. Porém, para janelas grandes, por exemplo V, onde $W \subset V$, $|W| << |V|$, o número de exemplos de treinamento N precisa ser tão grande para fazer $E[\triangle(\psi_{V,N}, \psi_V)]$ aceitavelmente pequeno, que é freqüentemente impossível ter alguma estimativa confiável da esperança condicional.

A figura 3.8 ilustra o conceito acima. O MSE do operador ótimo sobre a janela W, Ψ_W, é ilustrado por uma reta paralela ao eixo das abscissas, o MSE do operador ótimo sobre a janela V, $W \subset V$, também é ilustrado por uma reta paralela ao eixo das abscissas, mais baixa que a anterior pois $MSE(\psi_W) \geq MSE(\psi_V)$, já que $W \subset V$. A função característica de $\psi_{W,N}$ estimada é melhor para N pequenos e aproxima-se de ψ_W quando N tende a infinito. A função característica de $\psi_{V,N}$, a medida que N tende a infinito, fica melhor que $\psi_{W,N}$ e, depois, fica melhor que ψ_W, aproximando-se de ψ_V.

Mais que tentar escolher o melhor filtro entre todas as possíveis funções características de n variáveis, $X_1, X_2, \ldots, X_n$, podemos restringir nossa escolha às melhores funções em uma subclasse C de funções. O filtro ótimo em C, denotado por $\psi_{C,W}$, claramente não dá resultado melhor do que ψ_W pois este último já é o ótimo sobre W. Vamos denotar por $\triangle(\psi_{C,W}, \psi_W)$ o custo da restrição. Sendo restrito, $\psi_{C,W}$ tem menos parâmetros para serem estimados e, assim, seu erro de estimação, $E[\triangle(\psi_{C,W,N}, \psi_{C,W})]$, pode ser significativamente menor do que o erro de estimação de ψ_W, $\triangle(\psi_{W,N}, \psi_W)$. Desta forma, a restrição à subclasse C é benéfica se o erro de estimação de $\psi_{C,W}$ mais o custo de estimação for menor que o erro de estimação de ψ_W, que traduz-se nas seguinte

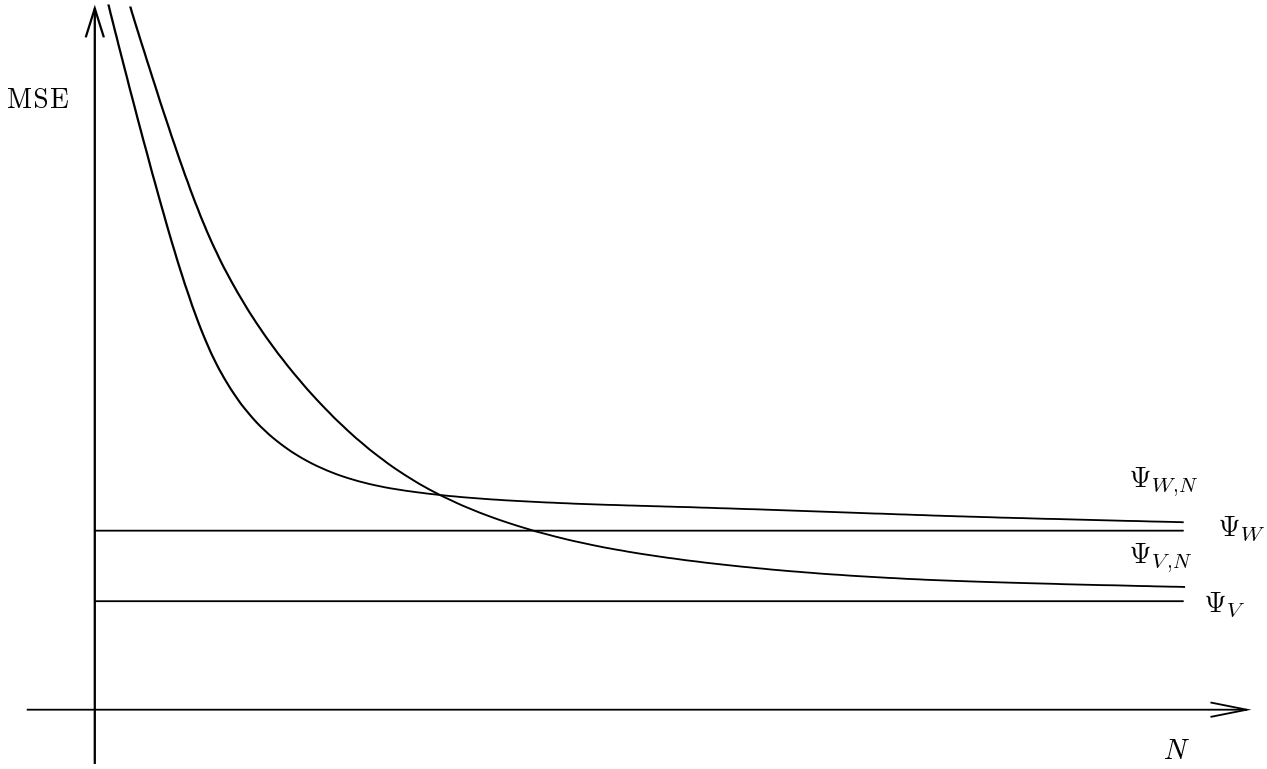


Figura 3.8: Minimização do MSE

desigualdade:

$$E[\triangle(\psi_{C,W,N}, \psi_{C,W})] + \triangle(\psi_{C,W}, \psi_W) < E[\triangle(\psi_{W,N}, \psi_W)]. \tag{3.19}$$

Ambos os lados da Eq. 3.19 envolvem N, por isso a utilidade da restrição depende do tamanho da amostra. Assumindo que $E[\triangle(\psi_{C,W,N}, \psi_W)] \to 0$ e $E[\triangle(\psi_{W,N}, \psi_W)] \to 0$ quando $N \to \infty$, então a desigualdade da Eq. 3.19 é tipicamente satisfeita para N pequeno e não é satisfeita para N grande. Essas afirmações ficam claras analisando-se a figura 3.8.

Em geral, o filtro ótimo sobre um conjunto de variáveis aleatórias tem MSE menor que o filtro ótimo sobre um subconjunto daquelas variáveis. Se a janela V contém a janela W, então ψ_W é *restrito* relativamente a V e o incremento de erro por otimizar sobre W ao invés de sobre V é $\triangle(\psi_W, \psi_V)$. Porém, por causa do excesso de configurações a serem estimadas quando usamos V, a precisão será pior para ψ_V do que para ψ_W, ou em outras palavras, $E[\triangle(\psi_{V,N}, \psi_V)] \geq E[\triangle(\psi_{W,N}, \psi_W)]$.

O filtro ótimo sobre V é melhor porquê ele envolve mais observações, porém a otimização sobre V é preferível a otimização sobre W, se e somente se

$$E[\triangle(\psi_{V,N}, \psi_V)] < E[\triangle(\psi_{W,N}, \psi_W)] + \triangle(\psi_W, \psi_V). \tag{3.20}$$

Quando V é significativamente maior que W, esta desigualdade é satisfeita somente para N proibitivamente grandes.

Tendo um número suficiente de observações para utilizar, podemos optar por usar um operador restrito ótimo $\psi_{C,V}$ sobre V. O operador restrito projetado sobre V, $\psi_{C,V,N}$, é superior ao operador não restrito projetado sobre W, $\psi_{W,N}$, se

$$E[\triangle(\psi_{C,V,N}, \psi_{C,V})] + \triangle(\psi_{C,V}, \psi_V) < E[\triangle(\psi_{W,N}, \psi_W)] + \triangle(\psi_W, \psi_V). \tag{3.21}$$

Se o custo da restrição é muito alto, então o aumento da precisão pode não ser suficiente para fazer o método de restringir o espaço benéfico. A figura 3.8 ilustra esses efeitos da restrição e do número de exemplos de treinamento.

Se mudamos nossa perspectiva e olhamos V como uma janela de referência, então otimizar sobre $W \subset V$ é chamado de *restrição otimizada*. Revertendo a Eq. 3.20, a restrição é benéfica se e somente se,

$$E[\triangle(\psi_{W,N}, \psi_W)] + \triangle(\psi_W, \psi_V) < E[\triangle(\psi_{V,N}, \psi_V)] \tag{3.22}$$

Esta desigualdade tem a mesma forma da Eq. 3.19. Lá a otimização não restrita se dava sobre W e havia uma classe de restrição C restringindo a otimização; aqui a otimização não restrita se dá sobre V e a restrição se manifesta pela otimização restrita à subjanela W. Com relação ao MSE, temos que: $\text{MSE}\langle\psi_V\rangle < \text{MSE}\langle\psi_W\rangle$ pois $W \subset V$. Temos também que: $E[\text{MSE}\langle\psi_{V,N}\rangle] \to \text{MSE}\langle\psi_V\rangle$ quando $N \to \infty$, e que $E[\text{MSE}\langle\psi_{W,N}\rangle] \to \text{MSE}\langle\psi_W\rangle$ quando $N \to \infty$, como na figura 3.8. Porém, para N pequeno,

$$\text{MSE}\langle\psi_V\rangle < \text{MSE}\langle\psi_W\rangle < E[\text{MSE}\langle\psi_{W,N}\rangle] < E[\text{MSE}\langle\psi_{V,N}\rangle] \tag{3.23}$$

e a restrição sobre W é benéfica neste caso.

Normalmente C é caracterizado por alguma restrição algébrica, tal como restringir o operador à classe dos operadores lineares, ou dos operadores crescentes, ou dos operadores idempotentes, ou dos operadores "stack", ou dos W-operadores, etc.

3.6 Indução

Discutimos até agora o projeto estatístico de W-operadores a partir de pares de imagens observadas e ideais, (h_i,g_i), respectivamente. Vimos como achar o valor de $y|\mathbf{x}$ que minimiza um certo critério de erro para a configuração dada $\mathbf{x}$, mas nada falamos sobre qual valor seria atribuido pelo sistema na aplicação do operador em uma configuração não observada. A isso dá-se o nome de *indução*.

Uma das definições da palavra "indução" (ou "generalização" no contexto de aprendizado computacional [7]) encontrada no dicionário Aurélio é: "Raciocínio em que de fatos particulares se tira uma conclusão genérica". De fato, como ψ é uma função e, portanto, tem que estar definida sobre todas as configurações possíveis, esta palavra aplica-se muito bem ao que queremos que aconteça ao valor atribuido à saída y para uma dada configuração $\mathbf{x}$ não observada. Se o valor que deveria ser atribuído a $\mathbf{x}$ for induzido corretamente a partir das distribuições estimadas para as outras configurações observadas, então dizemos que ele tem um "bom poder de indução".

Existem diversos métodos de aprendizado computacional que produzem um operador a partir de um conjunto de exemplos de treinamento. Normalmente é difícil compará-los com o objetivo de escolher "o melhor" método de todos pois existem muitas variáveis que influenciam na comparação. Por exemplo, a quantidade e a distribuição conjunta dos exemplos de treinamento, o tempo de projeto do operador, a robustez do operador projetado, a quantidade de memória usada para representar o operador projetado, a velocidade de aplicação do operador e etc. Porém, há um certo consenso que a escolha do método depende bastante do problema que se quer resolver, isto é, existe um consenso que não existe **o método** de aprendizado que sirva para todos os problemas [7].

Para escolher um método simples de implementar, com boa capacidade de indução e sobre o qual tivéssemos certo controle sobre a representação, uma pesquisa bibliográfica na área de aprendizado

computacional e seus métodos foi realizada.

Dentre os vários métodos conhecidos, as redes neurais [112] ainda são as mais usadas. Elas são simples de implementar, existem referências que estudam como melhorar a eficiência delas [38], de como implementar operadores morfológicos usando redes neurais [113], de comparação de implementações de redes neurais [114], de discussão do poder de indução das redes neurais [115] e etc. Porém, decidimos não adotá-las por duas razões: (1) o projeto de um operador via redes neurais não se limita ao fornecimento de exemplos positivos e negativos para a rede, ele envolve também o projeto da topologia da rede, e isso é uma desvantagem sob nosso ponto de vista; (2) a representação via redes neurais não permite uma manipulação simples das propriedades algébricas do operador.

Uma outra representação bastante conhecida e usada é a representação via árvores de decisão. Elas são rápidas de construir, estendem o operador aprendido consistentemente e tem um bom poder de indução [116, 7, 117]. Essa foi a forma escolhida para a representação dos operadores "aperture" na primeira implementação do sistema.

A representação mais adequada em termos de facilidade de manipulação algébrica do operador é via a base do operador [18, 21]. No capítulo 5, apresentamos a extensão do algoritmo ISI [73, 118] para níveis de cinza e alguns exemplos de comparação com as árvores de decisão.

Outra alternativa considerada foi o uso das "Supporting Vector Machines" (SVM) [119]. Porém, pouco resultados eram conhecidos na época (1997), principalmente com relação aos algoritmos de indução e suas complexidades [120]

3.7 O sistema de aprendizado PAC

Esta seção apresenta as partes do sistema implementado para o projeto automático de operadores de imagens.

Há quatro módulos principais no sistema, quais sejam: (*i*) *observação* ou *estimação*, (*ii*) *decisão*, (*iii*) simplificação da representação e (*iv*) *aplicação*. A figura 3.9 ilustra o fluxograma do sistema.

- O módulo de estimação (*i*) consiste de uma biblioteca de funções para a especificação de uma janela W, funções para a especificação de arquivos de pares de imagens de treinamento, funções de percorrimento da janela nas imagens de treinamento e funções para o armazenamento eficiente das configurações observadas. O programa principal deste módulo vai transladar a janela sobre a imagem e ir guardando as configurações e rótulos associados a elas. A saída desse módulo é uma tabela das configurações observadas, juntamente com uma lista dos rótulos associados para cada configuração e o número de vezes que o rótulo foi observado associado àquela configuração.

- O módulo de decisão (*ii*) consiste de uma biblioteca de funções para a escolha do rótulo final que será associado a cada configuração. Esta escolha é feita segundo o critério de minimização de erro escolhido pelo usuário. A entrada do módulo é uma tabela de exemplos e rótulos (a saída do primeiro módulo). A saída é uma tabela em que cada linha contém uma configuração, um rótulo e a quantidade de vezes que a configuração foi observada.

- O módulo de simplificação (*iii*) consiste de uma biblioteca de funções para construir a representação do operador. Na parte das funções da biblioteca de níveis de cinza, há dois métodos

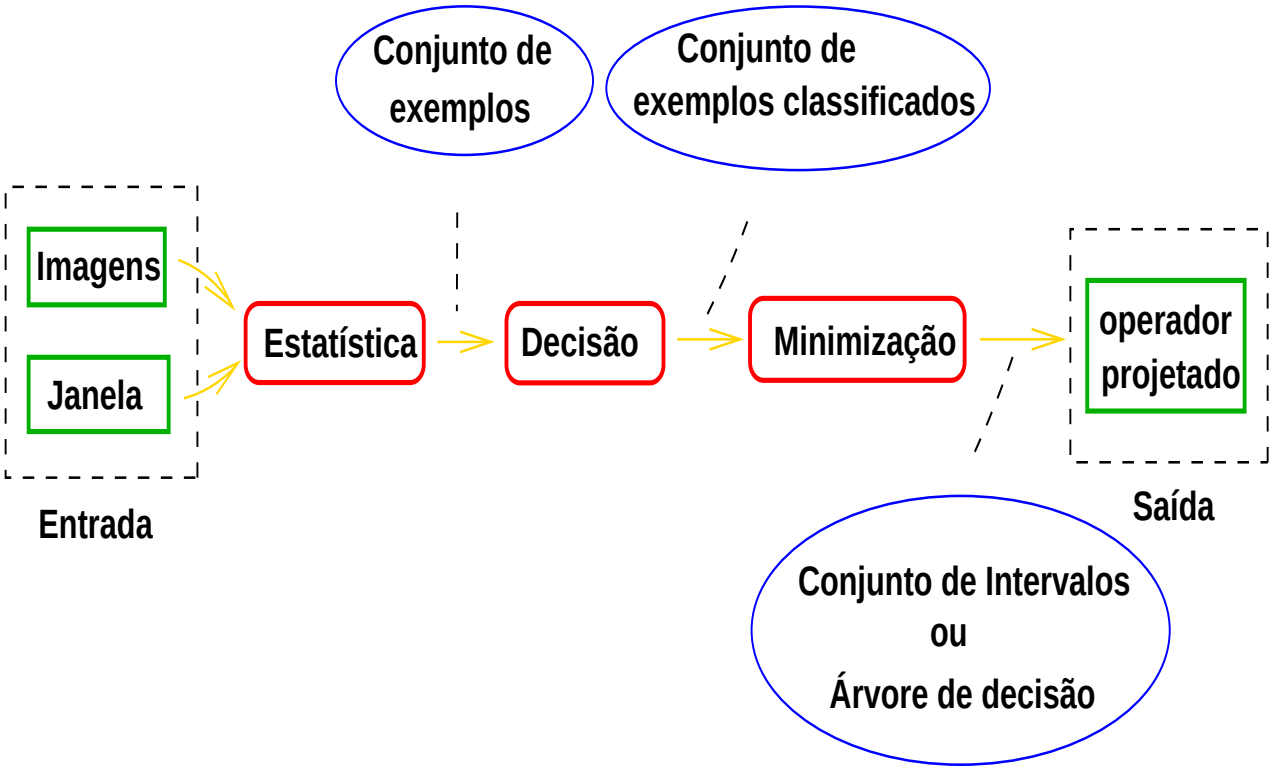


Figura 3.9: Fluxograma do sistema PAC

implementados: árvores de decisão e ISI. Estes métodos serão detalhados nos capítulos que seguem. A entrada do módulo é uma tabela de exemplos (saída do módulo *iii*), e a saída é o operador de imagens representado via uma árvore de decisão, ou via intervalos maximais do núcleo.

- O procedimento de aplicação (*iv*) é muito semelhante ao primeiro, mas agora não há necessidade de ir guardando configurações, apenas dar os rótulos corretos de acordo com a representação. A entrada é um operador de imagens (representado via árvores ou intervalos) e a saída é uma imagem.

Capítulo 4

Operadores "Aperture" e seu projeto

Neste capítulo apresentamos uma subclasse importante dos operadores de imagens em níveis de cinza denominada operadores *"aperture"*. A motivação para o estudo desses operadores veio da dificuldade do projeto estatístico de W-operadores em níveis de cinza, que deve-se aos problemas de estimação estatística inerentes ao tamanho dessa classe de operadores. A classe dos W-operadores é formada por operadores invariantes por translação e localmente definidos no espaço por uma janela W, $W \subset E$. O tamanho desta classe, i.e., a quantidade de operadores nessa classe, é exponencial no tamanho da janela W, isto é, $l^{(l^{|W|})}$, onde l é a quantidade possível de níveis de cinza. Embora esta classe de operadores seja menor que a classe de todos os operadores de imagens possíveis, ela é muito grande pois l é tipicamente igual a 256 níveis.

A solução encontrada para tratar o projeto de operadores em níveis de cinza foi restringir uma classe de operadores que, além de serem invariantes por translação e localmente definidos por uma janela espacial W, e são localmente definidos nos níveis de cinza por uma janela K, $K \subset Z$. Assim, quando K não é muito grande, uma boa estimação estatística é possível. Esses operadores foram inicialmente chamados de WK-operadores [62] e posteriormente renomeados para operadores "aperture".

Nas seções que seguem apresentaremos a formalização algébrica dos operadores "aperture".

4.1 Configurações "Aperture"

Nesta seção vamos definir as configurações "aperture" e mostrar como elas são obtidas. Uma configuração "aperture" é uma configuração de janela em que cada ponto está restrito a um pequeno intervalo dos inteiros. Formalmente, seja K um subconjunto dos inteiros, por exemplo, $K = [-k, k], k \in \mathbf{N}$. Uma *configuração "aperture"* é uma função de W em K e o conjunto de todas as possíveis configurações sobre W é denotado por K^W.

Seja u uma configuração de janela tal que $u \in L^W$ e seja z um valor inteiro, $z \in Z$. A *translação vertical* de u por $z \in Z$ é dada por, para qualquer $x \in W$, $u_z(x) = (u + z)(x) = u(x) + z$.

Há duas maneiras equivalentes de obter configurações "aperture" a partir de uma configuração de janela: transladando verticalmente a configuração para dentro da janela $W \times K$ (centrada na origem), ou transladando verticalmente a janela $W \times K$. Em ambos os casos, os pontos fora de

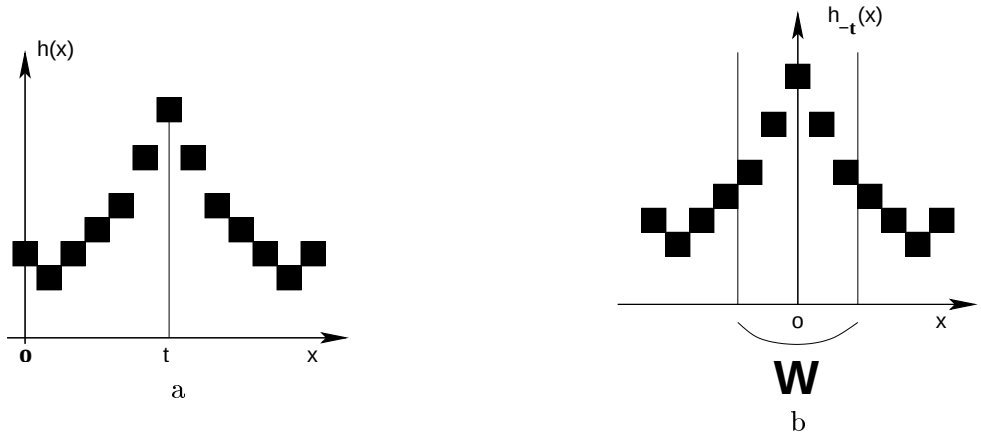


Figura 4.1: (a) $h(x)$, (b) $h_{-t}|W$

$W \times K$ têm que ser projetados para dentro de $W \times K$, isto é, os valores maiores ou menores que $|k|$ são saturados em k, ou $-k$.

Seja $u \in L^W$ uma configuração de janela. A configuração "aperture" é uma função $u_{-z}|K$, $u_{-z}|K \in K^W$, dada por, para qualquer $x \in W$,

$$(u_{-z}|K)(x) = \bigwedge \left\{ \bigvee \{-k, u(x) - z\}, k \right\}. \tag{4.1}$$

É importante dizer que o valor da translação vertical z é normalmente uma função de u, e a forma de escolher essa função será discutida mais adiante.

As figuras 4.1 e 4.2 ilustram o janelamento "aperture". A figura 4.1a mostra uma função discreta h, a figura 4.1b mostra a translação de h por $-t$ e a configuração de janela $u = h_{-t}|W$. a figura 4.2c mostra a translação vertical de u por $-z$, e a figura 4.2d mostra a configuração de janela $u_{-z}|K, K = \{-1, 0, 1\}$.

A figura 4.2a mostra a translação vertical de u por $-z$, e a figura 4.2b mostra a configuração de janela $u_{-z}|K, K = \{-1, 0, 1\}$.

4.2 Invariância por translação vertical, ou nos níveis de cinza

Definição 4.1 *Seja Ψ um operador de imagens. Ψ é dito invariante por translação vertical, ou nos níveis de cinza sse, para qualquer $z \in L$ e qualquer $f \in L^E$, $\Psi(f + z) = \Psi(f) + z$.*

A figura 4.3 ilustra este conceito.

Os operadores invariantes por translação espacial e vertical são também denominados T-operadores [14]. A *dilatação* e a *erosão* definidas via soma e diferença de Minkowski [14] são T-operadores. Porém, nem todas as dilatações e erosões são T-operadores.

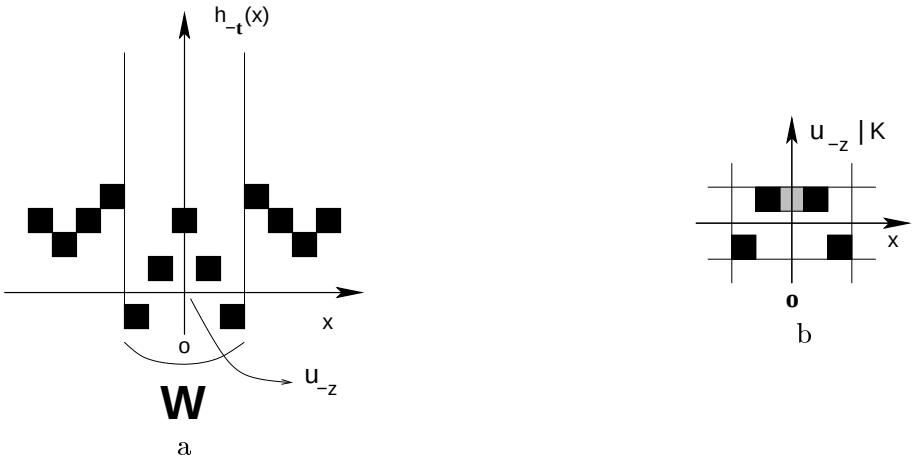


Figura 4.2: (a) u_{-z} (b) $u_{-z}|K$

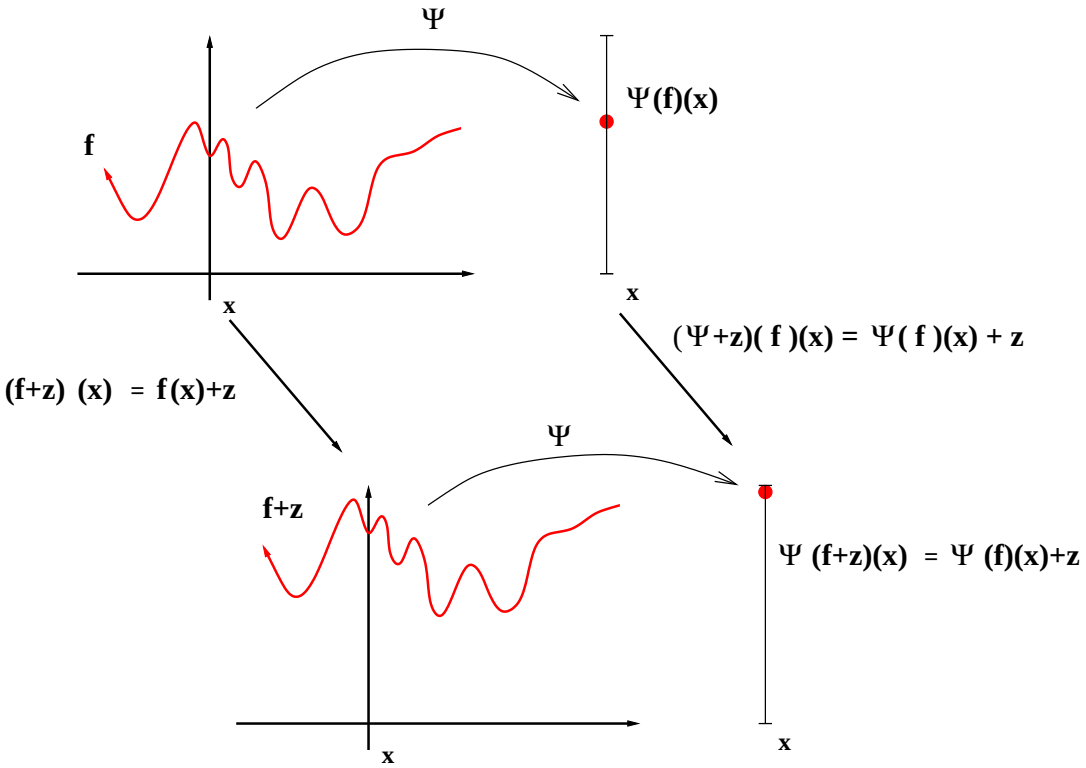


Figura 4.3: Invariância por translação nos níveis de cinza.

4.3 Definição local nos níveis de cinza

Como vimos, se $\Psi : L^E \to L^E$ é um W-operador, então ele pode ser caracterizado por uma função característica $\psi : L^W \to L$. Esta função, por sua vez, também pode ser localmente definida nos níveis de cinza.

Definição 4.2 *Seja Ψ um W-operador de imagens, $\Psi : L^E \to L^E$, e seja ψ a função característica de Ψ, $\psi : L^W \to L$. A função ψ é dita localmente definida em K sse, para qualquer $u \in L^W$ e qualquer $z \in L$,*

$$\psi(u) = z + \phi_z(u_{-z}|K), \tag{4.2}$$

onde ϕ_z é uma função de K^W em M, e M é um intervalo de Z, $M = [-m, m], m \in \mathbf{N}$. Note que a função ϕ_z é dependente de z.

4.4 Operadores ζ_I-"aperture"

Os operadores ζ_I-"aperture" são W-operadores invariantes por translação e localmente definidos nos níveis de cinza. Nesta seção, vamos escrevê-los em termos de funções características que são invariantes por translação e localmente definidas nos níveis de cinza.

Definição 4.3 *Uma função $\psi : L^W \to L$ é chamada de função K-característica se ela é localmente definida em K e ao mesmo tempo invariante por translação nos níveis de cinza.*

Proposição 4.4 *Sejam $u \in L^W$, $\zeta_I(u)$ uma função de L^W em L invariante por translações nos níveis de cinza, e $\psi : L^W \to L$ uma função característica definida por,*

$$\psi(u) = \zeta_I(u) + \phi(u_{-\zeta_I(u)}|K), \tag{4.3}$$

onde ϕ é uma função de K^W em M. Então ψ é uma função K-característica, isto é, ela é localmente definida e invariante por translação vertical.

A prova desta proposição é uma conseqüência imediata das definições 4.2 e 4.3.

Definição 4.5 *Um operador Ψ caracterizado por uma função K-característica ψ é chamada de operador ζ_I-"aperture".*

Definição 4.6 *Seja $u^t = f_{-t}|W$ uma configuração de L^W num ponto $t \in E$, seja $\zeta_I(u^t)$ uma função de L^W em L invariante por translações nos níveis de cinza, e seja ϕ uma função de K^W em M. Um operador ζ_I-"aperture", Ψ, é um mapeamento de L^E em L^E, dado por,*

$$\Psi(f)(t) = \zeta_I(u^t) + \phi(u^t_{-\zeta_I(u^t)}|K). \tag{4.4}$$

O número de funções K-características dentro da classe dos operadores ζ_I-"aperture" é:

$$|M|^{(|K|^{|W|})}. \tag{4.5}$$

Se $|K|$ e $|M|$ são pequenos, então a quantidade de operadores de K^W em M é bem menor do que a quantidade de operadores de L^W em L.

4.5 Caracterização ζ_I-"apertures"

Nesta seção mostraremos como os ζ_I-"apertures" podem ser caracterizados usando as funções K-características.

Teorema 4.7 *Seja ϕ é uma função característica de K^W em M, e seja $\mathcal{K}(\phi)$ o kernel de ϕ. Seja $\lambda_{a,b}$ a função sup-geradora de intervalo $[a,b]$. Então, ϕ pode ser representada por:*

$$\phi(v) = \bigvee\{y \in M : \bigvee\{\lambda_{a,b}(v) : [a,b] \subseteq \mathcal{K}(\phi)(y)\} = 1\} \tag{4.6}$$

para qualquer configuração $v \in K^W$.

Portanto, como a função K-característica $\psi : L^W \to L$ pode ser escrita na forma $\psi(u) = \zeta(u) + \phi(u_{-\zeta(u)}|K)$, dada a sup-representação da função característica ϕ, podemos representar os operadores em termos de sup-geradoras.

Teorema 4.8 *Seja Ψ um operador ζ_I-"aperture" e ψ sua função K-característica. Se ψ é caracterizado por uma função característica ϕ, então Ψ pode ser escrito por:*

$$\Psi(f)(t) = \zeta(u^t) + \bigvee\{y \in M : \bigvee\{\lambda_{a,b}(u^t_{-\zeta(u^t)}|K) : [a,b] \in \mathbf{B}(\phi)(y)\} = 1\}, \tag{4.7}$$

onde $u^t = f_{-t}|W$ é uma configuração de L^W num ponto $t \in E$, e $\zeta_I(u^t)$ é uma função de L^W em L invariante por translação nos níveis de cinza.

A prova dos teoremas 4.7 e 4.8 encontram-se em [62].

4.5.1 Operadores (ζ_I,ζ_O)-"Aperture"

Os operadores ζ_I-"aperture" formam uma subclasse dos W-operadores que não descrevem alguns operadores úteis para segmentação de imagens. Por exemplo, o operador "threshold" não é um ζ_I-"aperture" pois ele não é invariante por translação vertical, isto é, $T_h(f+z) \neq T_h(f) + z, \forall f \in L^E, \forall z \in L$.

Por outro lado, a descrição formal dos W-operadores pode ser feita pela fórmula 4.4, desde que $\zeta_I(u) = 0, \forall u \in L^W$. Desta forma, tanto os W-operadores, quanto os ζ_I-operadores, podem ser descritos pela mesma fórmula. Fazendo uma pequena mudança em 4.4, podemos representar todos os W-operadores, inclusive os *classificadores* (i.e., os operadores que associam uma configuração $\mathbf{x}$ a um rótulo). A vantagem de escrever várias classes de operadores com apenas uma fórmula é que, além de facilitar a descrição dos operadores, também facilita a implementação do sistema de aprendizado.

Sejam ζ_I e ζ_O, $\zeta_I, \zeta_O : L^W \to L$, duas funções de L^W em L. ζ_I está associada às translações na imagem observada, isto é, é a função que posiciona a "aperture". ζ_O está associada às translações na imagem ideal, isto é, é a função que "posiciona" o resultado da função característica.

Definição 4.9 *Seja $u^t = f_{-t}|W$ uma configuração de L^W num ponto $t \in E$ e seja ϕ uma função de K^W em M. Um operador (ζ_I,ζ_O)-"aperture" Ψ é um mapeamento de L^E em L^E, dado por,*

$$\Psi(f)(t) = \zeta_O(u^t) + \phi(u^t_{-\zeta_I(u^t)}|K). \tag{4.8}$$

Desta maneira, a definição de operador (ζ_I,ζ_O)-"aperture" generaliza a definição de W-operador quando $K = M = L$ e $\zeta_O(u) = \zeta(u)_I = 0, \forall u$; também generaliza a definição dos classificadores, quando $M = L$ e $\zeta_O(u) = 0$. A palavra generalização aqui tem o mesmo sentido que na frase: "a classe dos W-operadores generalizam a dos operadores invariantes por translação quando $W = E$".

Da mesma forma que no caso dos W-operadores, aqui também podemos assumir a existência da função característica e então definir o operador "aperture".

4.6 Projeto de Operadores "Aperture"

Nesta seção, analisamos o projeto estatístico de operadores "aperture", isto é, analisamos o que acontece quando, além do operador ser restrito espacialmente por uma janela W, também for restrito nos níveis de cinza observados em W (por um intervalo $K = [-k, k]$), e nos níveis de cinza observados na imagem ideal (por um intervalo $M = [-m, m]$).

Sejam $\zeta_I : L^W \to L$ e $\zeta_O : L^W \to L$ as funções de posicionamento da "aperture" na imagem observada e na ideal, respectivamente. Seja $\mathbf{X}$ uma variável aleatória tal que $\mathbf{X} = (X_1, X_2, \ldots, X_n)$, onde $X_i \in L$.

Seja $\mathbf{X}^*$ a variável aleatória $\mathbf{X}$ posicionada por ζ_I e restrita por K, isto é, $\mathbf{X}^* = (\mathbf{X} - \zeta_I(\mathbf{X}))^* = (X_1^*, X_2^*, \ldots, X_n^*)$, onde $(\bullet)^*$ indica a restrição, isto é,

$$X_j^* = \begin{cases} X_j & : \quad -k \leq X_j \leq k \\ k & : \quad X_j > k \\ -k & : \quad X_j < -k \end{cases} \tag{4.9}$$

Geometricamente, as observações dentro de $W \times K$ não são modificadas e as que caem fora de $W \times K$ são projetadas verticalmente para dentro dos limites de $W \times K$ (de cima ou de baixo).

Como vimos, as funções características da forma $\psi(\mathbf{X}) = \zeta_O(X) + \phi(\mathbf{X}^*)$, $\psi : L^W \to L$ e $\phi : K^W \to M$, caracterizam os operadores "aperture". O teorema a seguir demonstra quem deve ser o estimador que minimiza o MSE de um operador "aperture".

Teorema 4.10 *O MSE de um operador "aperture", Ψ, caracterizado por uma função característica ψ, que é caracterizada por uma função K-característica ϕ, é dado por:*

$$MSE(\Psi) = E[((Y - \zeta_O(\mathbf{X})) - \phi(\mathbf{X}^*))^2] \tag{4.10}$$

Dem.:

$$\begin{aligned} MSE(\Psi) &= MSE(\psi) && (4.11)\\ &= E[(Y - \psi(\mathbf{X}))^2] = && (4.12)\\ &= E[(Y - (\phi(\mathbf{X}^*) + \zeta_O(\mathbf{X})))^2] && (4.13)\\ &= E[((Y - \zeta_O(\mathbf{X})) - \phi(\mathbf{X}^*))^2] && (4.14) \end{aligned}$$

■

A estimação pode ser feita exatamente da mesma forma como na equação 3.17, com

$$\phi_N((\mathbf{x} - \zeta_I(\mathbf{x}))^*) = \lfloor 0.5 + \frac{1}{N((\mathbf{x} - \zeta_I(\mathbf{x}))^*)} \sum_{i=1}^{N((\mathbf{x}-\zeta_I(\mathbf{x}))^*)} (y_i - \zeta_O(\mathbf{x}))|(\mathbf{x} - \zeta_I(\mathbf{x}))^* \rfloor \tag{4.15}$$

Note que a hipótese que ψ é caracterizado por ϕ, isto é, que $\psi(\mathbf{X}) = \zeta(X) + \phi(\mathbf{X}^*)$ é fundamental pois ela implica que M seja tal que:

$$|\lfloor 0.5 + \frac{1}{N((\mathbf{x}-\zeta_I(\mathbf{x}))^*)} \sum_{i=1}^{N((\mathbf{x}-\zeta_I(\mathbf{x}))^*)} (y_i - \zeta_O(\mathbf{x}))|(\mathbf{x}-\zeta_I(\mathbf{x}))^* \rfloor| \leq m \tag{4.16}$$

Se esta condição não vale, então este valor será projetado para dentro de M (porque $\phi : K^W \to M$) e portanto $\psi(\mathbf{X}) \neq \zeta_O(X) + \phi(\mathbf{X}^*)$. Neste caso, o MSE($\Psi$) = MSE($\psi$) pode não ser igual a MSE(ϕ).

4.6.1 Restrição nos níveis de cinza

A análise da restrição é mais complicada para os operadores "aperture" do que para os W-operadores. Para não complicar mais ainda, vamos supor que M é suficientemente grande para não haver restrição nos níveis de cinza de saída.

Seja ψ_W a função característica sobre W. Relativo à variável aleatória Y, o estimador MSE ótimo de ψ_W é dado por $E[Y|\mathbf{X}]$. Porém, se $\mathbf{X}$ é restrito por K e ψ é a função característica sobre $\mathbf{X}^*$, o estimador MSE ótimo será $E[Y|\mathbf{X}^*]$

Assim, o erro da restrição por usar ψ ao invés de ψ_W é dado por:

$$\triangle(\psi, \psi_W) = E[|Y - \zeta(\mathbf{X}) - E[Y|\mathbf{X}^*]|^2] - E[|Y - E[Y|\mathbf{X}]|^2] \tag{4.17}$$

A estimação padrão baseada em realizações envolve estimar $E[y|\mathbf{x}]$. Se o intervalo tem s valores, então há s^n configurações $\mathbf{x}$ para serem estimadas. Um estimador análogo pode ser usado para estimar $E[Y, \mathbf{x}^*]$; entretanto, como o intervalo de X^* tem somente $2k+1$ valores, há somente $(2k+1)^n$ configurações para serem estimadas. Desta forma, a precisão pode ser melhor que o aumento de erro devido à restrição.

A análise da relação entre as "apertures" é, também, mais complicada. Se $W_1 \subset W_2$ e $K_1 \subset K_2$, então as equações 3.22 e 3.23 aplicam-se com $\mathcal{A}_1 = W_1 \times K_1$ e $\mathcal{A}_2 = W_2 \times K_2$ em lugar de W e V, respectivamente; entretanto, se $\mathcal{A}_1$ e $\mathcal{A}_2$ não são ordenados pela inclusão, então as comparações dos filtros estimados como função do tamanho da amostra de treinamento depende da geometria particular das janelas e do modelo dos processos aleatórios envolvidos na geração da imagem. Estendendo a equação 4.17 à esta análise,

$$\triangle(\psi_{\mathcal{A}_1}, \psi_{\mathcal{A}_2}) = E[|Y - \zeta(\mathbf{X}) - E[Y|\mathbf{X}_1^*]|^2] - E[|Y - \zeta(\mathbf{X}) - E[Y|\mathbf{X}_2^*]|^2] \tag{4.18}$$

onde $\mathbf{X}_1^*$ e $\mathbf{X}_2^*$ resultam da projeção das funções para $\mathcal{A}_1$ e $\mathcal{A}_2$, respectivamente, e a diferença pode ser negativa. Assim, mesmo sabendo que $E[\triangle(\psi_{\mathcal{A}_1,N}, \psi_{\mathcal{A}_1})] \to 0$ e $E[\triangle(\psi_{\mathcal{A}_2,N}, \psi_{\mathcal{A}_2})] \to 0$ quando $N \to \infty$, o relacionamento entre os dois erros de precisão para um N particular depende das distribuições multivariadas de $(\mathbf{X}_1^*, Y)$, e $(\mathbf{X}_2^*, Y)$, e estas distribuições podem ser complicadas.

4.7 Exemplo ilustrativo de estimação estatística

Nesta seção, mostraremos um exemplo para ilustrar a estimação estatística das probabilidades condicionais via exemplos de treinamento.

X_1	X_2	X_3	Y_1	#	Y_2	#	Y_3	#	Y_4	#	Y_5	#	Y_6	#
3	3	2	3	1	2	4								
94	93	92	94	1	93	27	92	2						
111	111	111	113	3	112	12	111	96	110	11	109	1		
114	114	114	116	2	115	9	114	62	113	17	112	4	111	1
129	129	129	131	2	130	11	129	84	128	10	127	2		

Tabela 4.1: Alguns exemplos de treinamento

X_1	X_2	X_3	Y
3	3	2	2
94	93	92	93
111	111	111	111
114	114	114	114
129	129	129	129

Tabela 4.2: Alguns exemplos de treinamento

A figura 4.4(a) mostra o resultado da convolução do sinal mostrado na figura 4.4(b) por um núcleo plano de 11 pontos (foi selecionada uma parte do sinal). Este é apenas um de um conjunto de sinais 1D semelhantes que vão ser usados neste exemplo.

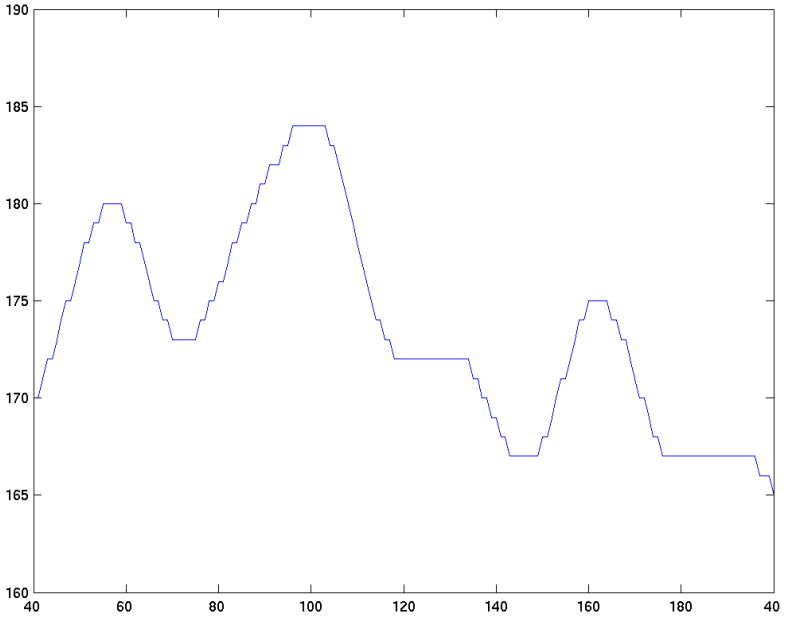
A tabela 4.1 mostra alguns dos 1773 padrões diferentes que foram observados no conjunto completo de sinais. As três primeiras colunas apresentam os valores observados, x_1, x_2, x_3, quando posicionamos a janela $W = \{w_1 = -1, w_2 = 0, w_3 = 1\}$ no domínio do sinal. As colunas seguintes apresentam os respectivos valores observados, y, nos sinais ideais (os sinais antes da convolução) e o número de vezes que cada valor foi observado.

Caso estivéssemos estimando o W-operador MSE ótimo, então os valores associados àquelas configurações seriam os valores arredondados de $E[Y|(3,3,2)]$, $E[Y|(94,93,92)]$, $E[Y|(111,111,111)]$, $E[Y|(114,114,\ 114)]$ e $E[Y|(129,129,129)]$, como mostrados na tabela 4.2

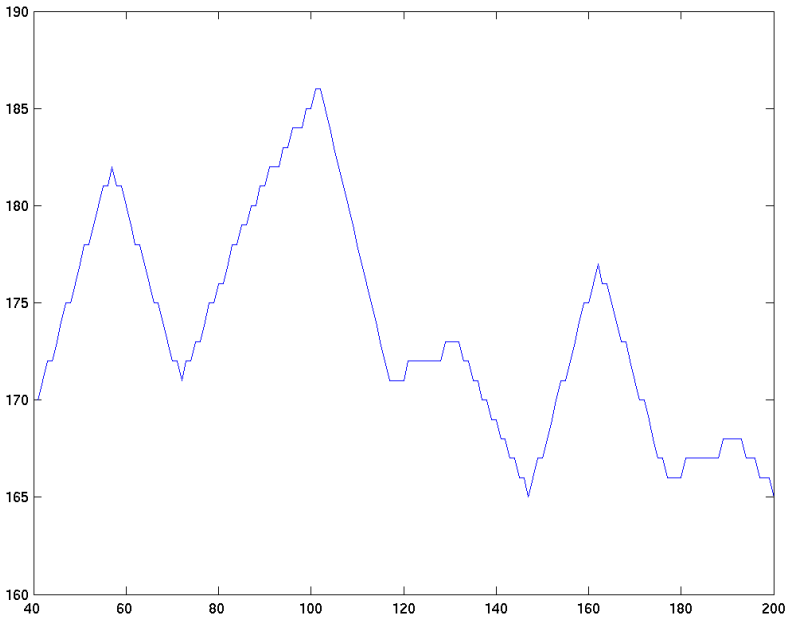
Caso estivéssemos estimando o operador "aperture" sobre a mesma janela, restrito aos intervalos $K = \{-1,0,1\}$ e $M = \{-3,\ldots,3\}$, e com janela posicionada no sinal, então teríamos a tabela 4.3, que apresenta as configurações "aperture".

X_1^*	X_2^*	X_3^*	3	2	1	0	-1	-2	-3
0	0	-1	0	17	334	6026	1143	97	0
1	0	-1	0	0	84	4889	69	0	0
-1	0	0	0	31	320	6680	1199	82	0
0	0	0	29	380	2223	13756	2211	388	29
1	0	0	0	77	1146	6013	359	28	0
-1	0	1	0	0	70	5259	74	0	0
0	0	1	0	94	1180	6701	311	21	0

Tabela 4.3: Exemplos de treinamento restritos por K e M



(a)



(b)

Figura 4.4: Embaçamento para o exemplo ilustrativo: (a) sinal embaçado e (b) sinal original.

X_1^*	X_2^*	X_3^*	Y
0	0	-1	0
1	0	-1	0
-1	0	0	0
0	0	0	0
1	0	0	0
-1	0	1	0
0	0	1	0

Tabela 4.4: Conjunto $\mathcal{M}_{\mathcal{A}}$ dos exemplos rotulados

A tabela 4.4 mostra $\mathcal{M}_{\mathcal{A}}$, isto é, a tabela de exemplos após a etapa de decisão segundo o critério MSE.

No sistema desenvolvido, tanto a tabela 4.2, como a tabela 4.4, podem ser usadas como entrada para a fase de simplificação da representação do W-operador ou, respectivamente, do operador "aperture".

4.8 Posicionamento da "aperture"

Para um operador "aperture" Ψ de parâmetros W, K e M, a função de posicionamento da "aperture", ζ_I, tem forte influência no erro de estimação. Como a função ζ_I posiciona a janela $W \times K$ em algum nível de cinza dentro de L, essa posição deve ser escolhida de tal forma que uma boa parte da imagem observada pela janela W não caia fora de $W \times K$, ou equivalentemente, de forma que a quantidade de sinal projetado para dentro de $W \times K$ seja mínima.

Se $t \in E$ é um ponto do domínio da imagem e $W \subset E$ é a janela que define o operador, seja $h(t)$ o sinal observado em t. Se, por exemplo, escolhermos centralizar a janela $W \times K$ em $(t, h(t))$, caso em que $\zeta_I(h(t)|W) = h(t)$, podemos incorrer em erros significativos se h é um sinal corrompido por ruído aditivo pois boa parte do sinal pode estar fora da "aperture".

Como o erro da restrição K depende da massa de probabilidade $(\mathbf{X}, Y)$ fora de $W \times K$, para reduzí-la, seria interessante escolher o posicionamento em função das observações $X_1, X_2, \ldots, X_n$. Se fazer o posicionamento no sinal, como no exemplo anterior, pode-se incorrer em erros, em outros casos esse posicionamento é conveniente. Por exemplo, no caso de desembaçamento é melhor posicionar no sinal, como mostraram os experimentos.

O posicionamento da "aperture", onde $K = M = [-3, 3]$, é ilustrado nas Figs. 4.5 e 4.6, que mostram ruído branco aditivo e embaçamento, respectivamente. Em cada figura, as partes a e c dão o sinal ideal (não corrompido) e as partes b e d os sinais observados (corrompidos). Sinais são mostrados como pontos sólidos; o sinal $\times$ marca o centro da "aperture", e pontos sombreados mostram as projeções dos pontos que caem fora da "aperture" para dentro dela. Nas partes a e b da Fig. 4.5, o "aperture" é posicionado verticalmente no valor observado, $\mathbf{x}^*$ difere de $\mathbf{x}$ em quatro pontos, e $(y - \zeta(\mathbf{x}))^* \neq y - \zeta(\mathbf{x})$; nas partes c e d, a "aperture" é posicionada verticalmente na mediana de $\mathbf{x}$, $\mathbf{x}^*$ difere de $\mathbf{x}$ em um ponto, e $(y - \zeta(\mathbf{x}))^* = y - \zeta(\mathbf{x})$. Nas partes a e b da Fig. 4.6, a "aperture" é posicionada verticalmente no valor observado, $\mathbf{x} = \mathbf{x}^*$, e $(y - \zeta(\mathbf{x}))^* = y - \zeta(\mathbf{x})$; nas partes c e d, a "aperture" é posicionada verticalmente na mediana de $\mathbf{x}$, $\mathbf{x} = \mathbf{x}^*$, e $(y - \zeta(\mathbf{x}))^* \neq y - \zeta(\mathbf{x})$.

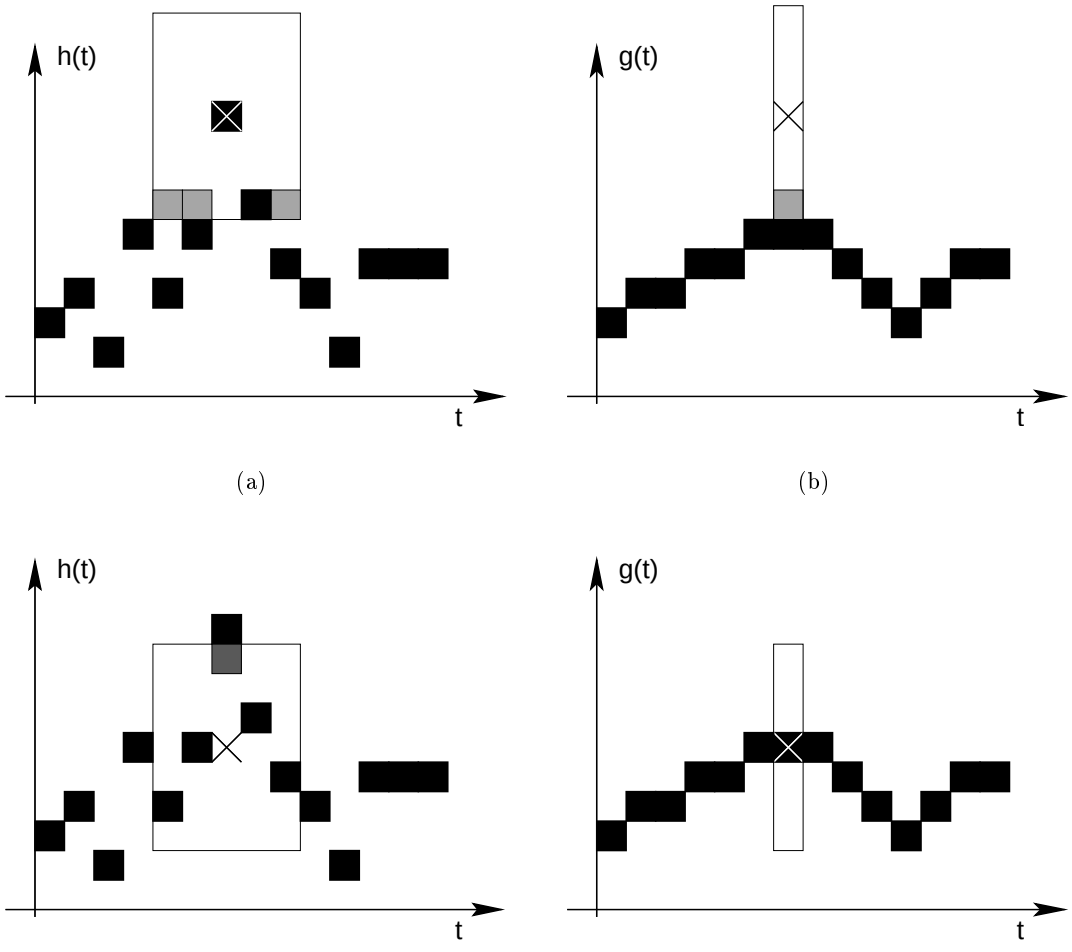


Figura 4.5: Posicionamento da "aperture" para ruído branco aditivo: (a) "aperture" sobre o sinal observado posicionado no valor observado; (b) valor observado no sinal ideal se o posicionamento é no valor observado; (c) "aperture sobre o sinal observado com posicionamento na mediana; (d) valor observado no sinal ideal se o posicionamento é na mediana.

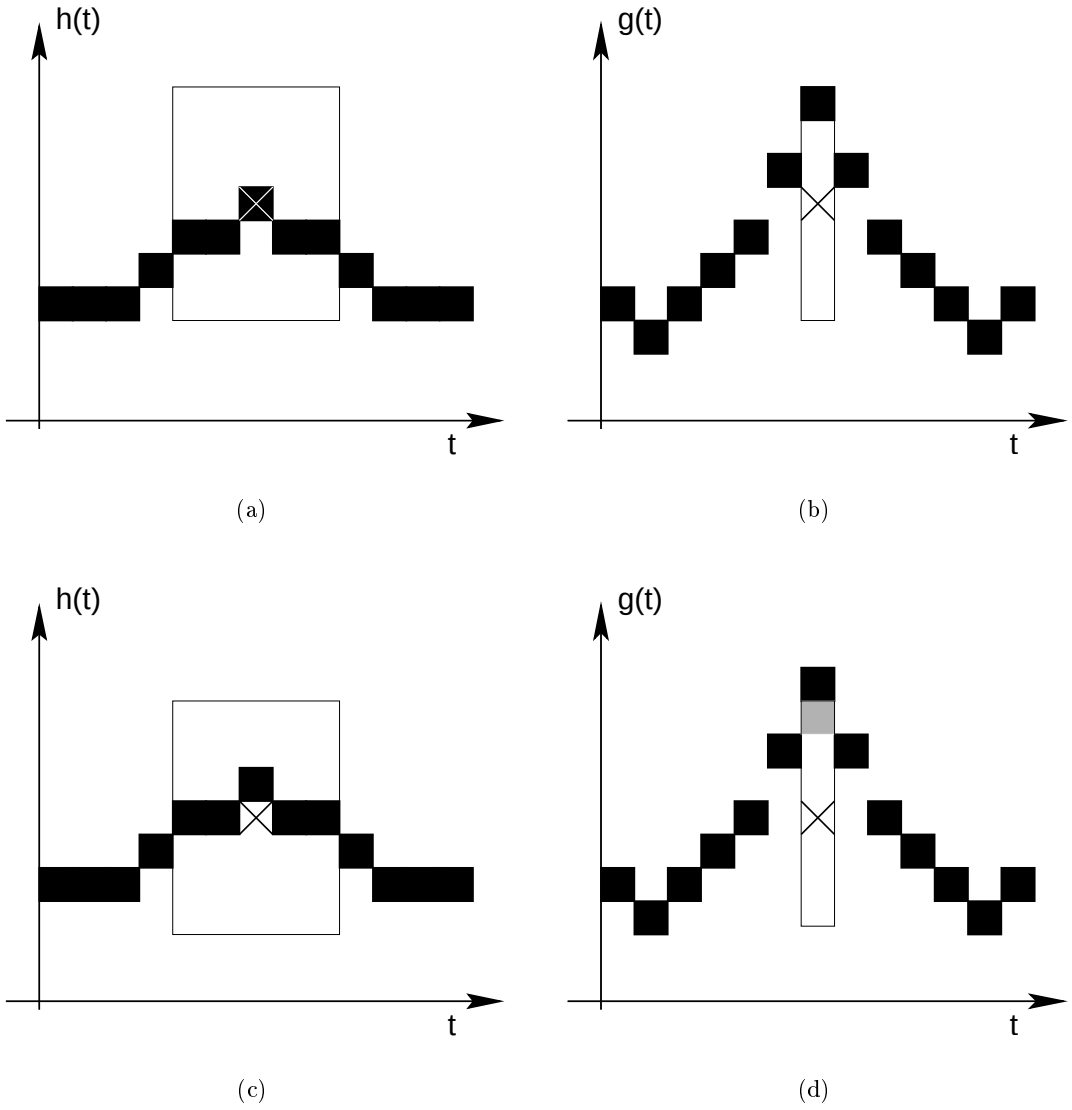


Figura 4.6: Posicionamento da "aperture" para desembaçamento: (a) "aperture" sobre o sinal observado posicionado no valor observado; (b) valor observado no sinal ideal se o posicionamento é no valor observado; (c) "aperture sobre o sinal observado com posicionamento na mediana; (d) valor observado no sinal ideal se o posicionamento é na mediana.

4.9 Estudo simulado

Nesta seção apresentamos um estudo simulado para ilustrar o que acontece quando fazemos a restrição nos níveis de cinza em relação ao quanto da imagem deixamos de observar. Para isso, vamos considerar um passeio aleatório que é corrompido por ruído aditivo. Usando sinais deste tipo, vamos estimar a probabilidade de um ponto do sinal cair fora da "aperture".

O sinal original é formado por um processo do tipo passeio aleatório no domínio discreto que começa em 0 e cresce, ou decresce, uma simples unidade de cada vez com igual probabilidade, isto é, se denominarmos de ξ_t a variável aleatória que rege o passeio aleatório, então $\xi_t \in \{-1, 1\}$ e $P(\xi_t = 1) = P(\xi_t = -1) = \frac{1}{2}$.

Seja $\mathbf{Y}$ um processo aleatório tal que $y_0 = 0$, y_1, ..., y_t, formam uma seqüência de t pontos tais que $y_t = y_{t-1} + \xi_t$. Note que esta seqüência também pode ser vista como um sinal tal que,

- $y_0 = 0$
- $y_1 = \xi_1$
- $y_2 = \xi_1 + \xi_2$
- ...
- $y_t = \sum_{j=1}^{t} \xi_j$

O valor esperado de y_t é dado por:

$$E[y_t] = E[\sum_{j=1}^{t} \xi_{\mathbf{j}}] = \sum_{j=1}^{t} E[\xi_{\mathbf{j}}] = 0, \tag{4.19}$$

onde a segunda igualdade segue da primeira pela independência de $\xi_{\mathbf{j}}$.

A variância de y_t é dada por:

$$Var[y_t] = Var[\sum_{j=1}^{t} \xi_{\mathbf{j}}] = \sum_{j=1}^{t} Var[\xi_{\mathbf{j}}] = t, \tag{4.20}$$

onde a segunda igualdade segue da primeira por causa da independência de $\xi_{\mathbf{j}}$.

Vamos considerar agora a composição do passeio aleatório com um ruído $\mathbf{N}$ que tem distribuição gaussiana discreta de média 0 e variância inteira σ^2. A forma de composição é a adição, isto é, $\mathbf{X} = \mathbf{Y} + \mathbf{N}$. Os t pontos da seqüência aleatória podem ser escritos por $x_0 = y_0 + n_0$, $x_1 = y_1 + n_1$, ..., $x_t = y_t + n_t$. O sinal formado por esse processo é descrito por $x_0, x_1, \ldots, x_t$ tais que:

- $x_0 = n_0$
- $x_1 = \xi_1 + n_1$
- $x_2 = \xi_1 + \xi_2 + n_1$
- ...

- $x_t = \sum_{i=1}^{t} \xi_i + n_t$

O valor esperado de x_t é dado por:

$$E[x_t] = E[\sum_{j=1}^{t} \xi_{\mathbf{j}} + n_t] = \sum_{j=1}^{t} E[\xi_{\mathbf{j}}] + E[n_t] = 0, \tag{4.21}$$

onde a segunda igualdade segue da primeira também pela independência entre $\xi_{\mathbf{j}}$ e $N(0, \sigma^2)$.

A variância de x_t é dada por:

$$\begin{aligned} Var[x_t] &= Var[\sum_{j=1}^{t} \xi_{\mathbf{j}} + n_t] && (4.22)\\ &= Var[\sum_{j=1}^{t} \xi_{\mathbf{j}}] + Var[n_t] && (4.23)\\ &= t + \sigma^2 && (4.24) \end{aligned}$$

onde a segunda igualdade segue da primeira pela independência entre $\xi_{\mathbf{j}}$ e $\mathbf{N}$, e a última igualdade segue da hipótese.

Para estudar o comportamento destes sinais em relação a uma janela espacial $W = \{w_{-m}, w_{-m+1}, \ldots, w_m\}$ e também a uma janela nos níveis de cinza $K = [-k, k]$. Vamos considerar a seqüência formada pelo processo $\mathbf{X}$ quando observada pela janela W centrada espacialmente na origem e posicionada verticalmente pela função $\zeta(\mathbf{x})$, que neste caso, por simplicidade, tomamos ser o valor observado na origem, $\zeta(\mathbf{x}) = x_0 = n_0$.

O problema que estamos interessados é saber quando o valor observado no ponto j cai na "aperture". Isso equivale a modelar se a variável aleatória $x'_j = x_j - x_0 = x_j - n_0$ cai na "aperture". A seqüência gerada $\mathbf{X}'$ tal que,

- $x'_0 = x_0 - x_0 = 0$
- $x'_1 = x_1 - x_0 = \xi_1 + n_1 - n_0$
- $x'_2 = x_2 - x_0 = \xi_2 + \xi_1 + n_2 - n_0$
- ...
- $x'_t = x_t - x_0 = \sum_{j=1}^{t} \xi_j + n_t - n_0$

Já vimos que uma forma de estudar uma variável aleatória é estudar os seus momentos. Assim, vamos calcular a média e a variância de x'_j.

A média de x'_j é dada por,

$$\begin{aligned} E[x'_t] &= E[x_t - x_0] \\ &= E[x_t] - E[x_0] \\ &= E[\textstyle\sum_{j=1}^{t} \xi_{\mathbf{j}}] + E[n_t] - E[n_0] \\ &= 0 \end{aligned} \tag{4.25}$$

por causa da independência entre x_j e x_0, e também pelo fato que $E[n_j] = 0, \forall j$.

A variância de x'_j é simples de calcular por causa da independência entre x_j e x_0.

$$\begin{aligned} Var[x'_t] &= Var[x_t - x_0] && (4.26)\\ &= Var[x_t] + Var[x_0] && (4.27)\\ &= t + \sigma^2 + \sigma^2 = t + 2\sigma^2 && (4.28) \end{aligned}$$

Aqui, a segunda igualdade segue da primeira pela independência e pelo fato que a variância da soma de variáveis independentes é a soma das variâncias dessas variáveis.

A probabilidade de uma observação x'_j cair fora da "aperture", e assim fazer com que a restrição seja aplicada, é limitada, de acordo com a desigualdade de Chebyshev, da seguinte forma,

$$P(|x'_j| \geq k+1) \leq \frac{\mathrm{Var}[x'_j]}{(k+1)^2} = \frac{|j| + 2\sigma^2}{(k+1)^2} \tag{4.29}$$

Para um σ^2 fixo, a probabilidade é pequena para pequenos $|j|$ e grande k, significando que ela é negligível para W pequenos e K grande. É claro que não queremos W muito pequeno ou senão nós teríamos um erro de restrição muito grande; também não podemos fazer K muito grande sem incorrer em erros de estimação inaceitáveis.

Caso o posicionamento fosse feito pela média dos valores de W, a média continuaria zero, mas a variância seria,

$$\begin{aligned} Var[x'_t] &= Var[x_t - 0] && (4.30)\\ &= Var[x_t] && (4.31)\\ &= t + \sigma^2 && (4.32) \end{aligned}$$

e a desigualdade de Chebyshev seria dada por,

$$P(|x'_j| \geq k+1) \leq \frac{\mathrm{Var}[x'_j]}{(k+1)^2} = \frac{|j| + \sigma^2}{(k+1)^2} \tag{4.33}$$

Porém, a desigualdade de Chebyshev é muito frouxa, significando que nas equações 4.29 e 4.33 a probabilidade de x'_j estar fora da "aperture" deve ser substancialmente menor que o limite teórico dado.

Para verificar isto, vamos simular um passeio aleatório com 20% de ruído gaussiano discreto aditivo, e estudar o que acontece na posição 10 a partir da origem, x_{10}, se a janela estiver centrada no valor observado. A figura 4.7 mostra os gráficos da probabilidade para as variâncias $5, 10, 15, 20$. Em cada caso, $\frac{\mathrm{Var}[x'_j]}{(k+1)^2}$ (O. C.) é apresentado junto com a correspondente probabilidade estimada de x'_{10} estar fora da "aperture" (O. A.). Quando $k \to \infty$, o limite e a probabilidade se aproximam de zero. É interessante notar que, para k pequeno, o limite de Chebyshev dá valores muito maiores do que as probabilidades estimadas.

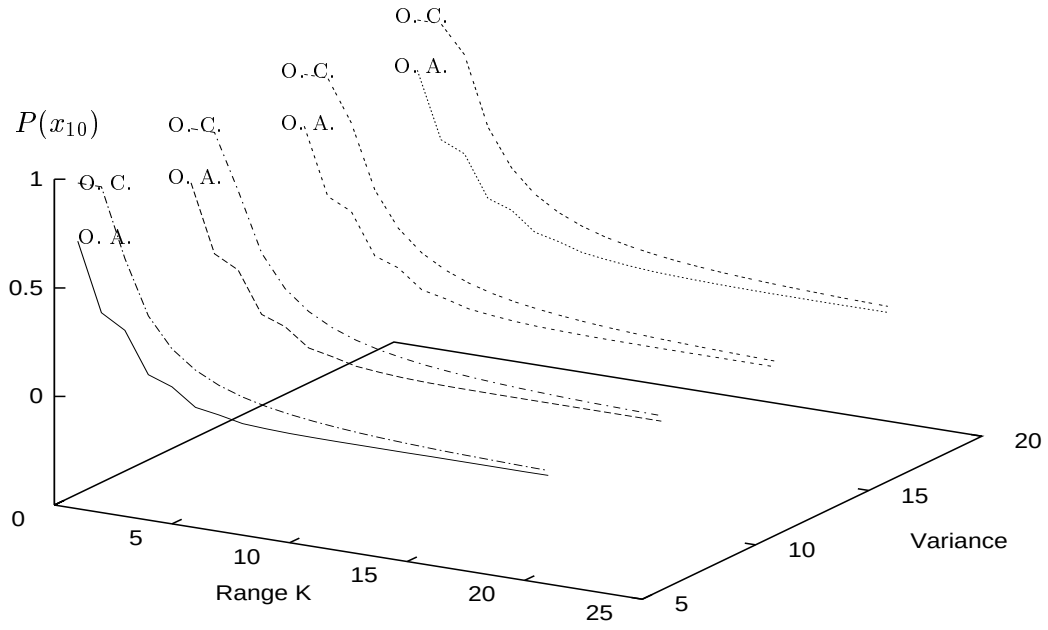


Figura 4.7: Limites de Chebyshev calculados e valores simulados

Capítulo 5

Representação e indução via árvores de decisão

No capítulo anterior, vimos o projeto estatístico de operadores "aperture" a partir de pares de imagens observadas e ideais, (h_i,g_i), respectivamente. Fixada a restrição, (W,K,M), e a regra de posicionamento da janela, estimamos as distribuições conjuntas, $y|\mathbf{x}$. Fixado um critério de risco, decidimos o valor do estimador $\hat{y}$ que minimiza o erro definido pelo critério de risco para cada $\mathbf{x}$ observado. A estimação atribui um valor para aquelas configurações observadas nas imagens h_i. No entanto, a função característica pode ainda não estar completamente definida pois não sabemos qual o valor dela para as configurações não observadas.

A alternativa mais simples seria atribuir uma constante inteira para elas. Outra alternativa seria atribuir a mediana ou a média amostral de Y (levando-se em conta todos os $\mathbf{x}$ observados) se queremos minimizar o MAE, ou o MSE, respectivamente. Porém, essas alternativas podem levar a erros muito grandes.

A alternativa que primeiro testamos foi a representação via árvores de decisão, que são muito conhecidas da comunidade de *aprendizado computacional* [37, 121, 122, 7, 119]. Elas são simples de implementar e é conhecido que elas generalizam bem para as configurações não observadas.

Embutido na questão da representação, está a preocupação de representarmos computacionalmente os operadores estimados de forma a termos uma representação compacta (que não ocupe muito espaço de memória) e de fácil aplicação, isto é, que o custo para achar o valor a ser atribuído a uma certa configuração não seja alto. No caso dos operadores ótimos lineares vistos no capítulo 2, a representação é muito simples (um vetor é suficiente) e o custo da representação é igual ao número de pontos da janela, isto é, basta termos os coeficientes da convolução. Porém, no caso dos operadores "aperture", isto não é tão simples.

Neste capítulo tratamos da indução e da representação no contexto do projeto de estatístico de operadores "aperture" via árvores de decisão. Ao final do capítulo, apresentamos um estudo extenso da aplicação dos operadores "aperture" para filtragem de ruído gaussiano e restauração de embaçamento ("deblurring").

5.1 Algoritmo básico de indução de uma DT

Os vários algoritmos que constroem (ou *induzem*, como é normalmente chamado este processo) uma árvore de decisão a partir de um conjunto de exemplos rotulados são muito similares no sentido que eles são baseado na estratégia de *divisão e conquista* [44].

A seguinte definição caracteriza um conjunto de exemplos rotulados.

Definição 5.1 *Um conjunto de exemplos rotulados é um conjunto de pares ordenados* $(\mathbf{x}, l)$*, onde* $\mathbf{x}$ *é uma configuração observada e* l *é o rótulo associado a ela após a aplicação do critério de minimização de erro.*

A próxima definição caracteriza um conjunto de exemplos rotulados em relação a mistura dos valores, ou rótulos, atribuídos às configurações.

Definição 5.2 *Seja* $\mathcal{M}_{\mathcal{A}}$ *um conjunto de exemplos rotulados. Dizemos que* $\mathcal{M}_{\mathcal{A}}$ *é puro se todos os exemplos dele tem o mesmo valor, ou rótulo.*

A idéia básica para construir uma árvore de decisão a partir de um conjunto de exemplos rotulados, $\mathcal{M}_{\mathcal{A}}$, é achar um teste, digamos T_0, que particiona $\mathcal{M}_{\mathcal{A}}$ em dois ou mais subconjuntos com pouca ou nenhuma mistura de rótulos, isto é, particionamos o conjunto em partes mais "puras". Para cada subconjunto não puro, aplicamos a mesma idéia novamente e achamos novos testes que tornam os subconjuntos mais puros, digamos $T_{1.1}, T_{1.2}, \ldots, T_{1.m}$, onde o par de índices $i.j$ indica nível (i) e o nó (j) do teste na árvore, respectivamente. Aplicando a idéia recursivamente até que não seja mais possível particionar (quando o subconjunto é unitário), ou até que os subconjuntos sejam puros, teremos uma partição de $\mathcal{M}_{\mathcal{A}}$ onde cada parte é resultante de uma seqüência de testes e, mais importante, é pura.

Uma estrutura de dados adequada para representar tal particionamento, baseado em uma seqüência de testes, é a estrutura de árvore enraizada [44]. Cada nó da árvore é associado a um teste, e cada folha é associada com um rótulo.

Por construção, se percorremos a árvore para uma configuração observada na fase de estimação, então o rótulo associado a ela será o mesmo que foi associado à configuração após a decisão. Isto garante que a função ψ parcialmente definida pelas configurações observadas será uma extensão consistente com a decisão.

5.1.1 Algoritmo básico de indução

O algoritmo básico induz a árvore de cima para baixo ("top-down"), isto é, da raiz para as folhas, seguindo recursivamente três passos para cada nó da árvore (não consideraremos os casos triviais no algoritmo). Seja $\mathcal{M}_{\mathcal{A}}^u \subset \mathcal{M}_{\mathcal{A}}$ o conjunto de exemplos rotulados considerados no nó u da árvore, e sejam $y, y_j \in L$ dois rótulos desses exemplos.

1. Se $\mathcal{M}_{\mathcal{A}}^u \neq \emptyset$ e $\mathcal{M}_{\mathcal{A}}^u$ é puro, isto é, todas as configurações $\mathbf{x}_j \in \mathcal{M}_{\mathcal{A}}^u$ são tais que seus rótulos são iguais (i.e., $y_j = y, \forall j$), então u é uma folha e receberá o rótulo y.

2. Se $\mathcal{M}^u_{\mathcal{A}} \neq \emptyset$ não é puro, considere **todas** as possíveis formas de particionar $\mathcal{M}^u_{\mathcal{A}}$ em duas ou mais partes, e avalie cada uma medindo quão bem elas particionam o conjunto.

3. Particione $\mathcal{M}^u_{\mathcal{A}}$ (i.e., divida os exemplos e crie um novo nó filho para cada parte) segundo a partição que recebeu a maior avaliação[1]

Há vários critérios diferentes (*medidas de impureza*) para avaliar a pureza da partição. Em termos de indução (generalização), não é conclusivo se um critério é melhor que outro [116].

Neste trabalho implementamos e usamos um *critério* chamado *"twoing"* [116]. Esse critério é normalmente usado para induzir árvores de decisão binárias e mede a qualidade de uma partição de dois subconjuntos, o esquerdo ("**l**eft") e o direito ("**r**ight").

Seja S_r e S_l os dois subconjuntos resultantes da partição de $\mathcal{M}^u_{\mathcal{A}}$. Seja $P_r = P\{\mathbf{x} \in S_r : \mathbf{x} \in \mathcal{M}^u_{\mathcal{A}}\}$ e seja $P_l = P\{\mathbf{x} \in S_l : \mathbf{x} \in \mathcal{M}^u_{\mathcal{A}}\}$, isto é, a probabilidade de uma configuração $\mathbf{x}$ estar no subconjunto S_r, ou S_l, respectivamente. Seja $p(y_j|S_r) = P\{y = y_j : \mathbf{x}_j \in S_r\}$ e seja $p(y_j|S_l) = P\{y = y_j : \mathbf{x}_j \in S_l\}$, isto é, a probabilidade de uma configuração $\mathbf{x}_j$ que tem rótulo y_j estar em S_r, ou S_l, respectivamente. A medida "twoing" é definida por:

$$T_W(S_r, S_l) = P_r P_l [\sum_{y_j \in M} |p(y_j|S_l) - p(y_j|S_r)|]^2 \tag{5.1}$$

onde M é o conjunto de todos os rótulos que aparecem em $\mathcal{M}^u_{\mathcal{A}}$.

A interpretação desta medida é muito simples: o fator $P_r P_l$ é maximizado pela divisão balanceada dos exemplos entre S_r e S_l. A soma, por sua vez, é maximizada quando a mistura dos rótulos entre S_r e S_l é minimizada. O "twoing" então é uma medida de pureza que deve ser maximizada para que os subconjuntos fiquem mais puros.

5.1.2 O algoritmo OC1 para indução de uma ODT

Para construir árvores de decisão oblíquas, adaptamos e usamos o *algoritmo OC1* desenvolvido por Murthy [117]. A razão de usarmos ele, ao invés de outros algoritmos que constroem árvores oblíquas [116, 123, 110, 124], é devido a uma heurística que introduz uma parte aleatória para achar eficientemente os coeficientes de um hiperplano de decisão oblíquo em cada nó da árvore. É importante salientar que esta heurística não garante que o hiperplano achado é o que melhor divide os exemplos. Porém, como o processo inicia-se pelo melhor hiperplano paralelo e depois tenta melhorá-lo, então cada partição feita por um hiperplano oblíquo nunca vai ser pior do que a feita pelo hiperplano paralelo. Esta última condição assegura que o método será, no mínimo, tão bom quanto os métodos clássicos de indução de árvores de decisão paralelas.

O algoritmo OC1 consiste de um número pré-definido de iterações das seguintes (quatro) partes principais:

1. Encontre o hiperplano paralelo H que melhor divide o conjunto de exemplos classificados $\mathcal{M}_{\mathcal{A}}$, de acordo com o critério de pureza escolhido.

[1] Caso exista mais que uma, escolha qualquer uma delas.

2. Repita R vezes:

   Faça uma busca gulosa local [44] que seleciona o primeiro hiperplano H' cuja medida de pureza é melhor do que a do hiperplano H (este método também é conhecido como "hill climbing" [125]). O objetivo disto é perturbar o hiperplano atual para achar um outro que melhore a pureza da divisão. Quando a impureza de H' for menor que de H, então $H \leftarrow H'$.

   Faça no máximo J vezes, ou até que a impureza diminua: perturbe o hiperplano H em uma direção aleatória e faça H igual a esse hiperplano modificado caso a impureza diminua.

   Escolha um hiperplano H' aleatoriamente.

Note que há um número exponencial [110] de maneiras de particionar (resultando em partições diferentes) o conjunto de exemplos rotulados em um dado nó u da árvore, $\mathcal{M}_{\mathcal{A}}^{u}$, com um hiperplano. Daí a necessidade de um método eficiente (baseado na heurística "hill climbing"). Abaixo detalhamos a aplicação deste processo.

A equação de um hiperplano no espaço de dimensão $|W|$ (onde o reticulado está mergulhado) é dada por:

$$\sum_{i=1}^{|W|}(a_i x_i) + a_{|W|+1} = 0, \tag{5.2}$$

onde $|W|$ é o número de pontos na janela que define a função característica ψ e $a_i \in R$ são os coeficientes do hiperplano.

Se substituímos um exemplo rotulado $(\mathbf{x}_j, l_j) = ((\mathbf{x}_{j,1}, \ldots, \mathbf{x}_{j,|W|}), l_j) \in \mathcal{M}_{\mathcal{A}}^{u}$ na equação 5.2, teremos:

$$\sum_{i=1}^{|W|}(a_i \mathbf{x}_{j,i}) + a_{|W|+1} = c_j,$$

onde o sinal de c_j indica se o ponto $\mathbf{x}_j$ do reticulado está acima ou abaixo do hiperplano (se $c_j > 0$ então $\mathbf{x}_j$ está acima).

Se o hiperplano divide $\mathcal{M}_{\mathcal{A}}^{u}$ perfeitamente, então todos os exemplos rotulados com o mesmo rótulo terão o mesmo sinal, i.e., se $l_j = l_i$, então o sinal de c_j é igual ao sinal de c_i.

Se o hiperplano não divide $\mathcal{M}_{\mathcal{A}}^{u}$ perfeitamente, então o algoritmo ajusta os coeficientes do hiperplano individualmente, achando o valor ótimo local para um coeficiente de cada vez. Este ajuste é feito da seguinte maneira: seja a_p o coeficiente a ser ajustado (uma variável) e seja $(\mathbf{x}_j, l_j)$ o exemplo rotulado em $\mathcal{M}_{\mathcal{A}}^{u}$. Substituindo as coordenadas de $\mathbf{x}_j$ na equação do hiperplano, sabemos se o exemplo está acima, ou abaixo, do hiperplano verificando o sinal de c_j.

$$U_p = \frac{a_p \mathbf{x}_{j,p} - c_j}{\mathbf{x}_{j,p}} < a_p \tag{5.3}$$

Pela definição, $\mathbf{x}_j$ está acima do hiperplano se $a_i > U_i, \forall i$ e abaixo caso contrário (assumindo $\mathbf{x}_{j,p} > 0$, o que pode ser assegurado via uma transformação linear que leva todas as configurações para o quadrante positivo).

Substituindo todos os pontos de $\mathcal{M}_{\mathcal{A}}^{u}$ na equação 5.3, obteremos $|\mathcal{M}_{\mathcal{A}}^{u}|$ restrições para o valor a_p. Para achar o valor a_p que melhor divide $\mathcal{M}_{\mathcal{A}}^{u}$, ordena-se $U = \{U_1, \ldots, U_{|\mathcal{M}_{\mathcal{A}}^{u}|}\}$ e computa-se a impureza da divisão para um novo a_r, que é o ponto central entre cada par de $U_i's$ consecutivos. Isto é feito com $|\mathcal{M}_{\mathcal{A}}^{u}| - 1$ comparações.

O algoritmo 1 mostra a forma de perturbar o hiperplano corrente. Neste algoritmo, P_{stag} é um parâmetro que designa a "probabilidade de estagnação", isto é, a probabilidade que um hiperplano seja perturbado a uma localização que não muda a medida de impureza. Este parâmetro decresce cada vez que o algoritmo faz uma "perturbação estagnante".

O OC1 permite-nos escolher 3 modos de perturbar os coeficientes, quais sejam: seqüencial, o primeiro melhor e o aleatório [117]. Em nossos experimentos, testamos apenas a ordem seqüencial. Em [117], os autores dizem que a ordem das perturbações dos coeficientes não parece afetar a acurácia da classificação.

Algorithm 1 Perturb(hiperplano H, índice p)

1: **for** $j = 0$ to $|\mathcal{M}_W^t|$ **do**
2: $U_j \leftarrow \frac{a_p \mathbf{x}_{j,p} - c_j}{\mathbf{x}_{j,p}}$; {Compute os valores de U_p}
3: **end for**
4: Sort(U) ; {Ordena U em ordem não decrescente}
5: $a_p' \leftarrow$ melhor quebra univariável do vetor ordenado U ;
6: $H' \leftarrow H(a_p, a_p')$; {O novo hiperplano é obtido substituindo a_p por a_p'}
7: **if** impurity(H') < impurity(H) **then**
8: $a_p \leftarrow a_p'$;
9: $P_{move} \leftarrow P_{stag}$;
10: **else if** impurity(H') = impurity(H) **then**
11: $a_p \leftarrow a_p'$ with probability P_{move};
12: $P_{move} \leftarrow P_{move} - 0.1 * P_{stag}$;
13: **end if**

Até agora explicamos as duas partes principais do algoritmo. Agora vejamos duas partes não determinísticas do algoritmo. Elas são importantes para evitar soluções mínimas locais de medidas de impureza.

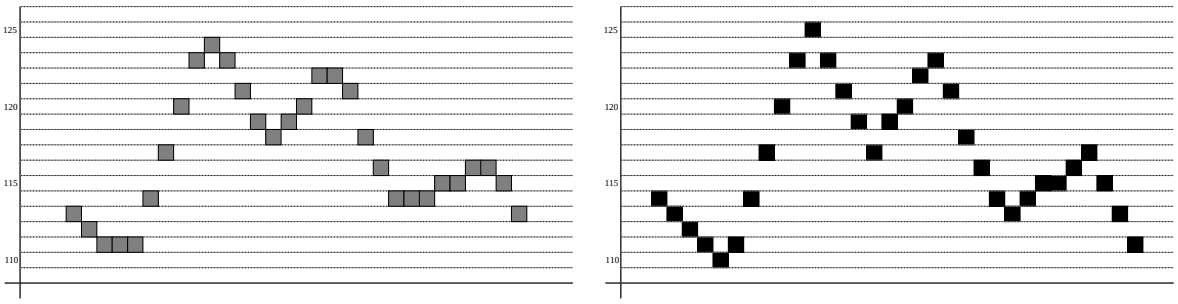
A primeira parte não-determinística tenta perturbar o hiperplano em uma direção aleatória, um número especificado de vezes. Para isso, escolhe-se um vetor aleatório $\mathbf{v} = (v_1, \ldots, v_{|W|+1})$ e perturba-se o hiperplano por um certo fator $\alpha \in R$ na direção de $\mathbf{v}$.

$$H' = \sum_{i=1}^{|W|} (a_i + \alpha v_i)\mathbf{x}_i + (a_{|W|+1} + \alpha v_{|W|+1}) \tag{5.4}$$

Se a medida de impuridade do novo hiperplano decresce, então o algoritmo sai do laço e começa a perturbação determinística para os coeficientes individualmente.

O ponto importante desta parte é achar um valor ótimo para α. O modo de fazer isso é o mesmo usado para achar o valor ótimo para os coeficientes individuais pois a substituição de todos os exemplos correntes do nó $\mathcal{M}_{\mathcal{A}}^u$ na equação, resultará em $|\mathcal{M}_{\mathcal{A}}^u|$ restrições para o valor de α.

A segunda parte não-determinística do algoritmo é baseada na idéia de evitar um mínimo local fazendo diversas buscas locais. Ela é útil quando o algoritmo não pode escapar de um mínimo local somente com a perturbação aleatória. Neste caso, o algoritmo irá recomeçar a busca a partir de um novo hiperplano aleatoriamente escolhido.



(a) (b)

Figura 5.1: Embaçamento para o exemplo ilustrativo: (a) sinal embaçado e (b) sinal original.

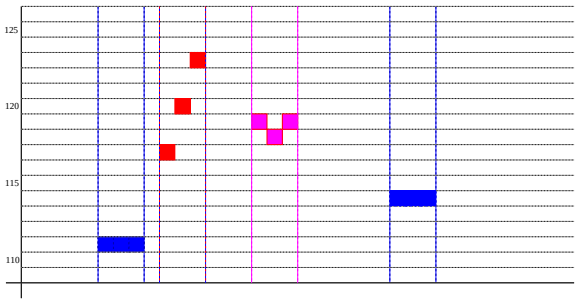


Figura 5.2: Quatro posicionamentos de W e os padrões escolhidos.

5.2 Exemplo ilustrativo

Nesta seção, mostraremos um exemplo completo de como projetar operadores "aperture" usando árvores de decisão. Depois, mostramos o resultado da aplicação do operador projetado em um sinal de teste.

A figura 5.1(a) mostra o resultado da convolução do sinal mostrado na figura 5.1(b) pelo núcleo $\{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}$. Estes dois sinais serão usados para estimar as probabilidades condicionais $P(Y|\mathbf{X})$ para cada $\mathbf{x}$ observado.

A figura 5.2 ilustra quatro posições $\{5, 9, 15, 24\}$ do sinal onde posicionamos uma janela de três pontos, $W = \{w_1 = -1, w_2 = 0, w_3 = 1\}$. A janela é transladada sobre todos os pontos do domínio dos sinais observados. Escolhemos estas posições e destacamos as quatro configurações para ilustrar melhor.

A tabela 5.1 mostra os padrões escolhidos e seus respectivos valores observados nos sinais ideais (os sinais antes da convolução).

A tabela 5.2 mostra o conjunto dos exemplos coletados com uma janela de três pontos ($W = \{w_1 = -1, w_2 = 0, w_3 = 1\}$) e, ao lado, seus respectivos valores observados quando restritos ao intervalo $K = \{-2, -1, \ldots, 2\}$.

A tabela 5.3 mostra $\mathcal{M}_{\mathcal{A}}$, isto é, a tabela após a etapa de decisão segundo o critério MSE. Nela, cada exemplo aparece com seu rótulo final apenas uma vez, independentemente de quantas vezes ele

X_1	X_2	X_3	Y
111	111	111	110
117	120	123	120
119	118	119	117
114	114	114	113

Tabela 5.1: Três exemplos de treinamento

X_1	X_2	X_3	Y		X_1^*	X_2^*	X_3^*	Y^*
114	113	112	113		1	0	-1	0
113	112	111	112		1	0	-1	0
112	111	111	111		1	0	0	0
111	111	111	110		0	0	0	-1
111	111	114	111		0	0	2	0
111	114	117	114		-2	0	2	0
114	117	120	117		-2	0	2	0
117	120	123	120		-2	0	2	0
120	123	124	123		-2	0	1	0
123	124	123	125		-1	0	-1	1
124	123	121	123		1	0	-2	0
123	121	119	121		2	0	-2	0
121	119	118	119		2	0	-1	0
119	118	119	117		1	0	1	-1
118	119	120	119		-1	0	1	0
119	120	122	120		-1	0	2	0
120	122	122	122		-2	0	0	0
122	122	121	123		0	0	-1	1
122	121	118	121		1	0	-2	0
121	118	116	118		2	0	-2	0
118	116	114	116		2	0	-2	0
116	114	114	114		2	0	0	0
114	114	114	113		0	0	0	-1
114	114	115	114		0	0	1	0
114	115	115	115		-1	0	0	0
115	115	116	115		0	0	1	0
115	116	116	116		-1	0	0	0
116	116	115	117		0	0	-1	1
116	115	113	115		1	0	-2	0
115	113	111	113		2	0	-2	0

Tabela 5.2: Exemplos de treinamento

	X_1^*	X_2^*	X_3^*	Y^*
1	-2	0	2	0
2	-2	0	1	0
3	-2	0	0	0
4	-1	0	-1	1
5	-1	0	0	0
6	-1	0	1	0
7	-1	0	2	0
8	0	0	-1	1
9	0	0	0	-1
10	0	0	1	0
11	0	0	2	0
12	1	0	-2	0
13	1	0	-1	0
14	1	0	0	0
15	1	0	1	-1
16	2	0	-2	0
17	2	0	-1	0
18	2	0	0	0

Tabela 5.3: Rotulação final

apareceu no sinal. A figura 5.3 mostra todos as possíveis configurações do reticulado arranjadas em forma do diagrama de Hasse. Os números impressos acima dos pontos do reticulado representam os rótulos finais. Os pontos sem rótulos são as configurações não observadas no treinamento. Esta tabela é a entrada principal do sistema para a fase de simplificação da representação do operador.

Vamos mostrar como construir o primeiro nó[2] de uma árvore de decisão paralela para representar este operador usando o critério "twoing". Começando do conjunto $\mathcal{M}_{\mathcal{A}}$, o algoritmo tem que testar todas as possíveis maneiras de dividir $\mathcal{M}_{\mathcal{A}}$ em dois subconjuntos diferentes usando hiperplanos paralelos aos eixos do diagrama. Há 4 hiperplanos representativos para a variável X_1 ($X_1 = -1.5$, $X_1 = -0.5$, $X_1 = 0.5$, $X_1 = 1.5$), e outros 4 para a variável X_3 ($X_3 = -1.5$, $X_3 = -0.5$, $X_3 = 0.5$, $X_3 = 1.5$). A variável X_2 vale 0 para todos os exemplos e, portanto, não precisa ser considerada. Denotemos por $TW(X_i = l)$ o valor do "twoing" para o hiperplano $X_i = l$ ($i = 1, 3$ e $l \in \{-1.5, -0.5, 0.5, 1.5\}$). Computando o valor "twoing" para cada hiperplano, temos os seguintes valores: $TW(X_1 = -1.5) \approx 0.0395$, $TW(X_1 = -0.5) \approx 0.0314$, $TW(X_1 = 0.5) \approx 0.0314$, $TW(X_1 = 1.5) \approx 0.0395$, $TW(X_3 = -1.5) \approx 0.0247$, $TW(X_3 = -0.5) \approx 0.0988$, $TW(X_3 = 0.5) \approx 0.0314$, $TW(X_3 = 1.5) \approx 0.0395$. O hiperplano que maximiza TW é $X_3 = -0.5$. Os subconjuntos resultantes de dividir em $X_3 = -0.5$ são $\{0 : (12, 13, 16, 17); 1 : (4, 8)\}$ e $\{-1 : (9, 15); 0 : (1..3, 5..7, 10, 11, 14, 18)\}$, onde os números fora dos parenteses são os rótulos associados aos exemplos, e os números dentro dos parenteses são as posições dos exemplos na tabela 5.3.

Para cada subconjunto, um nó filho é criado e o processo é repetido novamente se o subconjunto não é puro. A figura 5.4 mostra a árvore paralela resultante, os subconjuntos associados a cada nó, e as respectivas equações dos hiperplanos associados a cada nó. As folhas da árvore mostram o valor

[2]Os demais nós seguem do mesmo algoritmo

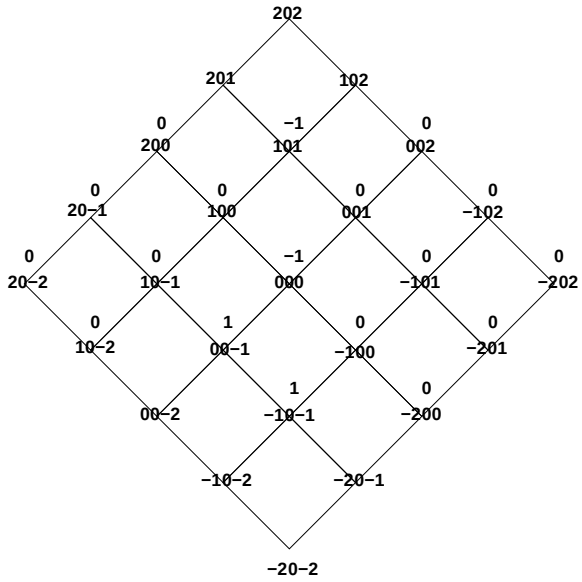


Figura 5.3: Diagrama de Hasse do reticulado

a ser atribuido à configuração $\mathbf{x}$. A figura 5.5 mostra os hiperplanos imersos no reticulado como linhas tracejadas. Estas linhas estão rotuladas da seguinte maneira: *"root"* é a linha que representa a partição da raiz da árvore; *right (r)* representa o filho direito de um nó e *left (l)* representa o filho esquerdo de um nó. Algumas linhas são rotuladas com combinações dos rótulos *right, r*, ou *left, l*, como em *lrll*. Estas combinações representam o caminho a ser seguido para alcançar aquele nó na árvore de decisão.

Para computar $\psi(\mathbf{x})$, começamos substituindo $\mathbf{x}$ na equação do nó raiz e checando se o resultado é maior do que zero. Se ele for maior do que zero, então seguimos para o nó do filho direito; se ele não é maior, então seguimos para o filho esquerdo. Isto é feito recursivamente até alcançarmos uma folha, que contém o valor a ser atribuido à configuração.

A figura 5.6 mostra a árvore de decisão oblíqua gerado pelo algoritmo OC1 para a mesma tabela de treinamento. A figura 5.7 mostra o resultado do algoritmo dentro de um diagrama de Hasse.

A figura 5.8(b) (obtida da convolução do sinal da figura 5.8(a)) mostra um outro sinal 1D que usamos para testar os operadores. As figuras 5.8(c) e 5.8(d) mostram os resultados de aplicar os operadores representados pelas árvores paralela e oblíqua, respectivamente. Para este sinal, o operador oblíquo dá o melhor resultado.

5.3 Exemplo de aplicação com sinais simulados

Nesta seção, demonstramos a performance dos operadores "aperture" para a redução de ruído e desembaçamento. O embaçamento é conseguido por um núcleo plano de convolução e o ruído, gaussiano discreto e independente do sinal, é adicionado a 10% dos pontos.

Relembrando nossa discussão inicial sobre restrições, bastante atenção foi dada para a relação entre o tamanho da janela de níveis de cinza, número de exemplos de treinamento e performance do operador projetado. Em todos os caso, a performance do operador "aperture" foi comparada com

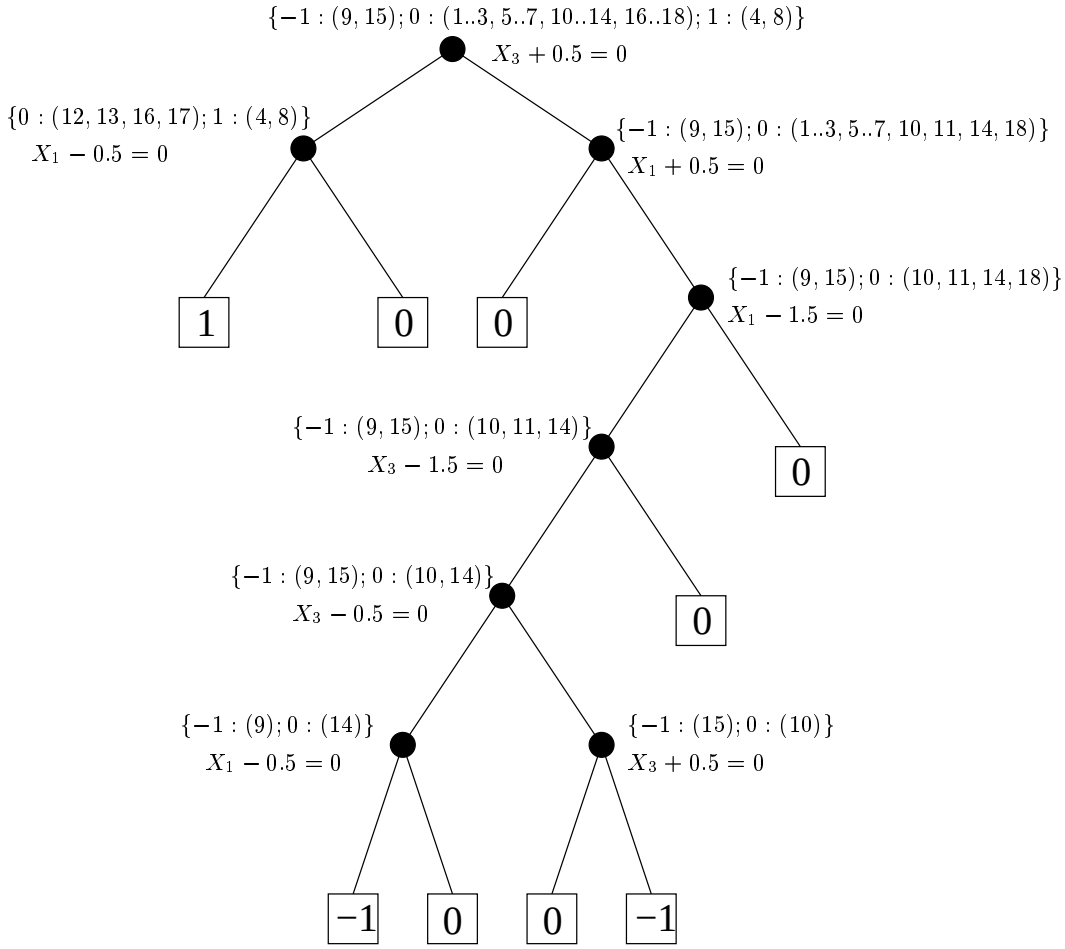


Figura 5.4: Árvore de decisão paralela

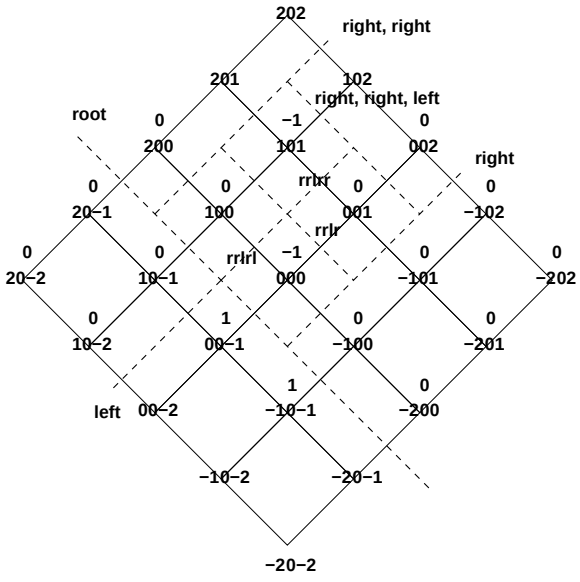


Figura 5.5: Reticulado e árvore paralela

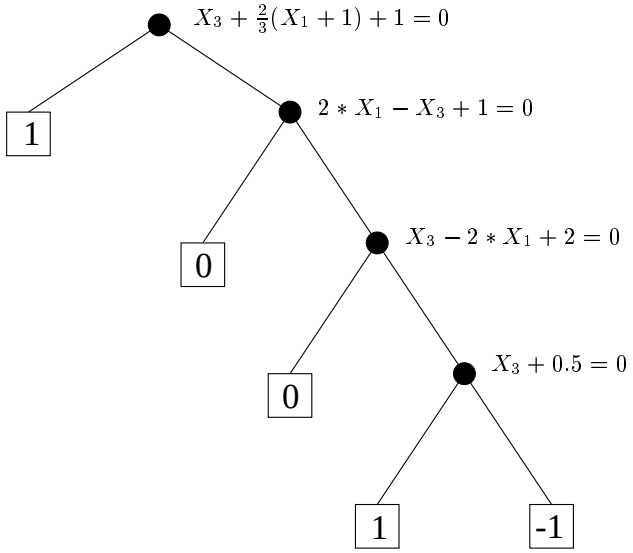


Figura 5.6: Árvore oblíqua

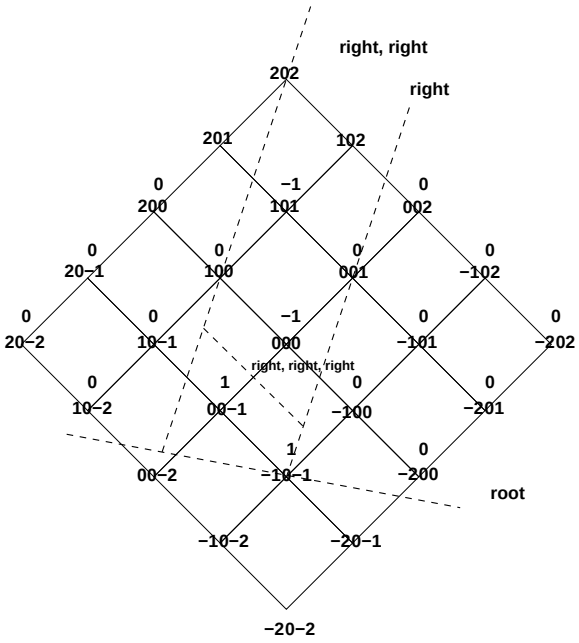


Figura 5.7: Reticulado e árvore oblíqua

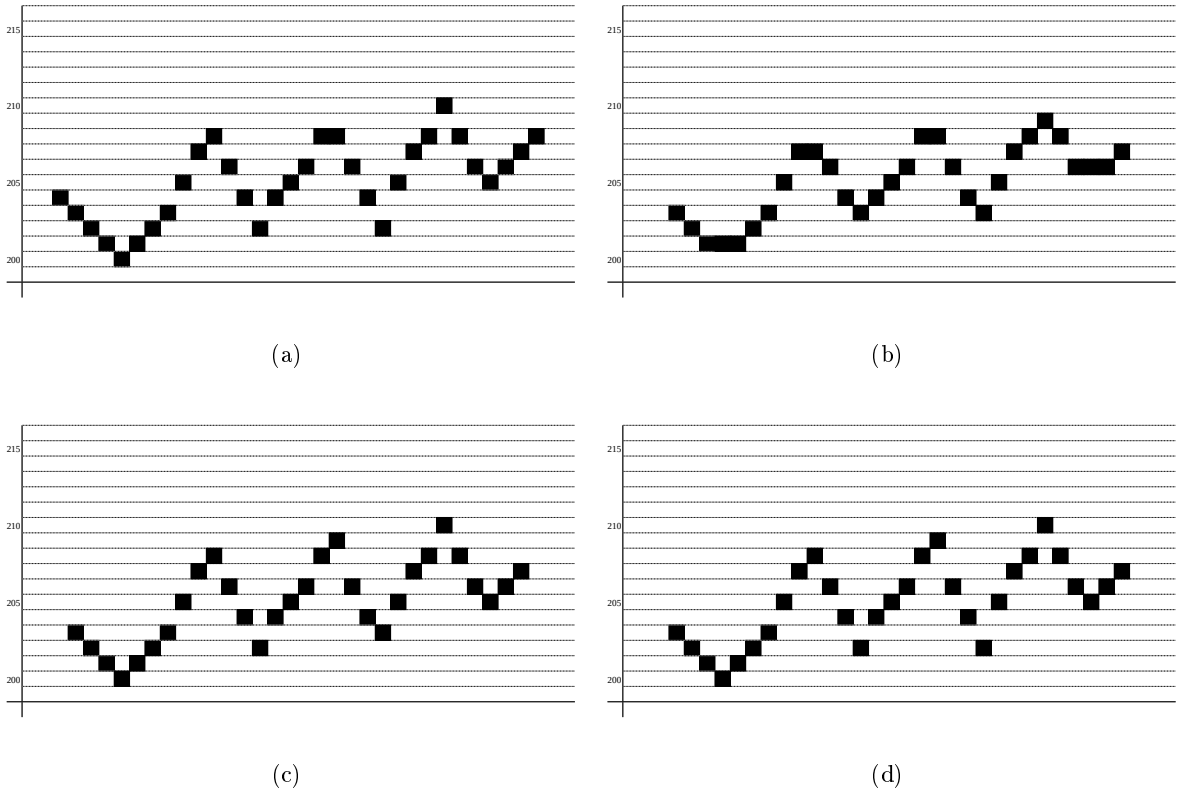
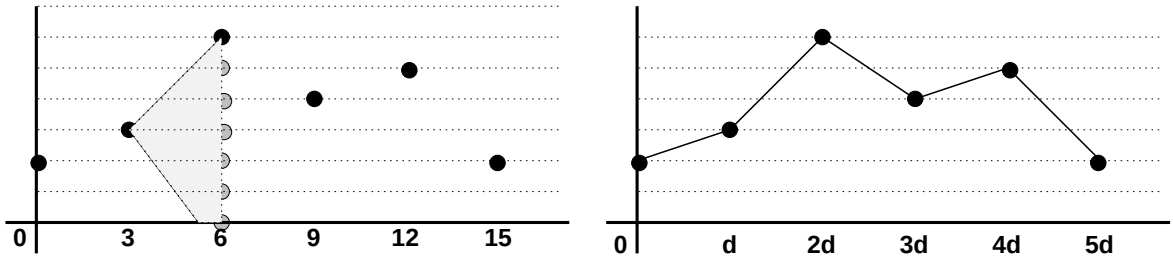


Figura 5.8: Aplicação do operador "aperture": (a) sinal de teste; (b) sinal embaçado; (c) saída do operador representado pela árvore paralela; (d) saída do operador representado pela árvore oblíqua.



(a) (b)

Figura 5.9: Construção da função serra: (a) pontos aleatórios igualmente espaçados; (b) função serra passando pelos pontos aleatórios.

a performance do operador linear de observação finita. Fizemos experimentos sobre diversos tipos de sinais e a performance foi bastante consistente; entretanto, no interesse de manter este trabalho num tamanho razoável, iremos apresentar apenas os resultados para uma certa classe de sinais. Por causa disso, também, iremos restringir a maior parte das análises mostradas para os resultados do MSE (e porquê o usamos como critério de minimização do erro). Porém, fizemos também o estudo para o MAE e os resultados foram semelhantes. A razão principal para usar o MSE em todos os experimentos foi para poder comparar com os filtros lineares ótimos, que minimizam o MSE.

Os resultados experimentais apresentados nesta seção usam sinais do tipo **serra** que são gerados pelo seguinte modelo. Seja $r_0 \in [0, 255]$, um número aleatóriamente escolhido seguindo uma distribuição uniforme. Sejam $r_d, \ldots, r_{md}$, m pontos igualmente distribuidos sobre o domínio do sinal E, (i.e., $d = \lfloor \frac{|E|}{m} \rfloor$ tais que o valor de cada $r_i, i > 0$ é escolhido aleatoriamente (segundo uma distribuição uniforme) segundo a regra: $r_{(i+1)d} \in \{0 \leq r_{id} - d, r_{id} - d + 1, \ldots, r_{id} - 1, r_{id}, r_{id} + 1, \ldots, r_{id} + d - 1, r_{id} + d \leq 255\}$, $\forall i, 0 \leq i < m$ (veja a figura 5.9(a)). Note que d é a distância entre dois pontos aleatórios consecutivos $r_i, r_{(i+1)d}$, e que $r_i \in [0, 255]$, para todos os índices i.

Seja G um conjunto de funções e $g : E \rightarrow [0, 255]$ uma função desse conjunto. Chamamos g de *função serra* (veja a figura 5.9(b)) se g é gerada a partir de R juntando cada dois pontos consecutivos de R pela melhor aproximação de uma reta no espaço discreto, isto é, $g(x_{id}) = r_{id}$, $g(x_{(i+1)d}) = r_{(i+1)d}$, e $g(x), id \leq x \leq (i+1)d$, é dado por,

$$g(x) = \lfloor 0.5 + \frac{(d(i+1) - x)r_{id}}{d} + \frac{(x - id)r_{(i+1)d}}{d} \rfloor \tag{5.5}$$

onde $\lfloor p \rfloor$ significa o menor inteiro que aproxima o valor p.

5.3.1 Desembaçamento

Neste estudo, os operadores "aperture" são projetados para desembaçar sinais serra embaçados por um núcleo de convolução plano de 11 pontos. Nós usamos 20 pares de sinais de treinamento e 10 de teste, cada um com tamanho de 1024 pontos. A figura 5.10 mostra um exemplo de um sinal serra e do respectivo sinal embaçado. A figura 5.11 mostra o gráfico da média do MSE após a aplicação do operador nos sinais de teste, para k e m fixos ($k = m = 7$), em relação ao número de

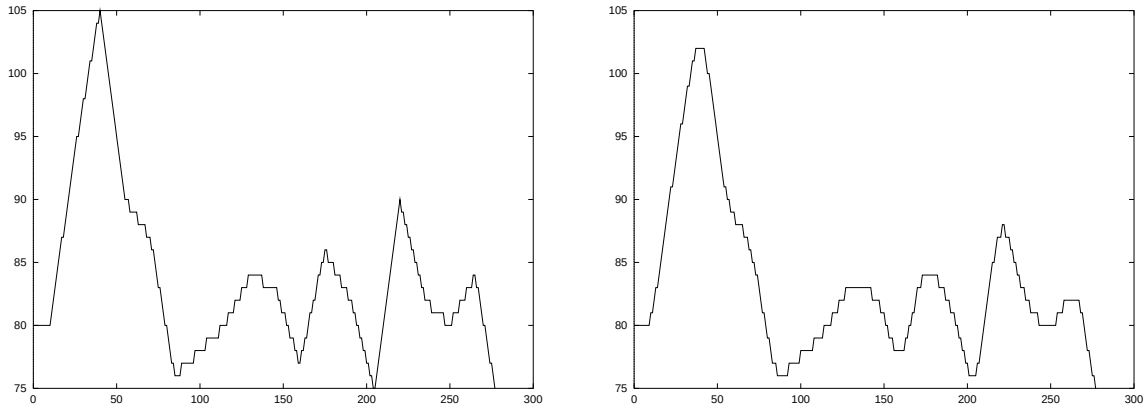


Figura 5.10: Embaçamento: (a) sinal serra; (b) sinal serra embaçado.

exemplos de treinamento. Cada curva é rotulada pelo k da janela do "aperture" e a curva com um valor simples de 23 corresponde ao MSE para o operador linear ótimo projetado sobre uma janela de comprimento 23. Para janelas além desse comprimento, não há melhora discernível dentre os operadores lineares ótimos projetados, que neste caso é uma aproximação finita do operador linear ótimo (filtro de Wiener), comumente usado para desembaçamento.

As curvas do MSE para o operador de janela (9×7) (denotaremos a "aperture" apenas por $W \times k$ quando $k = m$) e para o filtro linear de 23 pontos ficam planas bem cedo porque seus erros de estimação decrescem rapidamente com poucos exemplos de treinamento em relação aos operadores "aperture" de janelas maiores que precisam de mais exemplos para diminuir os erros de estimação. Por volta de 20000 exemplos de treinamento, todos os operadores "aperture" têm MSE mais baixo que os filtros lineares testados. Levando em conta apenas os erros devido a restrição, sabemos que,

$$M\left\langle\Psi_{33\times 7}\right\rangle < M\left\langle\Psi_{27\times 7}\right\rangle < M\left\langle\Psi_{21\times 7}\right\rangle < M\left\langle\Psi_{15\times 7}\right\rangle \tag{5.6}$$

O resultado dos erros experimentais mostram que a situação está invertida para $\mathbf{N} = 1000$ exemplos, isto é,

$$M\left\langle\Psi_{33\times 7,N}\right\rangle > M\left\langle\Psi_{27\times 7,N}\right\rangle > M\left\langle\Psi_{21\times 7,N}\right\rangle > M\left\langle\Psi_{15\times 7,N}\right\rangle \tag{5.7}$$

Para $\mathbf{N} = 20000$ exemplos de treinamento temos,

$$M\left\langle\Psi_{21\times 7,N}\right\rangle < M\left\langle\Psi_{15\times 7,N}\right\rangle < M\left\langle\Psi_{27\times 7,N}\right\rangle < M\left\langle\Psi_{33\times 7,N}\right\rangle \tag{5.8}$$

Isto mostra que 20000 exemplos de treinamento são suficientes para que o operador $\Psi_{21\times 7,N}$ seja melhor que os filtros lineares ótimos testados e que os operadores "aperture" $\Psi_{9\times 7,N}$, $\Psi_{27\times 7,N}$ e $\Psi_{33\times 7,N}$. Observando o método de treinamento proposto aqui, $E[\triangle(\psi_{\mathcal{A},N}, \psi_{\mathcal{A}})] \to 0$ quando $N \to \infty$. Portanto, a equação 5.6 vale no limite para os operadores "aperture" projetados aqui.

A figura 5.12 mostra as curvas do MSE para uma janela espacial fixa de largura 15 pontos. Observando que os operadores projetados $\Psi_{15\times 7}$ e $\Psi_{15\times 6}$ têm erro bastante próximos, nós vemos que as relações entre os filtros projetados é o que deveríamos esperar dos filtros "aperture" ótimos. O

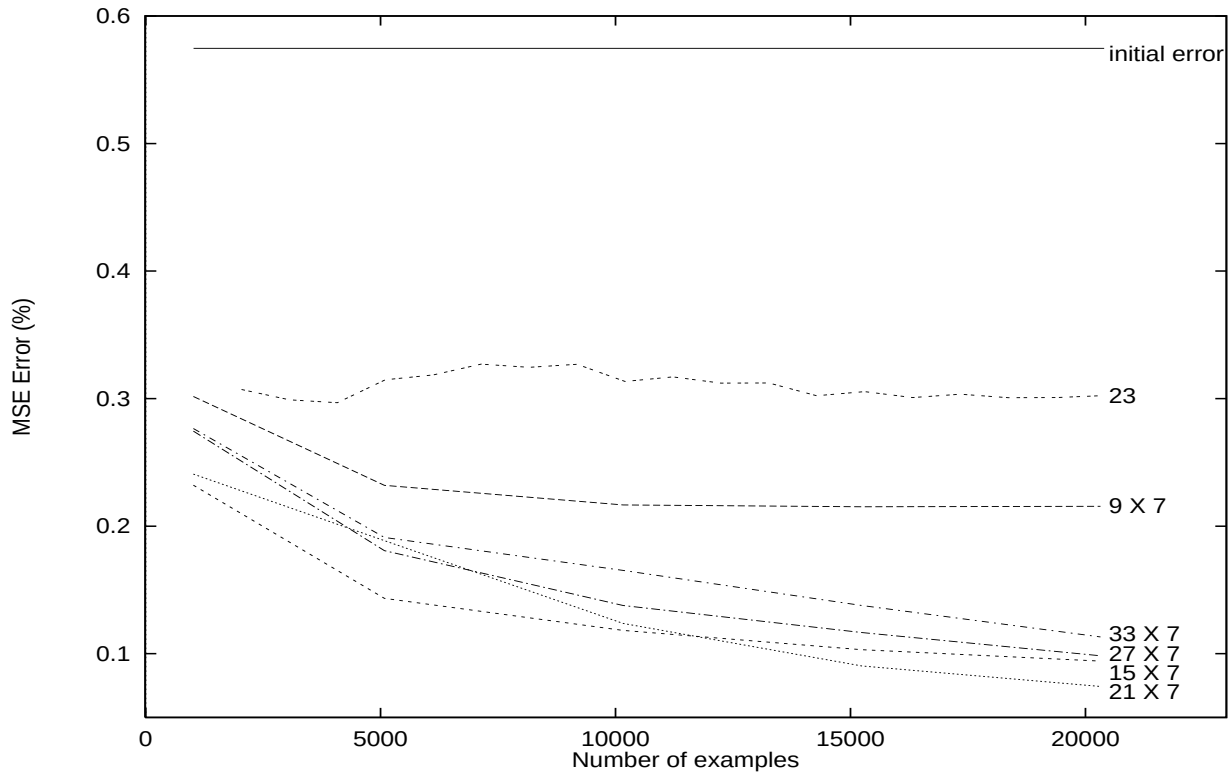


Figura 5.11: Gráfico do MSE para desembaçamento - altura fixada.

operador $\Psi_{15\times 7}$ aparece em ambos os gráficos das figuras 5.11 e 5.12. Comparando essas figuras, vemos que para $\mathbf{N} = 20000$ exemplos,

$$M\left\langle \Psi_{21\times 7,N}\right\rangle < M\left\langle \Psi_{15\times 7,N}\right\rangle < M\left\langle \Psi_{15\times 5,N}\right\rangle < M\left\langle \Psi_{27\times 7,N}\right\rangle < M\left\langle \Psi_{15\times 4,N}\right\rangle \tag{5.9}$$

Estes erros experimentais são bastante próximos e representam erros de um simples treinamento; entretanto, eles são indicativos do comportamento típico observado sobre muitos outros experimentos que deixamos de mostrar aqui.

5.3.2 Desembaçamento e filtragem de ruído gaussiano

Neste experimento, os operadores "aperture" são projetados para filtrar sinais do tipo serra que foram embaçados com um núcleo de convolução plano de 11 pontos e, também, corrompidos por ruído gaussiano (de média zero e variância 5 níveis de cinza) em 10% dos pontos. Foram gerados 75 sinais do tipo serra de comprimento 1024 pontos e seus respectivos sinais corrompidos (ruído + embaçamento). Este conjunto foi dividido em dois subconjuntos, um de 60 sinais para treinamento e um de 15 sinais para teste. A figura 5.13 mostra um exemplo de parte de um dos sinais originais e o respectivo sinal corrompido.

A figura 5.14 mostra os gráficos do MSE para alguns dos operadores projetados em função do número de exemplos de treinamento. Mais uma vez, o operador ótimo linear de 23 pontos estabiliza com muito menos treinamento. Quatro dos cinco operadores projetados cujos MSEs são mostrados no gráfico têm resultado melhor que o filtro linear ótimo em 40000 exemplos de treinamento, mas

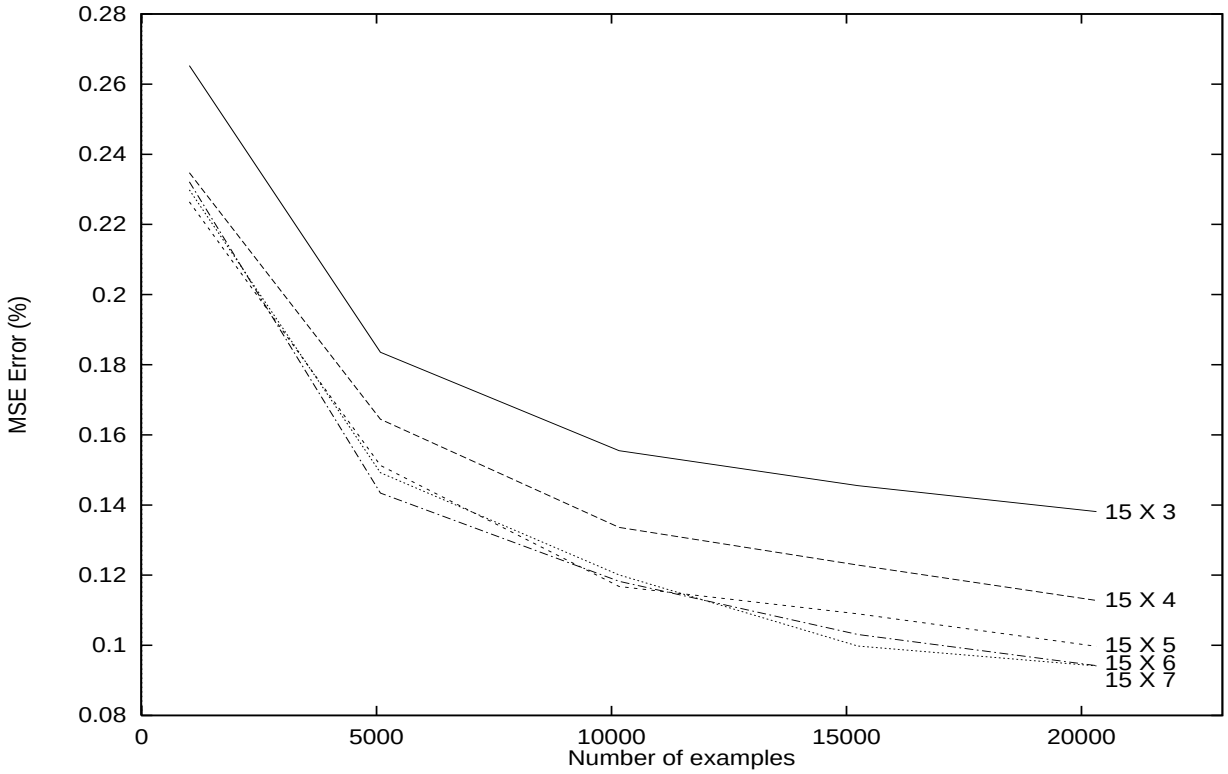


Figura 5.12: Gráfico do MSE para desembaçamento - largura fixada.

nenhum em 30000 exemplos. O efeito da excessiva restrição nos níveis de cinza pode ser visto comparando os resultados dos operadores "aperture" de $21,3,3$ e $21,5,5$.

5.3.3 Filtragem de embaçamento e ruído gaussiano usando operadores iterados de dois estágios

A iteração (composição) de operadores de janela resulta em um operador definido sobre uma janela maior que é a dilatação de todas as janelas que o compõe. Isto é, sejam W_1 e W_2 duas janelas espaciais ($W_1, W_2 \subset E$) e sejam Ψ_1 e Ψ_2 dois operadores de janela binários, ou níveis de cinza, definidos sobre W_1 e W_2, respectivamente. Então, o operador composto $\Psi = \Psi_2\Psi_1$ é definido sobre $W = W_1 \oplus W_2$, isto é, a dilatação de W_1 por W_2. A prova deste fato para os operadores binários encontra-se em [65] ou [126] e é facilmente estendida para operadores de janela em níveis de cinza.

Em termos de projeto estatístico, sabemos que o projeto de um operador definido sobre uma janela grande pode ter erro de estimação maior do que o projetado sobre uma janela pequena (para uma mesma quantidade de exemplos de treinamento). Assim, a idéia do projeto iterado é diminuir o erro de estimação quebrando o operador em dois, ou mais, operadores. O erro de estimação do operador iterado, se essa quebra for bem feita, deve ser menor que o erro do operador projetado sobre composição das janelas que o compõe. Note, no entanto, que o erro do operador composto não é necessariamente o mesmo do operador ótimo de janela maior. Mais formalmente, se $\Psi = \Psi_2\Psi_1$, então $\Delta(\Psi_2\Psi_1, \Psi_{opt}) < \Delta(\Psi, \Psi_{opt})$, onde Ψ_{opt} é o operador ótimo.

A forma de decompor o operador que resulte no operador projetado com menor erro para um certo

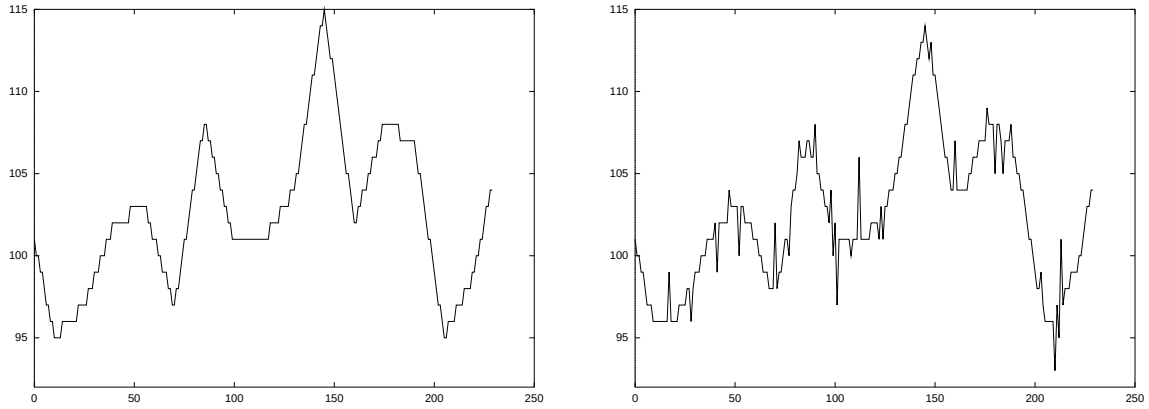


Figura 5.13: Embaçamento com ruído gaussiano aditivo: (a) sinal original; (b) sinal corrompido.

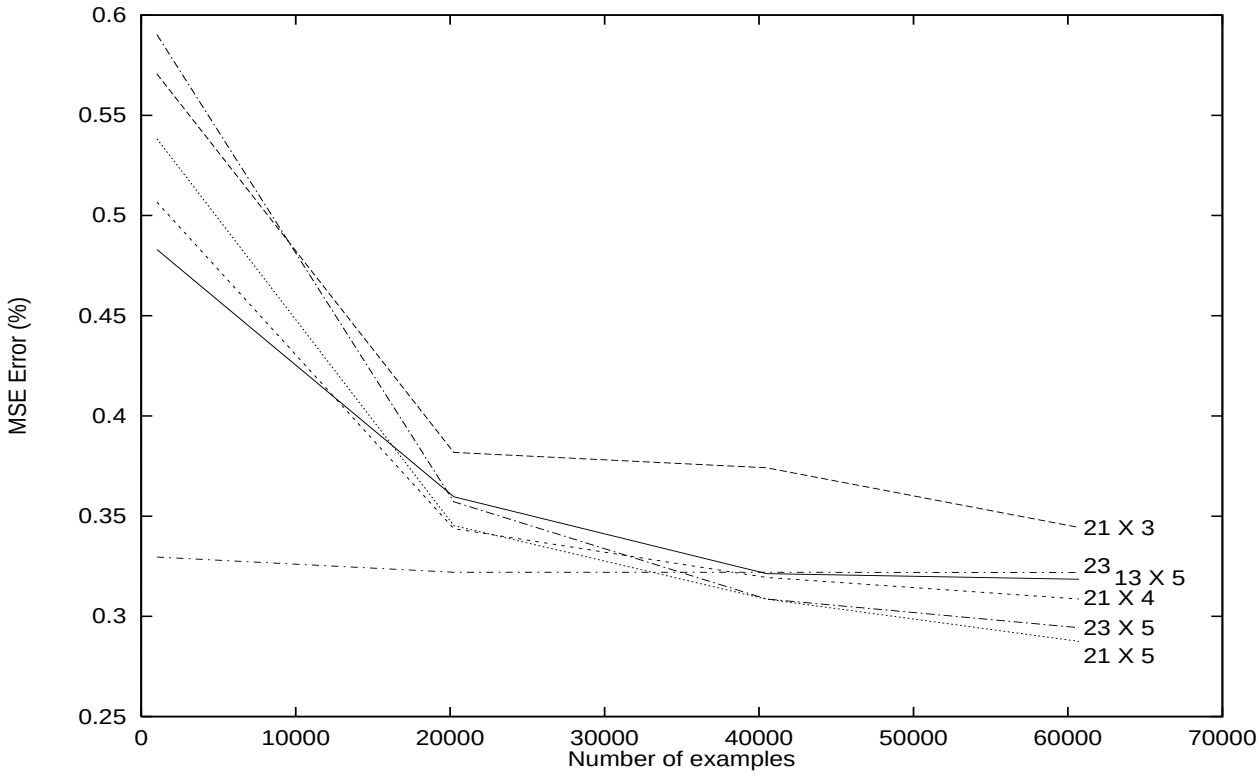


Figura 5.14: Curvas do MSE para a filtragem de embaçamento e ruído.

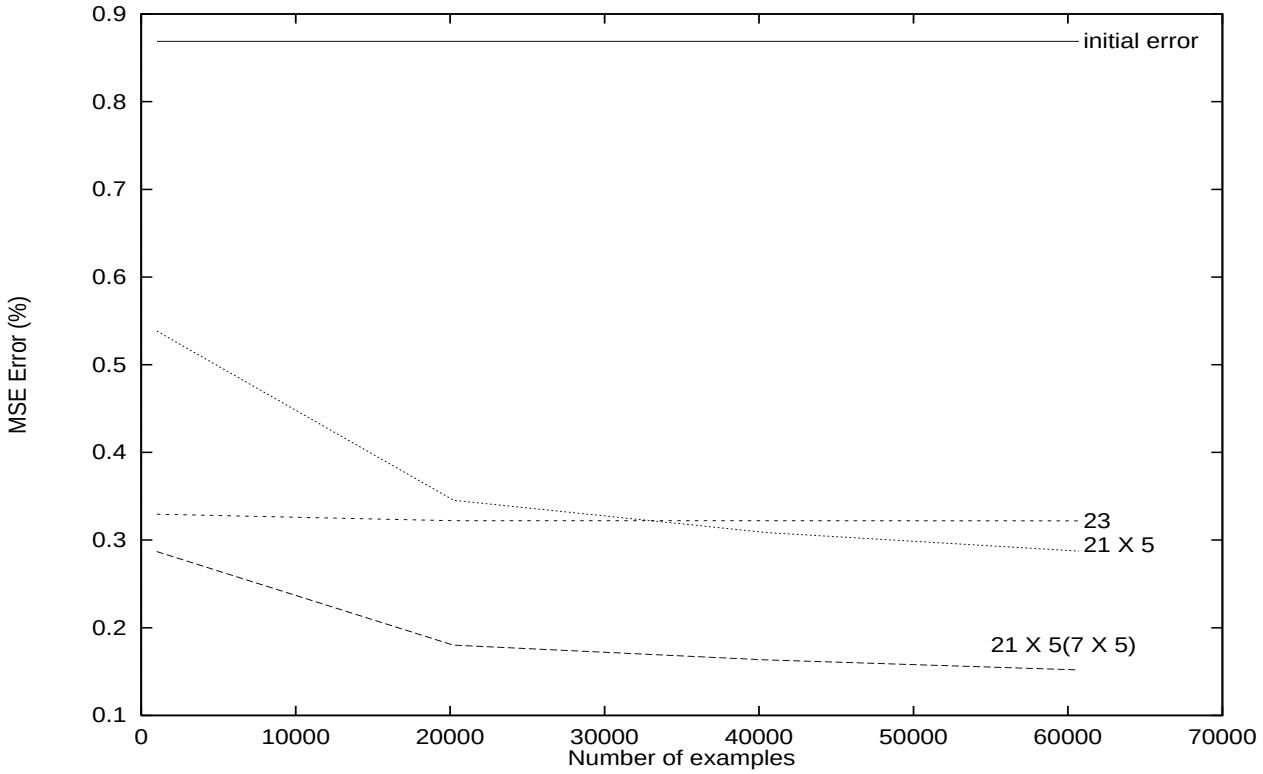


Figura 5.15: Gráficos MSE para os filtros iterados.

conjunto de imagens de treinamento ainda é um problema em aberto. Porém, várias heurísticas já foram testadas. No caso dos operadores iterados de dois estágios, eles são comprovadamente benéficos no projeto de operadores binários [80, 60, 126].

Neste experimento, nós projetamos um operador "aperture" com dois estágios para restaurar os sinais corrompidos por embaçamento e ruído que foram discutidos na seção anterior. Primeiramente, projetamos um operador para filtrar o ruído dos sinais embaçados. A classe de sinais observada consiste de sinais serra que foram embaçados pelo núcleo de convolução plano de 11 pontos e depois adicionado ruído gaussiano de média zero e variância de cinco níveis de cinza. A classe de sinais ideais consiste dos sinais embaçados. Para 60000 exemplos de treinamento, nós achamos que o operador "aperture" $\Phi_{7\times5,N}$, foi o que deu melhor resultado dentre os vários operadores testados. Esse operador foi aplicado em todos os sinais de treinamento e o resultado (os sinais sem ruído) foi usado como classe dos sinais observados para melhor estimar os sinais serra originais. O melhor operador "aperture" projetado, dentre todos os operadores que testamos, foi o operador $\Xi_{21\times5,N}$. O operador projetado composto é o operador "aperture" $\Omega_N = \Xi_{21\times5,N}\Phi_{7\times5,N}$. A figura 5.15 mostra os gráficos MSE para Ω_N, o filtro ótimo linear sobre a janela de 23 pontos, e o operador "aperture" não iterado mostrado na seção anterior, e o operador $\Psi 21 \times 5$. Note a melhora do resultado para o melhor operador "aperture" iterado projetado quando comparado com o melhor operador "aperture" projetado não iterado. Note também que iterar os operadores lineares não ajuda em nada porque iterar o operador ótimo de 23 pontos Λ_1 com um outro operador ótimo linear de 23 pontos Λ_2 para estimar os sinais serra originais produz um operador linear $\Lambda_2\Lambda_1$ sobre uma janela de 45 pontos que não se comporta melhor que o operador ótimo de 45 pontos, que resulta o mesmo que Λ_1 sobre os sinais embaçados e com ruído.

5.4 Exemplo de aplicação com imagens simuladas

Para mostrar o comportamento dos operadores "aperture" em imagens, projetamos esses operadores para desembaçar imagens criadas por um modelo booleano [127] cuja função primária é piramidal. O embaçamento é feito por um núcleo de convolução 3×3 não plano (veja a figura 5.16a). Nós usamos 10 imagens de treinamento e 10 imagens de teste, cada uma com tamanho de 256×256 pontos. A figura 5.17 mostra parte de uma imagem original e a figura 5.18 seu respectivo embaçamento.

As figuras 5.22 e 5.23 mostram os gráficos do MAE e do MSE, respectivamente, para seis dos operadores "aperture" testados ($\Psi_{3\times3,3}$; $\Psi_{3\times3,5}$; $\Psi_{17p,3}$; $\Psi_{17p,5}$; $\Psi_{5\times5,3}$; $\Psi_{5\times5,5}$, onde $p\times q, k = m$ refere-se a uma janela "aperture" e $17p$ refere-se à janela mostrada na figura 5.16b), em função do número de exemplos de treinamento.

O operador "aperture" que dá o menor MAE é o $\Psi_{17p,5}$. Este resultado é similar ao do operador "aperture" $\Psi_{5\times5,5}$. Todos os filtros testados têm resultado MAE melhor que o filtro linear ótimo sobre uma janela de 7×7 pontos. O operador que tem o menor MSE é o operador "aperture" $\Psi_{3\times3,5}$; entretanto, os operadores "aperture" ainda não estão estabilizados com 600000 exemplos e continuariam melhorando com mais treinamento. Realmente, observando a figura 5.22, vemos que os MAEs dos operadores "aperture" não estão estabilizados e, ainda assim, com 600000 exemplos, as janelas maiores já têm resultado melhor do que os operadores com janelas menores.

Em termos de aspecto visual, o operador "aperture" $\Psi_{17p,5}$ tem comportamento melhor que os outros operadores testados. A razão disso é que ele produz uma melhor restauração das arestas, o que é comum para operadores ótimos não-lineares. As figuras 5.19, 5.20 e 5.21 mostram o resultados do operador linear ótimo, do operador "aperture" $\Psi_{3\times3,5}$ e do operador "aperture" $\Psi_{17p,5}$, respectivamente.

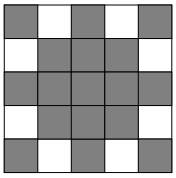
As figuras 5.24, 5.25, 5.26 e 5.27 mostram parte da imagem de teste, o resultado do operador ótimo linear 7×7, o resultado do operador "aperture" $\Psi_{3\times3,5}$, e o resultado do operador "aperture" $\Psi_{17p,5}$, respectivamente.

Estas imagens, e as anteriores, mostram a restauração superior das arestas pelo operador "aperture" $\Psi_{17p,5}$ e sua qualidade visual superior. Esta superioridade visual é ainda mais interessante se levarmos em conta que selecionamos um pedaço da imagem cujo MSE do filtro linear é ligeiramente superior ao MSE do operador "aperture" de $\Psi_{17p,5}$.

O resultado dos operadores diferem do mesmo fragmento da imagem original em 132, 97, e 78 pontos (descontando a borda) para o filtro ótimo linear, para o operador "aperture" $\Psi_{3\times3,5}$, e para o operador "aperture" $\Psi_{17p,5}$, respectivamente. Os resultados diferem do fragmento da imagem original em mais que um nível de cinza em 28, 17, e 11 pontos para os operadores "aperture" $\Psi_{17p,5}$ e $\Psi_{3\times3,5}$, e para o filtro linear, respectivamente. Isto explica o fato do MSE do operador "aperture" $\Psi_{17p,5}$ não ser tão bom para este fragmento. Isto é, embora ele tenha acertado num número menor de pontos, quando errou, errou por uma quantidade de níveis de cinza maior.

1	2	1
2	3	2
1	2	1

(a)



(b)

Figura 5.16: Núcleo de convolução e janela de 17 pontos.

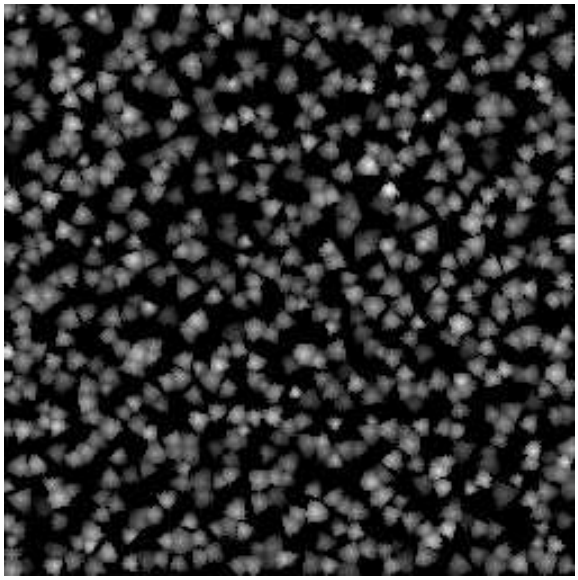


Figura 5.17: Modelo booleano: imagem original.

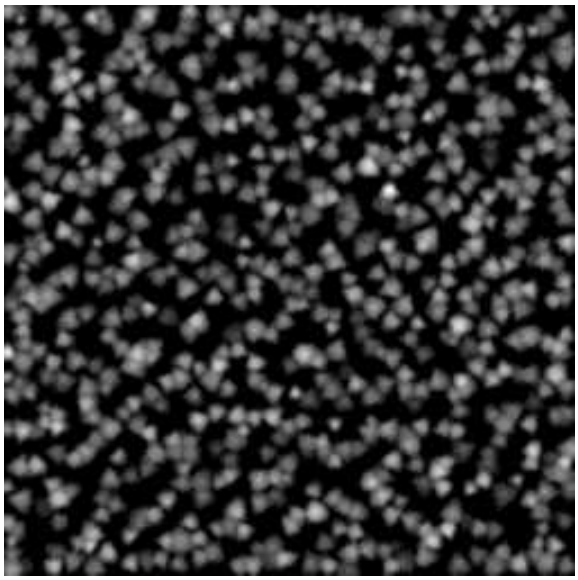


Figura 5.18: Modelo booleano: imagem embaçadas.

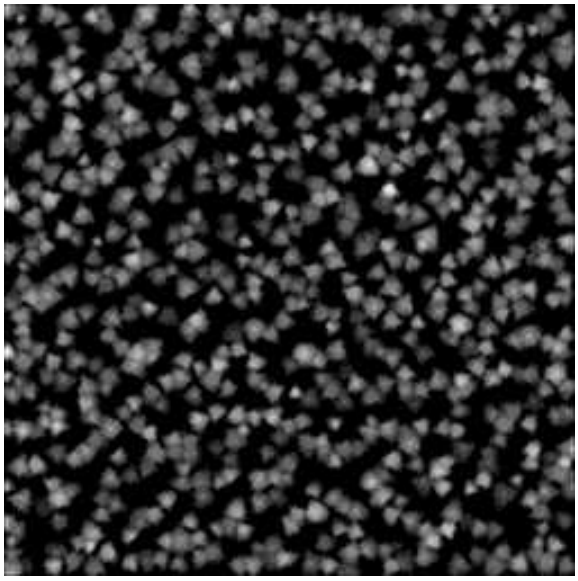


Figura 5.19: Resultado do operador linear sobre uma janela 7×7.

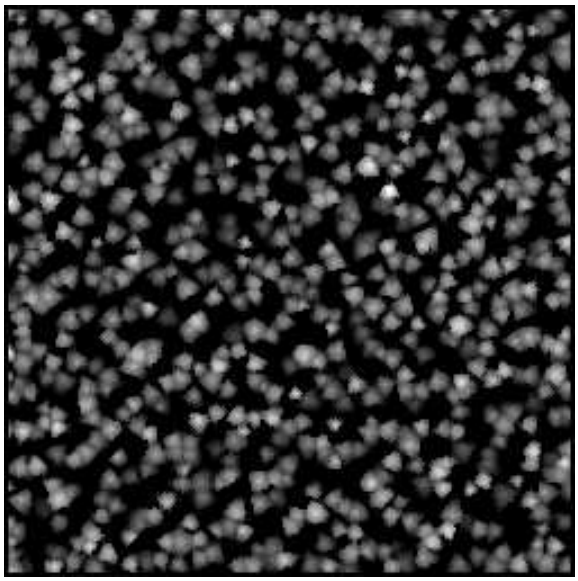


Figura 5.20: Resultado do operador "aperture" $\Psi_{3\times 3,5}$.

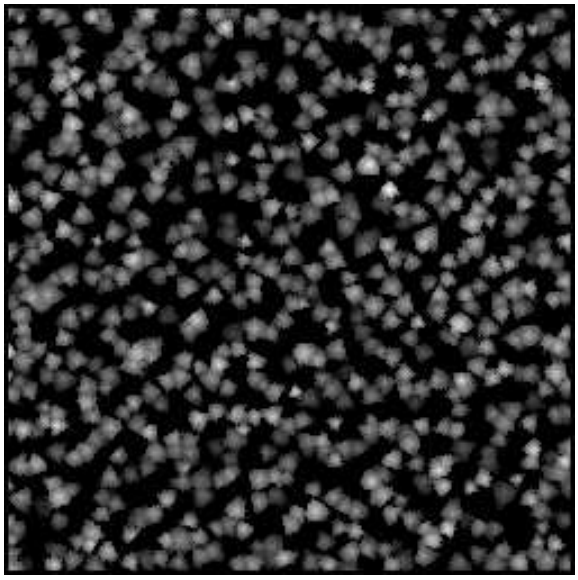


Figura 5.21: Resultado do operador "aperture" $\Psi_{17p,5}$.

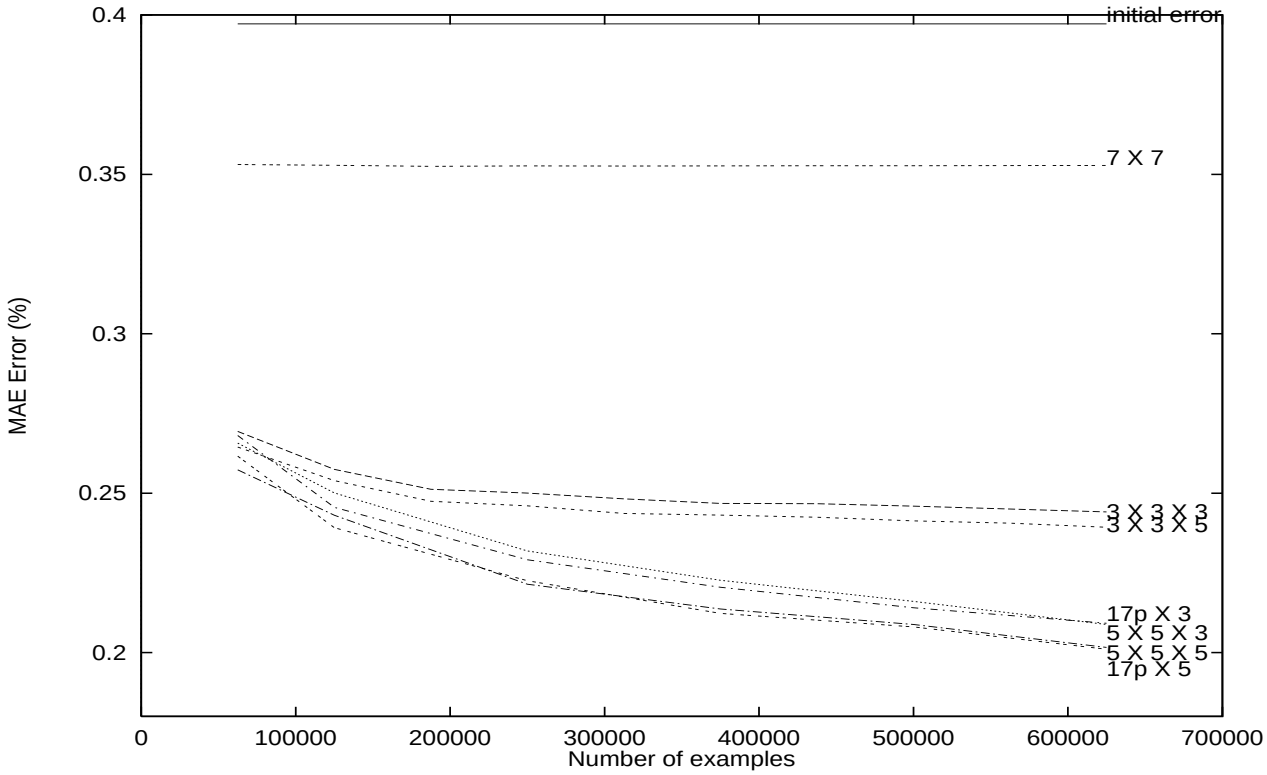


Figura 5.22: Curvas do MAE para os experimentos com as imagens do modelo booleano.

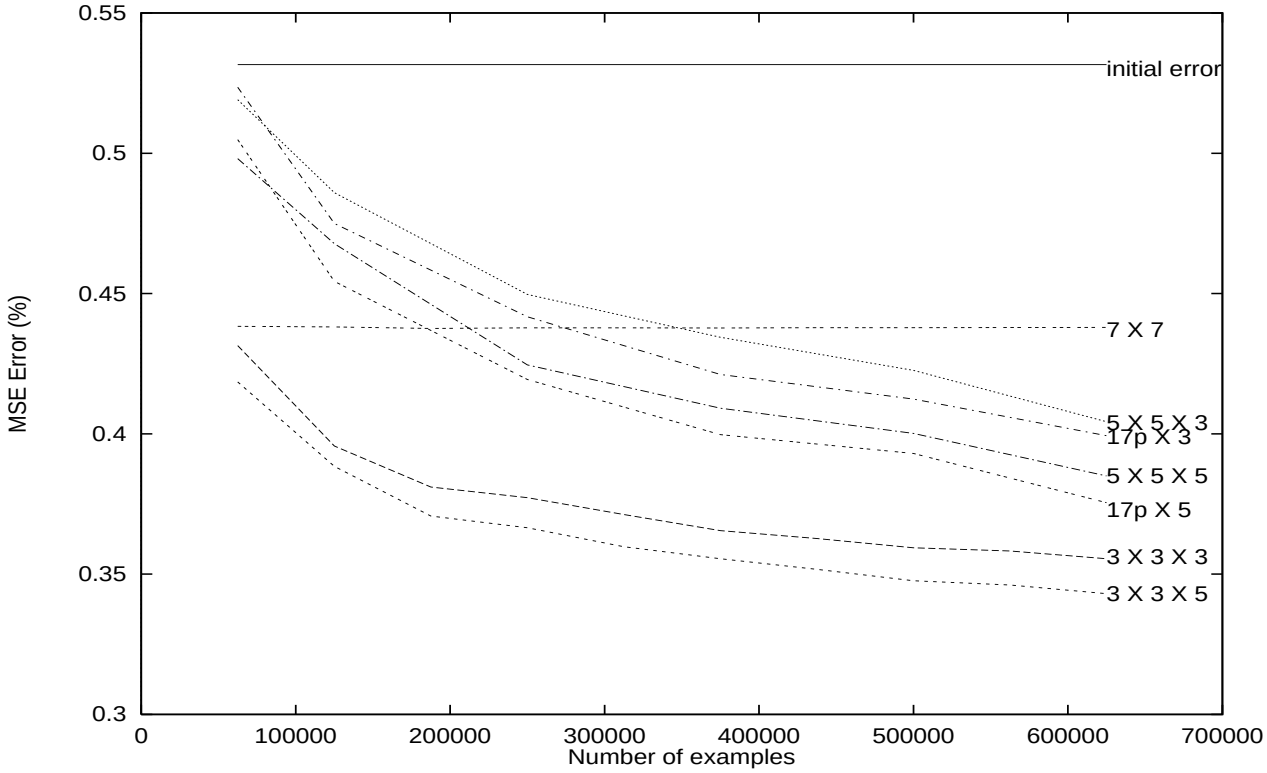


Figura 5.23: Curvas do MSE para os experimentos com as imagens do modelo booleano.

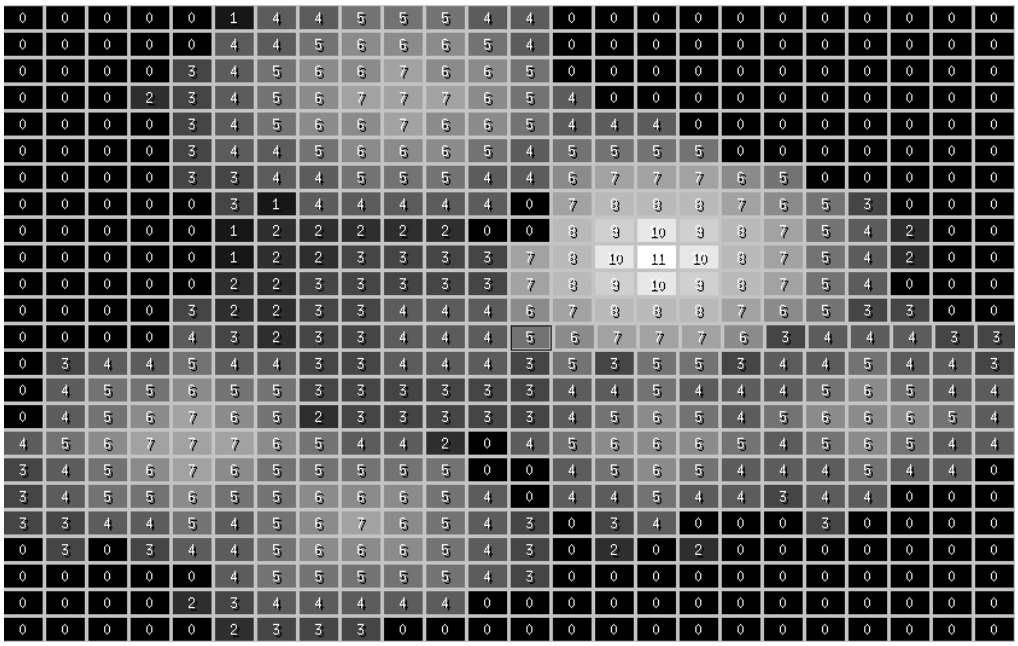


Figura 5.24: Parte de uma imagem do modelo booleano.

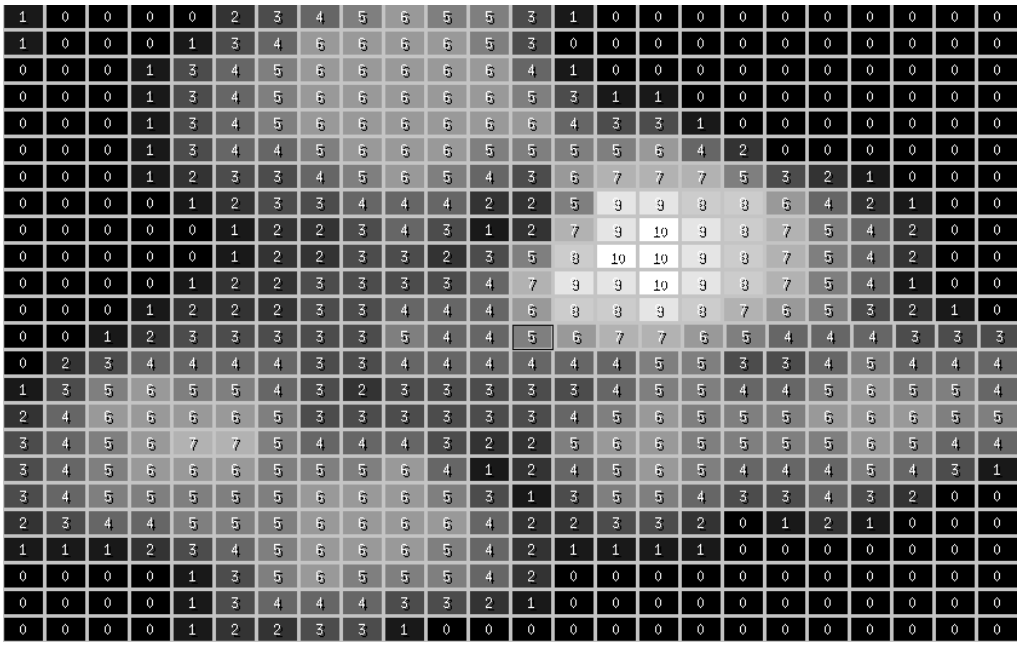


Figura 5.25: Resultado do filtro ótimo linear aplicado à imagem 5.24.

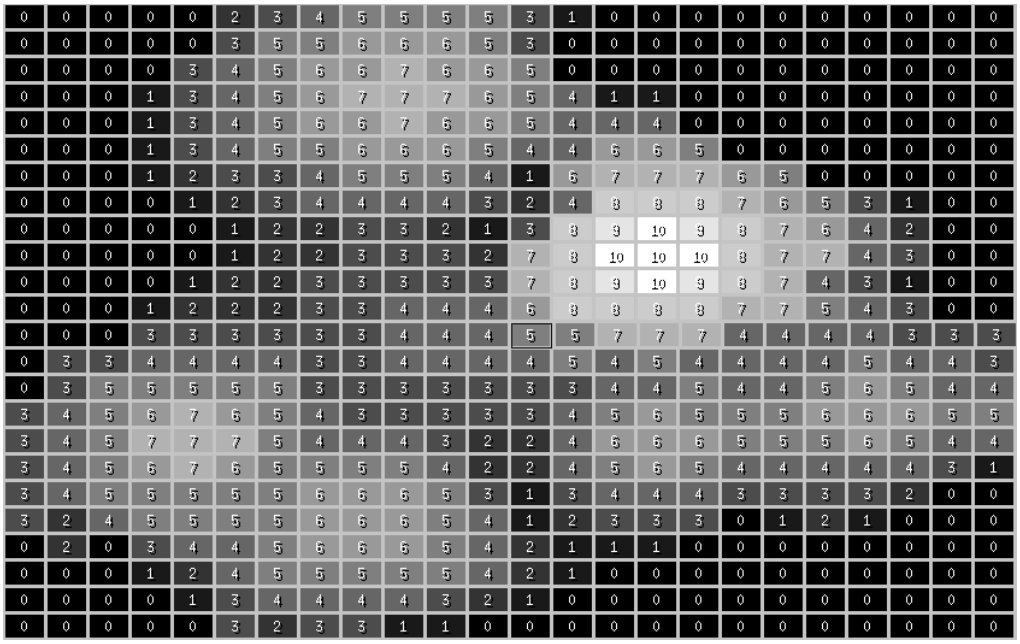


Figura 5.26: Resultado do operador "aperture" $\Psi_{3\times3,5}$ aplicado à imagem 5.24.

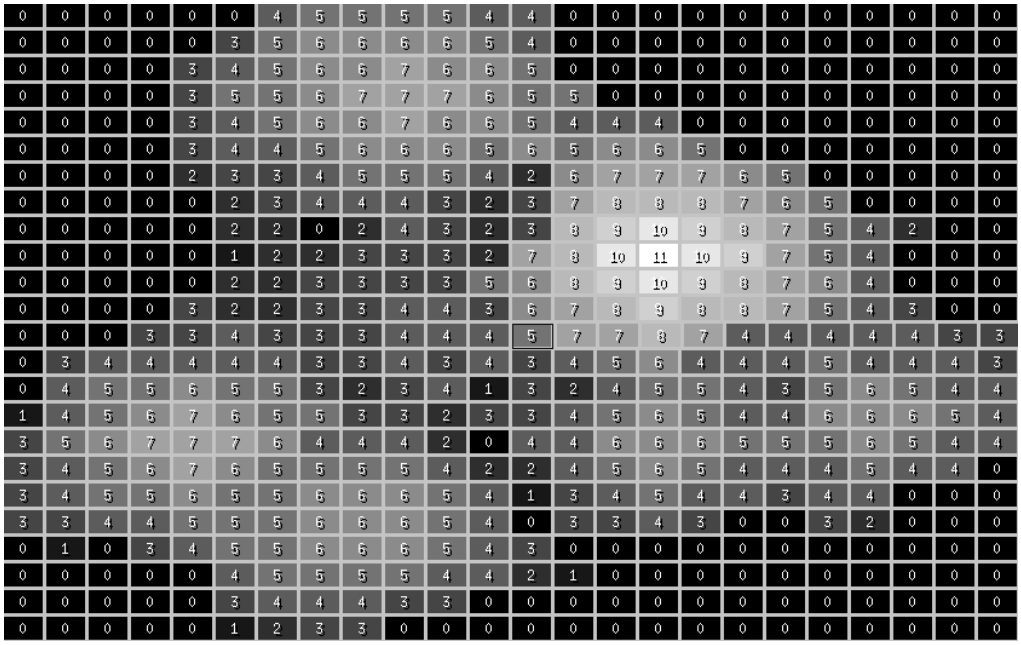


Figura 5.27: Resultado do operador "aperture" $\Psi_{17p,5}$ aplicado à imagem 5.24.

Capítulo 6

Representação e indução via algoritmo ISI

Conforme vimos, o projeto estatístico de operadores consiste de três etapas básicas, quais sejam, (1) estimação das probabilidades condicionais, (2) decisão, ou, escolha de um rótulo a ser associado para cada configuração observada em (1), e (3) indução e simplificação de representação do operador projetado.

Simplificar a representação do operador e generalizar a classificação das configurações não observadas a partir da tabela produzida pela etapa de decisão é uma das principais tarefas do projeto de operadores. No capítulo anterior vimos uma forma de fazer isso através das árvores de decisão, que além de simplificar a representação, induzem uma classificação para as configurações não observadas. Aquela técnica, porém, tem desvantagens em relação à técnica que mostraremos neste capítulo. A primeira desvantagem é que não é simples fazer a manipulação algébrica dos operadores usando a representação ODT, por exemplo, dados dois operadores representados por ODT, como fazer para construir um operador que é a união desses dois? A segunda desvantagem é que não é simples fazer aprendizado reforçado [7, 128, 116], isto é, dado um operador representado por uma ODT, como "consertar" o operador dando mais alguns exemplos?

O nome ISI vem do inglês "Incremental Splitting of Intervals" e está associado à idéia aplicada no algoritmo [73, 118] para computar, a partir da tabela de decisão $\mathcal{M}_{\mathcal{A}}$, o conjunto de intervalos do reticulado que representa a base do operador. Originalmente escrito para simplificar funções com entradas e saídas binárias (ou seja, funções do tipo $f : \{0,1\}^n \to \{0,1\}$), ele foi estendido para simplificar funções com entrada binária e saída multivalorada [12] (ou seja, funções do tipo $f : \{0,1\}^n \to \{0,1,\ldots,k\}$) e, mais recentemente, para simplificar a representação de funções em níveis de cinza com saída em níveis de cinza. Estas extensões são denominadas, respectivamente, de *ISI multiclassificação* e *ISI níveis de cinza*.

Por questões didáticas, apresentaremos primeiramente o ISI multiclassificação [12]. Em seguida apresentaremos nossa extensão do algoritmo para níveis de cinza.

6.1 ISI multiclassificação

O ISI multiclassificação é uma extensão do algoritmo ISI binário [73, 118] utilizado para simplificar a representação de classificadores de formas geométricas. Ele foi desenvolvido para auxiliar o projeto de operadores para classificação de caracteres em imagens binárias e para simplificar o projeto de um OCR (do inglês "Optical Character Recognition"[1]) morfológico [12, 129, 130].

Uma vez que o ISI multiclassificação está relacionado com operadores que se aplicam a imagens binárias, recordamos brevemente alguns conceitos relativos a elas. Imagens binárias são equivalentemente representadas por subconjuntos do domínio. Nesta seção adotaremos a representação de imagens binárias por subconjuntos de E, isto é, $S \subseteq E$.

Um operador de classificação pode ser modelado por um mapeamento do tipo $\Psi : \mathcal{P}(E) \rightarrow \{0, 1, \ldots, k\}^E$, onde $\mathcal{P}(E)$ representa o conjunto de imagens binárias definidas sobre E e $\{0, 1, \ldots, k\}^E$ representa o conjunto de imagens "classificadas". Um operador de classificação que seja invariante por translação e localmente definido por uma janela $W \subseteq E$ pode ser caracterizado por uma função do tipo $\psi : \mathcal{P}(W) \rightarrow \{0, 1, \ldots, k\}$, onde k representa o número de possíveis classes, da seguinte forma:

$$\Psi(S)(t) = \psi(S_{-t} \cap W), \forall t \in E, \forall S \subseteq E\,. \tag{6.1}$$

O *núcleo de ψ no nível i* é o conjunto $\mathcal{K}_i(\psi)$ dado por, para qualquer $X \in \mathcal{P}(W)$,

$$X \in K_i(\psi) \Longleftrightarrow \psi(X) \geq i. \tag{6.2}$$

Note que $\mathcal{K}_k(\psi) \subseteq \mathcal{K}_{k-1}(\psi) \subseteq \cdots \subseteq \mathcal{K}_1(\psi) \subseteq \mathcal{K}_0(\psi)$.

O mapeamento ψ tem a seguinte sup-decomposição [68]:

$$\psi(X) = max\{i : X \in \mathcal{K}_i(\psi)\}, \forall X \in \mathcal{P}(W). \tag{6.3}$$

A *base de ψ no nível i*, $\mathbf{B}_i(\psi)$, é o conjunto dos intervalos maximais de $\mathcal{K}_i(\psi)$. Em termos de base, a sup-decomposição pode ser escrita [68] como

$$\psi(X) = max\{i : X \in [A, B], [A, B] \in \mathbf{B}_i(\psi)\}, \forall X \in \mathcal{P}(W). \tag{6.4}$$

O ISI multiclassificação calcula incrementalmente cada uma das bases $B_i(\psi)$, do nível 1 ao nível k, aplicando o ISI binário repetidas vezes. É importante notar que os rótulos de classificação não precisam necessariamente ser seqüenciais no intervalo $[0, k]$. Para simplificar o entendimento do algoritmo, vamos supor a existência de um mapeamento dos rótulos no conjunto $\{0, 1, \ldots, k\}$ e que o valor 0 representa os valores do fundo da imagem.

Enquanto árvores de decisão geram uma partição do reticulado com um rótulo associado a cada uma das partes, o ISI multiclassificação gera uma coleção de intervalos, tal que os intervalos da subcoleção associada ao rótulo $i+1$ são subintervalos contidos nos intervalos da subcoleção associada ao rótulo i.

A operação fundamental do ISI binário é a quebra sucessiva de intervalos. Inicialmente recordamos esta operação e em seguida apresentamos o algoritmo ISI multiclassificação que também é baseada nesta operação. O algoritmo ISI binário, juntamente com heurísticas para melhorar seu desempenho, encontram-se descritas em [73, 118].

[1] Um programa que transforma a imagem de um texto digitalizado em um arquivo ASCII.

6.1.1 Quebra de um intervalo

Uma das partes importantes do algoritmo é computar a subtração de um intervalo de outro ou, como foi chamado originalmente, a quebra (do inglês "split") de um intervalo [73, 118].

Definição 6.1 *Seja* $[A, B] \subset [\emptyset, W]$ *um intervalo qualquer do reticulado das partes de* W*. Seja* $[C, D]$ *um subintervalo cuja intersecção com* $[A, B]$ *é não-vazia. A* diferença *entre os intervalos* $[A, B]$ *e* $[C, D]$*, é dada por,*

$$[A, B] \setminus [C, D] = \{X \in [\emptyset, W] : X \in [A, B] \text{ e } X \notin [C, D]\}. \tag{6.5}$$

A proposição a seguir [131, 73] apresenta a diferença entre dois intervalos descrita em termos de intervalos maximais $\mathbf{M}([A, B] \setminus [C, D])$ contidos na diferença $[A, B] \setminus [C, D]$.

Proposição 6.2 *Sejam* $[A, B]$ *e* $[C, D]$ *dois intervalos de* $[\emptyset, W]$*. Então,*

$$\mathbf{M}([A, B] \setminus [C, D]) = \{[A, B \cap \{c\}^c] : c \in C \cap A^c\} \cup \{[A \cup \{d\}, B] : d \in D^c \cap B\}, \tag{6.6}$$

onde o complemento $\cdot^c$ *é em relação ao conjunto* W*.*

No caso em que $C = D = X$, onde $X \in \mathcal{P}(W)$, a fórmula simplifica-se para,

$$\mathbf{M}([A, B] \setminus \{X\}) = \{[A, B \cap \{a\}^c] : a \in X \cap A^c\} \cup \{[A \cup \{b\}, B] : b \in B \cap X^c\}. \tag{6.7}$$

A *dimensão* de um intervalo $[A, B] \subseteq \mathcal{P}(W)$ é denotada $dim([A, B])$ e é dada pela cardinalidade do conjunto $B \setminus A$. Dois resultados interessantes que também estão provados em [73] é que:

- Todos os intervalos em $\mathbf{M}([A, B] \setminus [C, D])$ têm dimensão $dim([A, B]) - 1$.
- A quantidade de intervalos em $\mathbf{M}([A, B] \setminus [C, D])$ é $dim([A, B]) - dim([A, B] \cap [C, D])$.

6.1.2 Algoritmo para a construção da base

O algoritmo ISI binário consiste basicamente em aplicar o processo de quebra de intervalos, eliminando do reticulado $(\mathcal{P}(W), \subseteq)$ todos os elementos X tais que $\psi(X) = 0$. Assim, após a eliminação de todos eles, garante-se que os intervalos restantes cobrem elementos X tais que $\psi(X) = 1$, constituindo uma coleção de intervalos que pode ser interpretada como a base de um operador.

Uma base, ou seja, a coleção de intervalos maximais resultantes das extrações, pode conter algum intervalo redundante, isto é, pode conter um intervalo $[A, B]$ tal que todos os seus elementos estão contidos em algum outro intervalo da coleção. Por isso, o algoritmo ISI engloba uma segunda fase que consiste no cálculo de uma cobertura mínima [132]. Este cálculo seleciona uma menor subcoleção de intervalos suficiente para cobrir todos os elementos X tais que $\psi(X) = 1$.

Seja $(\mathcal{P}(W), \subseteq)$ o reticulado das partes de W, onde W é um conjunto de 3 elementos. Os elementos de $\mathcal{P}(W)$ são denotados por 000, 001, 010, 100, 011, 101, 110 e 111. A figura 6.1a mostra o reticulado $(\mathcal{P}(W), \subseteq)$ e três intervalos (marcados por elipses tracejadas) que cobrem os elementos X tais que $\psi(X) = 1$ (os elementos representados por círculos preenchidos). Porém, um dos intervalos pode

a b

Figura 6.1: Exemplo de cobertura mínima.

ser eliminado pois os dois outros intervalos vão continuar cobrindo todos os elementos X tais que $\psi(X) = 1$, como mostrado na figura 6.1b.

Como já vimos, em geral as funções processadas pelo algoritmo ISI não são completamente definidas. Assim, intervalos que não contém nenhum elemento X tal que $\psi(X) = 1$ podem ser eliminados durante a fase de extração, uma vez que eles não serão necessários na fase de cálculo da cobertura mínima. Na figura 6.2, círculos preenchidos são os pontos para os quais ψ vale 1, os não preenchidos são pontos para os quais ψ vale 0, e a ausência de círculo indica um ponto para o qual ψ não está definido. A figura 6.2a apresenta uma situação em que o elemento 001 é eliminado do intervalo $[000, 111]$. Desta operação, resultam três intervalos, a saber, $[000, 110]$, $[100, 111]$ e $[010, 111]$. Este último, o intervalo superior do reticulado (marcado com um losângulo tracejado), não contém nenhum elemento X tal que $\psi(X) = 1$. Portanto, ele pode ser descartado, como mostrado na figura 6.2b.

Figura 6.2: a) Os três intervalos após a extração de 001. b) Eliminação do intervalo $[010, 111]$ que não será necessário na cobertura mínima.

O algoritmo ISI multiclassificação consiste em aplicar repetidas vezes o ISI binário. Inicialmente todos os exemplos com rótulo 0 são eliminados do reticulado $(\mathcal{P}(W), \subseteq)$. Os intervalos resultantes formam a base no nível 1, significando que estes intervalos correspondem aos exemplos de rótulo maior ou igual a 1. Em seguida, são eliminados todos os exemplos de rótulo 1, a partir dos intervalos anteriores. Os intervalos resultantes formam a base no nível 2 e são contidos nos intervalos do nível 1. Este processo é repetido até todos os exemplos de rótulo $k - 1$ serem eliminados, resultando na base no nível k.

O algoritmo ISI multiclassificação tem como entradas:

- Um conjunto de intervalos iniciais do reticulado, $\mathbb{I}_{\text{init}}$, rotulados de acordo com o conhecimento prévio do usuário, ou de acordo com algum projeto anterior. Usualmente o conjunto inicial de intervalos é $\{[\emptyset, W]\}$, com o rótulo 0 como "default".

- Um parâmetro kex para controlar de quantos em quantos passos deve-se aplicar o procedimento para cálculo da cobertura mínima (isto é, um subconjunto dos intervalos que é suficiente para representar a função). Este procedimento é uma heurística para manter o número de intervalos relativamente pequeno durante todo o processo de extração.
- Um conjunto $\mathcal{M}$ de exemplos rotulados.

A saída do algoritmo é uma coleção de intervalos, mais precisamente, uma lista linear dos intervalos em $\mathbf{B}_k(\psi)$, $\mathbf{B}_{k-1}(\psi)$, ..., $\mathbf{B}_1(\psi)$.

Algorithm 2 Algoritmo ISI para encontrar a base de um W-operador Ψ binário para multiclassificação.

$\mathbb{I} \leftarrow \mathbb{I}_{\text{init}}$; $\mathbb{I}_{\text{final}} \leftarrow \emptyset$;
procex $\leftarrow 0$;
$(\mathcal{M},\mathcal{L}) = \text{sort}(\mathcal{M},0,\text{increasing})$;
$\mathcal{M}_1 = \mathcal{M}$;
for all $l \in \mathcal{L}$ (a menos do último) **do**
  $\mathcal{M}_0 = \{(X,l) : (X,l) \in \mathcal{M}\}$;
  $\mathcal{M}_1 = \mathcal{M}_1 \setminus \mathcal{M}_0$;
  for all $(X,l) \in \mathcal{M}_0$ **do**
    $\mathbb{I}_{\text{New}} \leftarrow \emptyset$; procex $\leftarrow$ procex $+ 1$;
    $\mathbb{I}_{\text{P}} \leftarrow \{[A,B] \in \mathbb{I} : X \notin [A,B]\}$;
    $\mathbb{I}_{\text{tmp}} \leftarrow \{[A,B] \in \mathbb{I} : X \in [A,B]\}$;
    for all $[A,B] \in \mathbb{I}_{\text{tmp}}$ **do**
      $\mathbb{I}_{\text{split}} \leftarrow \mathbf{M}([A,B] \setminus \{X\})$;
      for all $[A',B'] \in \mathbb{I}_{\text{split}}$ **do**
        if $\exists X \in \mathcal{M}_1$ such that $X \in [A',B']$ **then**
          if $[A',B'] \not\subset [E,F], \forall [E,F] \in \mathbb{I}_{\text{P}}$ **then**
            $\mathbb{I}_{\text{New}} \leftarrow \mathbb{I}_{\text{New}} \cup \{[A',B']\}$;
          end if
        end if
      end for
    end for
    $\mathbb{I} \leftarrow \mathbb{I}_{\text{P}} \cup \mathbb{I}_{\text{New}}$;
    if (procex $= kex$) or (this is the last iteration) **then**
      $\mathbb{I} \leftarrow min\_cover(\mathcal{M}_1, \mathbb{I})$;
      procex $\leftarrow 0$;
    end if
  end for
  Rotule todos os intervalos em $\mathbb{I}$ com o próximo rótulo ;
  $\mathbb{I}_{\text{final}} = \mathbb{I}_{\text{final}} \cup \mathbb{I}$;
end for
Return $\mathbb{I}_{\text{final}}$;

Inicialmente, o conjunto de rótulos é ordenado (a menos do rótulo 0, que será o primeiro rótulo a ser processado) em ordem crescente de freqüência, isto é, se o rótulo l_1 aparece associado a mais exemplos que o rótulo l_2, então os exemplos com rótulo l_1 vão ser processados depois dos exemplos

com rótulo l_2.

Explicamos a seguir as partes do algoritmo 2.

1. Linhas 5 a 30 (Laço principal) - começando pelo rótulo 0 e depois os outros na ordem computada por `sort(`$\mathcal{M}$`,0,increasing)`, os exemplos são extraídos seqüencialmente. Os exemplos com o rótulo mais freqüente, isto é, o último rótulo da ordem, não são processados.

2. Linhas 8 a 27 - Laço para processar cada exemplo do conjunto $\mathcal{M}$ cujo rótulo é l. Esta parte corresponde ao algoritmo ISI binário, onde os elementos de rótulo l são os exemplos a serem extraídos, enquanto os exemplos em $\mathcal{M}_1$ são os exemplos que devem ser cobertos pelos intervalos resultantes da extração.

3. Linha 9 - O conjunto que vai guardar os intervalos novos resultantes da extração de exemplos com rótulo l é inicializado com vazio. O contador de exemplos processados `procex` é incrementado.

4. Linhas 10 e 11 - O conjunto dos intervalos, $\mathbb{I}$, é dividido nos intervalos que contém o exemplo que está sendo processado e nos que não.

5. Linhas 12 a 21 - O exemplo que está sendo processado tem que ser retirado do conjunto dos intervalos que o contém. Isso vai ser feito intervalo por intervalo. Os intervalos novos que são gerados podem, ou não, estar contidos nos intervalos em $\mathbb{I}_\mathrm{P}$. Os intervalos que estão contidos são descartados pois já estão representados em $\mathbb{I}_\mathrm{P}$.

   - Linha 13 - O intervalo que está sendo processado é quebrado de acordo com a fórmula 6.7 e é colocado no conjunto $\mathbb{I}_{\mathrm{split}}$.
   - Linhas 14 a 20 - Laço para verificar se cada intervalo novo que foi criado será mantido. A ordem das condições abaixo é importante para o desempenho do algoritmo.
   - Linha 15 - Verifica se o intervalo que está sendo processado cobre alguma configuração observada. Em verdade, o algoritmo verifica apenas as configurações ainda não processadas.
   - Linha 16 - Verifica se o intervalo que está sendo analisado já não está contido em algum intervalo de $\mathbb{I}_\mathrm{P}$.
   - Linha 17 - Se o intervalo não está contido, ele é guardado.

6. Linha 23 - A cada *kex* exemplos processados, ou quando a iteração for correspondente ao último exemplo, uma cobertura mínima é calculada.

7. Linha 28 - Após todos os exemplos de rótulo l terem sido extraídos, os intervalos resultantes são armazenados como sendo a base no nível $l + 1$.

O exemplo a seguir ilustra o funcionamento do algoritmo. Seja novamente o reticulado $(\mathcal{P}(W), \subseteq)$, onde W é um conjunto de 3 elementos e seja $\psi : \mathcal{P}(W) \to \{0, 1, 2, 3\}$ uma função parcialmente definida, conforme tabela a seguir ($*$ indica valor indefinido):

X	$\psi(X)$
000	0
001	*
010	1
011	*
100	1
101	2
110	2
111	3

A figura 6.3 ilustra a dinâmica de funcionamento do algoritmo, enquanto a figura 6.4 mostra a base resultante para a função da tabela acima. A base no nível 0, $\mathbf{B}_0(\psi)$, é o intervalo $[\emptyset, W]$. Para calcular a base no nível 1, todos os elementos de rótulo 0 são extraídos do intervalo $[\emptyset, W]$, resultando em $\mathbf{B}_1(\psi) = \{[001, 111], [010, 111], [100, 111]\}$. Para calcular a base no nível 2, todos os elementos de rótulo 1 são extraídos dos intervalos em $\mathbf{B}_1(\psi)$, resultando em $\mathbf{B}_2(\psi) = \{[001, 111], [110, 111]\}$. Para calcular a base no nível 3, todos os elementos de rótulo 2 são extraídos dos intervalos em $\mathbf{B}_2(\psi)$, resultando em $\mathbf{B}_3(\psi) = \{[011, 111]\}$.

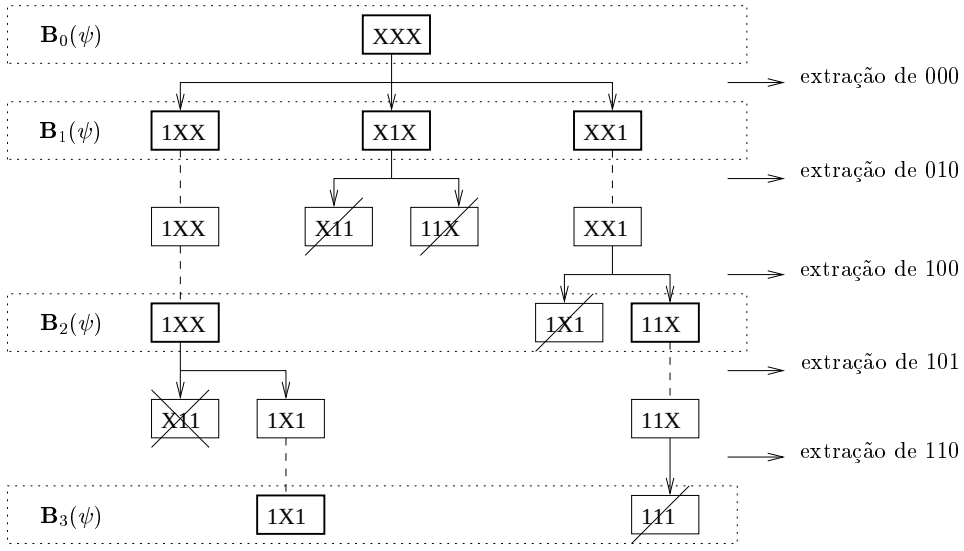


Figura 6.3: Dinâmica do algoritmo ISI multiclassificação.

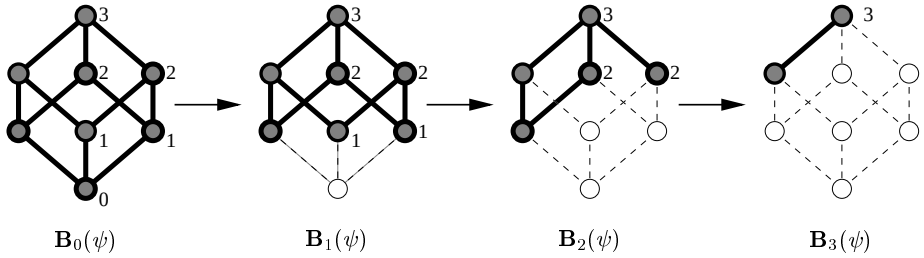


Figura 6.4: Resultado do algoritmo ISI multiclassificação.

A demonstração da corretude do algoritmo ISI binário [73] pode ser facilmente estendida para o caso multiclassificação. A análise de complexidade (tanto em tempo quanto em memória) no caso

médio do algoritmo não é simples pois depende da distribuição das configurações e dos rótulos. A análise de melhor e pior casos são simples. No melhor caso, as configurações tem todas o rótulo 0 e é trivial ver que o algoritmo tem complexidade de tempo e memória lineares no tamanho da entrada. No pior caso, o algoritmo é exponencial tanto em tempo quanto em memória, além do que a função é especificada para um número aproximadamente exponencial de linhas da tabela.

Observe que qualquer número inteiro pode ser representado de forma binária e, conseqüentemente, uma configuração em níveis de cinza pode ser representada como um vetor de números binários. Por exemplo, a configuração $\mathbf{x} = \{2, 0, 1\} \in K^W$, onde $K = [0, 7]$ e $W = \{W_1, W_2, W_3\}$, pode ser representada pelo vetor binário $\mathbf{x} = \{010\,000\,001\}$. Desta forma, poderia-se pensar em usar o algoritmo ISI multiclassificação também para os reticulados níveis de cinza. No entanto, o algoritmo ISI multiclassificação não pode ser usado para o reticulado níveis de cinza pois este não é isomorfo ao reticulado booleano que o representa (via o aumento do número de variáveis).

6.2 ISI níveis de cinza

Nesta seção apresentamos o algoritmo ISI níveis de cinza que é uma extensão natural dos ISI binário e multiclassificação. Do ISI binário estende-se a idéia de quebra de um intervalo para o reticulado níveis de cinza. A idéia de base encadeada (bases nos diferentes níveis) do ISI multiclasificação é adaptada de forma similar para o ISI níveis cinza. Nesta seção voltamos a usar a notação de vetores para os elementos do reticulado das configurações.

6.2.1 Quebra de um intervalo níveis de cinza

Quebrar um intervalo níveis de cinza é, também, um problema simples. Sua descrição algoritmica não é tanto. Dado um intervalo $[\mathbf{a}, \mathbf{b}] \subset K^W$, queremos subtrair de $[\mathbf{a}, \mathbf{b}]$ um ponto (uma configuração), ou um subintervalo $[\mathbf{c}, \mathbf{d}]$ que o intercepta.

Definição 6.3 *A* diferença *entre dois intervalos* $[\mathbf{a}, \mathbf{b}]$ *e* $[\mathbf{c}, \mathbf{d}]$*, é dada por,*

$$[\mathbf{a}, \mathbf{b}] \setminus [\mathbf{c}, \mathbf{d}] = \{\mathbf{x} \in K^W : \mathbf{x} \in [\mathbf{a}, \mathbf{b}] \text{ e } \mathbf{x} \notin [\mathbf{c}, \mathbf{d}]\}. \tag{6.8}$$

O problema algoritmico é descrever essa diferença em termos de intervalos maximais contidos em $[\mathbf{a}, \mathbf{b}] \setminus [\mathbf{c}, \mathbf{d}]$. Para isso vamos introduzir duas funções auxiliares ξ_I e ξ_O.

Definição 6.4 *Seja* $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ *um ponto e sejam* $O = (O_1, O_2, \ldots, O_n)$ *e* $I = (I_1, I_2, \ldots, I_n)$ *o menor e o maior elementos do reticulado* K^W*. As funções* $\xi_I, \xi_O : K^W \times \{1, 2, \ldots n\} \to K^W$ *são definidas pelas fórmulas:*

$$\xi_I(\mathbf{x}, i) = (I_1, I_2, \ldots, I_{i-1}, x_i - 1, I_{i+1}, \ldots, I_n) \tag{6.9}$$

$$\xi_O(\mathbf{x}, i) = (O_1, O_2, \ldots, O_{i-1}, x_i + 1, O_{i+1}, \ldots, O_n) \tag{6.10}$$

Proposição 6.5 *Sejam* $[\mathbf{a}, \mathbf{b}]$ *e* $[\mathbf{c}, \mathbf{d}]$ *dois intervalos de* K^W*, tais que* $[\mathbf{c}, \mathbf{d}] \wedge [\mathbf{a}, \mathbf{b}] \neq \emptyset$*. Seja* $\mathbf{a} = (a_1, a_2, \ldots, a_n)$*,* $\mathbf{b} = (b_1, b_2, \ldots, b_n)$*,* $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ *e* $\mathbf{d} = (d_1, d_2, \ldots, d_n)$*, onde* $a_i, b_i, c_i, d_i \in$

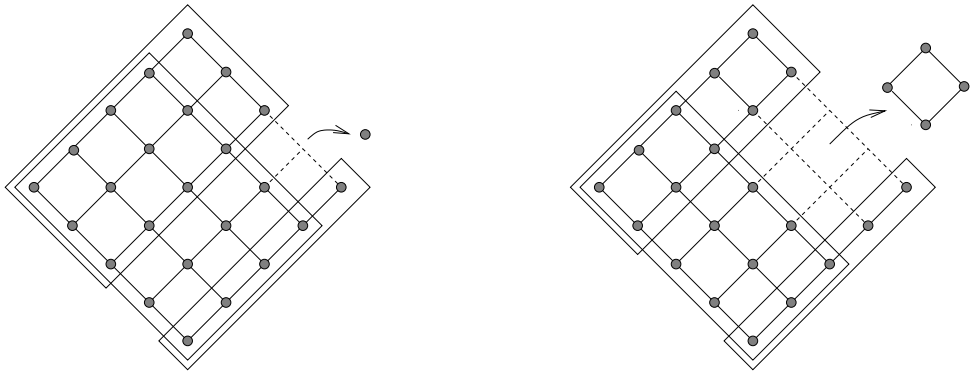
$K, \forall i \in [1,n]$. *O conjunto de intervalos maximais resultantes da subtração do intervalo* $[\mathbf{c},\mathbf{d}]$ *do intervalo* $[\mathbf{a},\mathbf{b}]$ *é denotado por* $\mathbf{M}([\mathbf{a},\mathbf{b}] \setminus [\mathbf{c},\mathbf{d}])$ *e dado pela fórmula,*

$$\mathbf{M}([\mathbf{a},\mathbf{b}] \setminus [\mathbf{c},\mathbf{d}]) = \{[\mathbf{a}, \mathbf{b} \wedge \xi_I(\mathbf{c},i)] : i \in [1,n]\} \cup \{[\mathbf{a} \vee \xi_O(\mathbf{d},i), \mathbf{b}] : i \in [1,n]\}. \tag{6.11}$$

No caso que $c = d = \mathbf{x}$, onde $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, a fórmula simplifica-se para,

$$\mathbf{M}([\mathbf{a},\mathbf{b}] \setminus \{\mathbf{x}\}) = \{[\mathbf{a}, \mathbf{b} \wedge \xi_I(\mathbf{x},i)] : i \in [1,n]\} \cup \{[\mathbf{a} \vee \xi_O(\mathbf{x},i), \mathbf{b}] : i \in [1,n]\}. \tag{6.12}$$

A figura 6.5 ilustra os intervalos resultantes após a extração de um ponto (intervalo trivial) e após a extração de um intervalo, respectivamente.



a b

Figura 6.5: (a) Exemplo de quebra de um intervalo por um ponto (intervalo trivial) e (b) por um intervalo.

Note que, diferentemente do reticulado booleano em que todos os intervalos em $\mathbf{M}([A,B] \setminus \{X\})$ têm dimensão $dim([A,B]) - 1$, no caso do reticulado níveis de cinza, a quantidade e a dimensão dos intervalos depende da localização do elemento extraído no reticulado. Para um intervalo de dimensão n, o número de intervalos resultantes da quebra pode variar de n até $2n$. A figura 6.6 mostra as três possibilidades para a quebra de um intervalo de dimensão 2.

6.2.2 Algoritmo para a construção da base

Sejam dois intervalos $[A_1,B_1]$ e $[A_2,B_2]$, tais que, existe um intervalo $[C,D]$ que intercepta ambos os intervalos (i.e., $[C,D] \cap [A_1,B_1] \neq \emptyset$ e $[C,D] \cap [A_2,B_2] \neq \emptyset$). No algoritmo ISI binário e no algoritmo ISI multiclassificação, se quebramos os intervalos $[A_1,B_1]$ e $[A_2,B_2]$ por $[C,D]$, é possível mostrar que nenhum intervalo resultante da quebra de $[A_1,B_1]$ está propriamente contido em nenhum intervalo resultante da quebra de $[A_2,B_2]$, e vice-versa [73, 118].

No entanto, no ISI níveis de cinza isto não é verdade. Por exemplo, no caso mostrado na figura 6.7, são dados dois intervalos [–1–2, 22] e [–22, 22], e o elemento a ser extraído é 02. Da quebra de [–1–2, 22] resultam três intervalos, [12, 22], [–1 –2, 21] e [–1–2, –12]. Da quebra do segundo resultam [–22, –12] e [12, 22]. Dos intervalos que resultaram da quebra, $[12,22] \leq [1-2,22]$. Isto implica que o algoritmo ISI níveis de cinza deve, além de verificar se os elementos resultantes da quebra de um intervalo por $[\mathbf{c},\mathbf{d}]$ não estão contidos em intervalos de $\mathbb{I}_\mathrm{P}$, verificar também se eles não estão contidos em algum outro intervalo resultante da quebra de outro intervalo por $[\mathbf{c},\mathbf{d}]$.

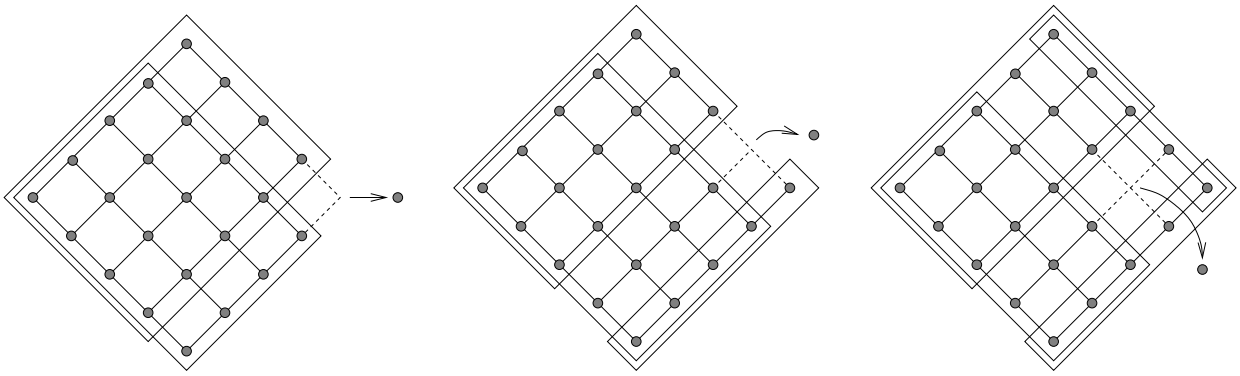


Figura 6.6: Três possíveis localizações de um ponto no reticulado de dimensão 2 e os intervalos resultantes pelas respectivas quebras.

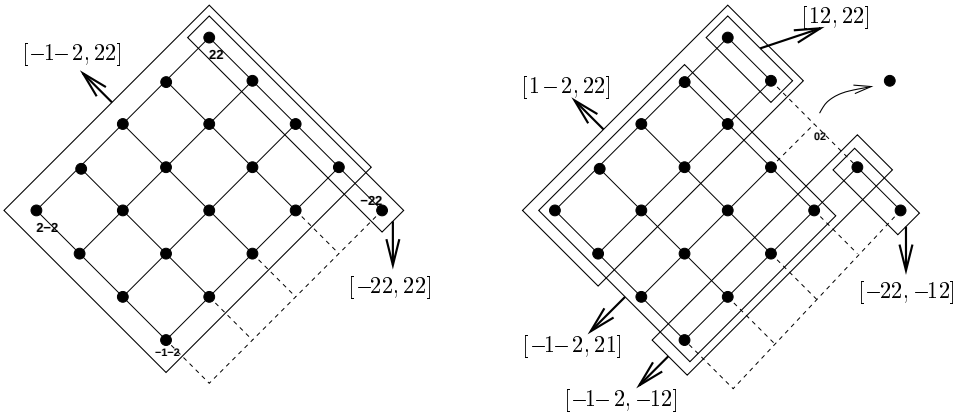


Figura 6.7: Da quebra de dois intervalos por um elemento comum podem resultar intervalos tais que um está contido no outro.

A estrutura do algoritmo ISI níveis de cinza é praticamente o mesmo do ISI multiclassificação. A única modificação necessária está na linha 15 do algoritmo 2, para incluir a verificação adicional comentada acima. Naturalmente, a quebra de intervalos binários da linha 13 deve ser substituída pela quebra de intervalos níveis de cinza.

Algorithm 3 Modificação nas linhas 16, 17 e 18 do algoritmo ISI para encontrar a base de um operador níveis de cinza.

if $[\mathbf{a}', \mathbf{b}'] \not\subset [\mathbf{e}, \mathbf{f}], \forall [\mathbf{e}, \mathbf{f}] \in \mathbb{I}_{\mathrm{P}} \cup \mathbb{I}_{\mathrm{tmp}}$ **then**
  $\mathbb{I}_{\mathrm{New}} \leftarrow \mathbb{I}_{\mathrm{New}} \cup \{[\mathbf{a}', \mathbf{b}']\}$;
end if

A complexidade de implementação do algoritmo para o caso níveis de cinza é maior, assim como o tempo de processamento, pois o reticulado é muito maior para um mesmo número de pontos da janela.

Com o objetivo de contornar o custo computacional envolvido na verificação da linha 15, propomos que para cada novo intervalo $[\mathbf{a}', \mathbf{b}'] \in \mathbb{I}_{\mathrm{split}}$, resultante de uma quebra, seja associada uma lista com todas as configurações $\mathbf{x} \in \mathcal{M}_1$ tais que $\mathbf{x} \in [\mathbf{a}', \mathbf{b}']$. Por um lado isto aumenta a quantidade de

memória usada, mas por outro lado diminui o tempo correspondente à verificação da linha 15, pois basta verificarmos se a lista associada ao intervalo é vazia, ou não.

Vamos ilustrar o funcionamento do algoritmo usando a função parcialmente definida apresentada no capítulo anterior e reapresentada aqui pela tabela 6.1.

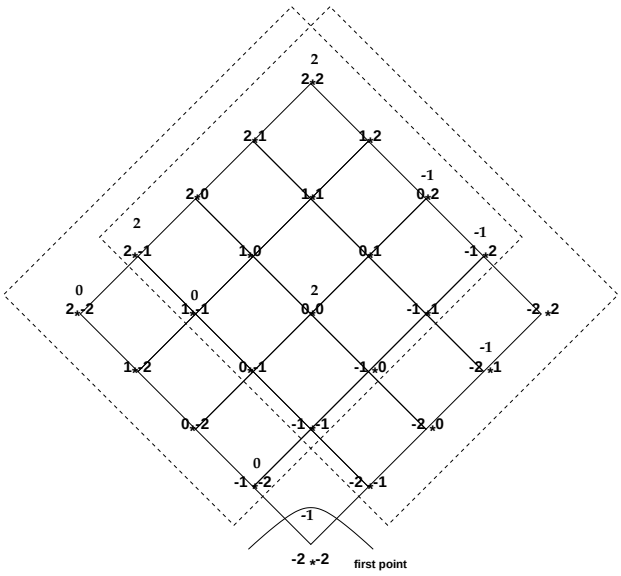
X_1^*	X_2^*	X_3^*	Y^*
-2	0	-2	-1
-2	0	1	-1
-1	0	-2	0
-1	0	2	-1
0	0	0	2
0	0	2	-1
1	0	-1	0
2	0	-2	0
2	0	-1	2
2	0	2	2

Tabela 6.1: Função característica parcialmente definida por $\mathcal{M}$

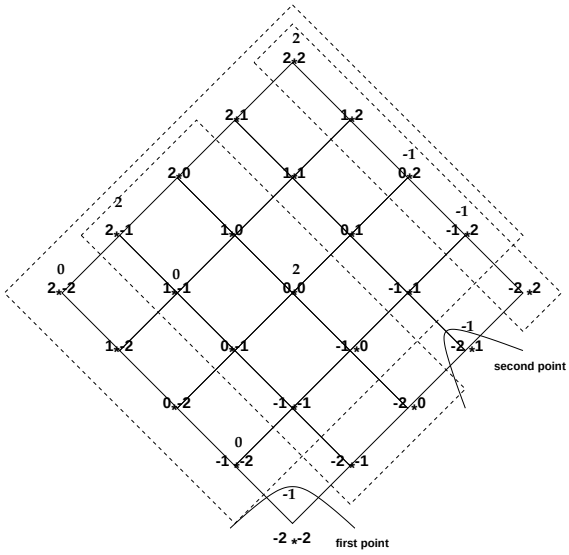
A figura 6.8 mostra os intervalos maximais que resultam após a extração de cada um dos exemplos com rótulo -1. Note que após a extração dos 4 exemplos com rótulo -1, os intervalos resultantes não cobrem nenhum exemplo com rótulo -1, mas cobrem todos os exemplos com rótulo maior que -1.

Em seguida, a figura 6.9 mostra os intervalos maximais que resultam após a extração de cada um dos exemplos com rótulo 0. Os intervalos resultantes após a extração de todos os exemplos com rótulo -1 e 0 cobrem todos os exemplos com rótulo maior que 0, mas não cobrem nenhum exemplo com rótulo menor ou igual a 0.

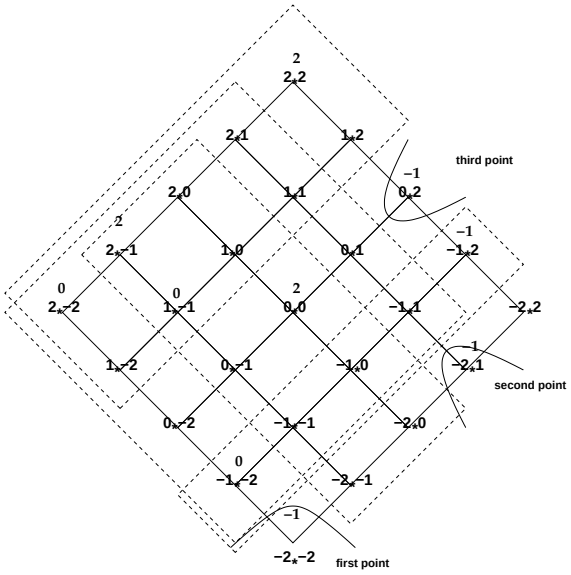
Note que os resultados mostrados nas figuras já estão simplificados, isto é, os intervalos não essenciais foram eliminados.



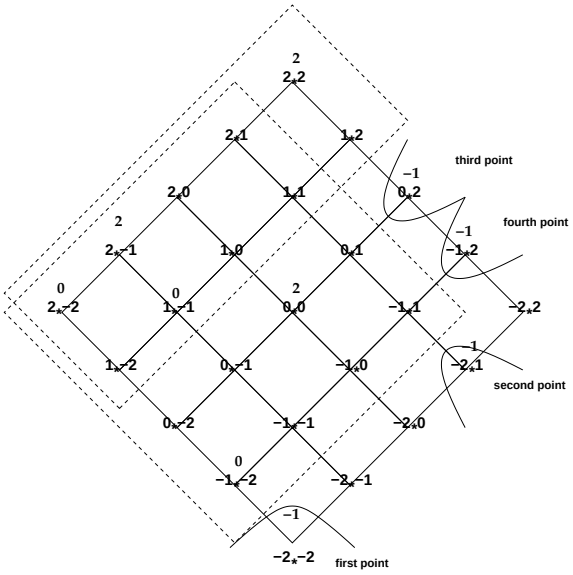
(a) Após a extração de $(-2\ -2)$



(b) Após a extração de (-21)

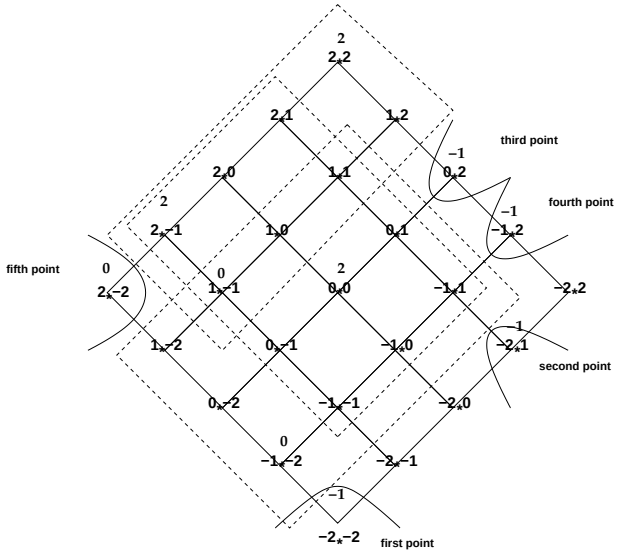


(c) Após a extração de (02)

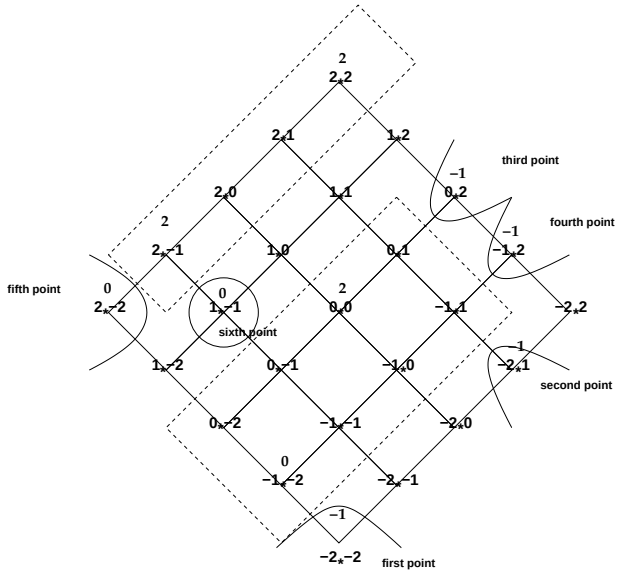


(d) Após a extração de (-12) e cálculo da cobertura mínima

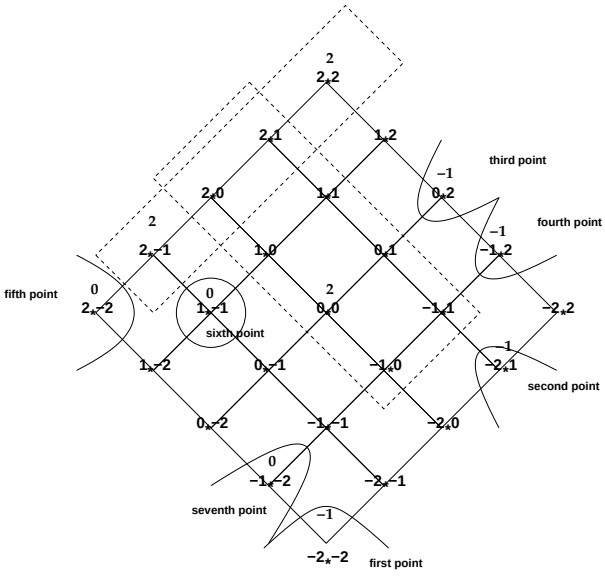
Figura 6.8: Intervalos maximais após extração dos elementos com rótulo -1.



(a) Após a extração de $(2-2)$



(b) Após a extração de $(1-1)$



(c) Após a extração de $(-1-2)$ e cálculo da cobertura mínima

Figura 6.9: Intervalos maximais após extração dos elementos com rótulo 0.

A base no nível -2 ($\mathbf{B}_{-2}$) é o reticulado completo. Como não há exemplos de rótulo -2, a base no nível -1 ($\mathbf{B}_{-1}$) é a mesma do nível -2. Quatro iterações de extração, para extrair todos os exemplos com rótulo -1, foram executados para se obter a base no nível 0 ($\mathbf{B}_0$), e mais três iterações para se obter a base no nível 1 ($\mathbf{B}_1$). Como não há exemplos com rótulo 1, a base do nível 2 é a mesma do nível 1. A dinâmica do algoritmo para o exemplo anterior é ilustrada na figura 6.10.

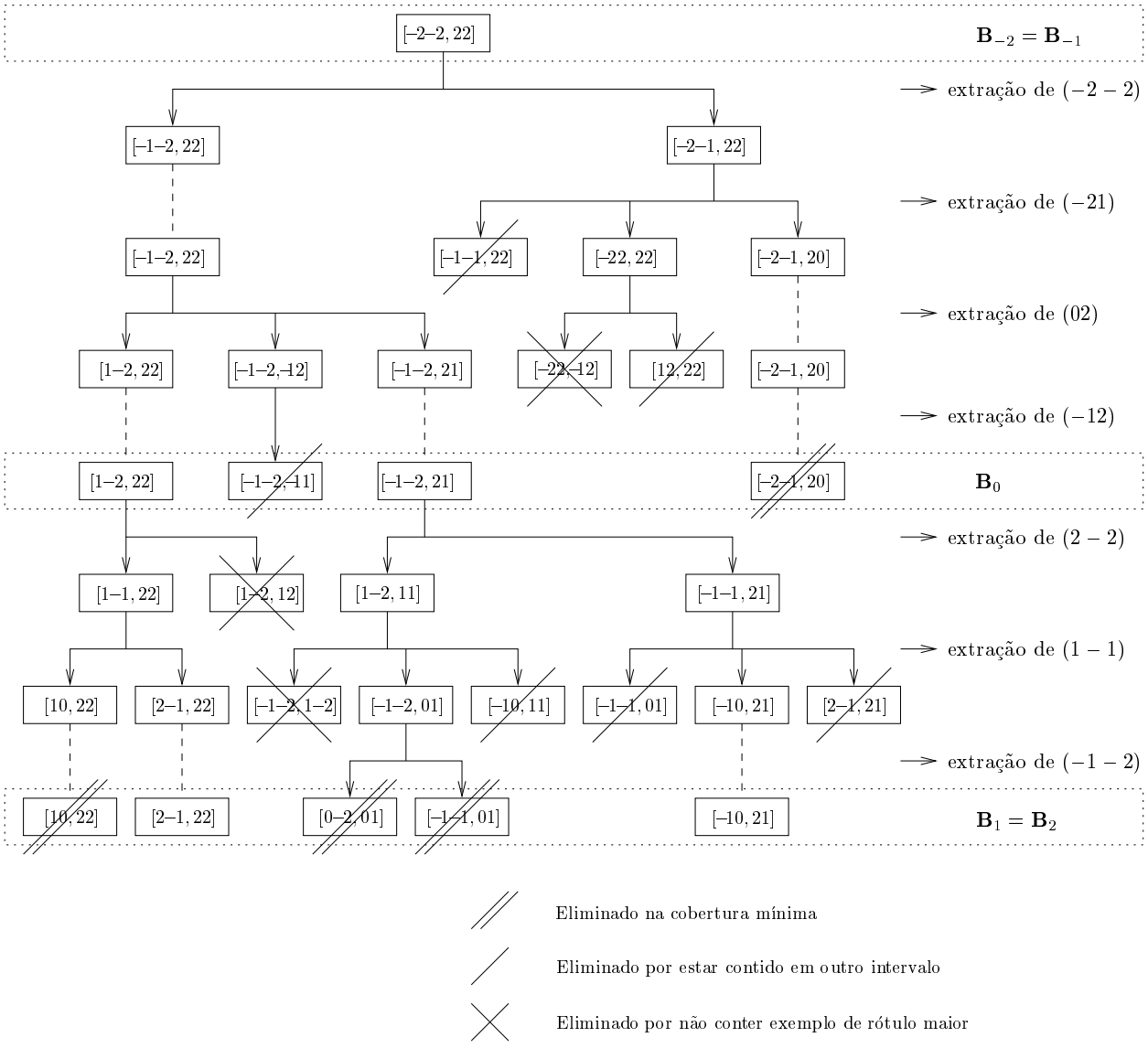


Figura 6.10: Dinâmica do algoritmo ISI níveis de cinza aplicada sobre os dados da tabela 6.1.

A base do operador resultante é ilustrada na figura 6.11.

Note que, tanto esta representação por bases, como as duas mostradas anteriormente, representam a mesma função computacional (parcialmente definida por $\mathcal{M}_W$). Uma diferença importante entre estas representações é a forma como elas classificam as configurações não observadas. A tabela 6.2 apresenta todas as configurações possíveis e os respectivos valores que serão atribuidos por cada

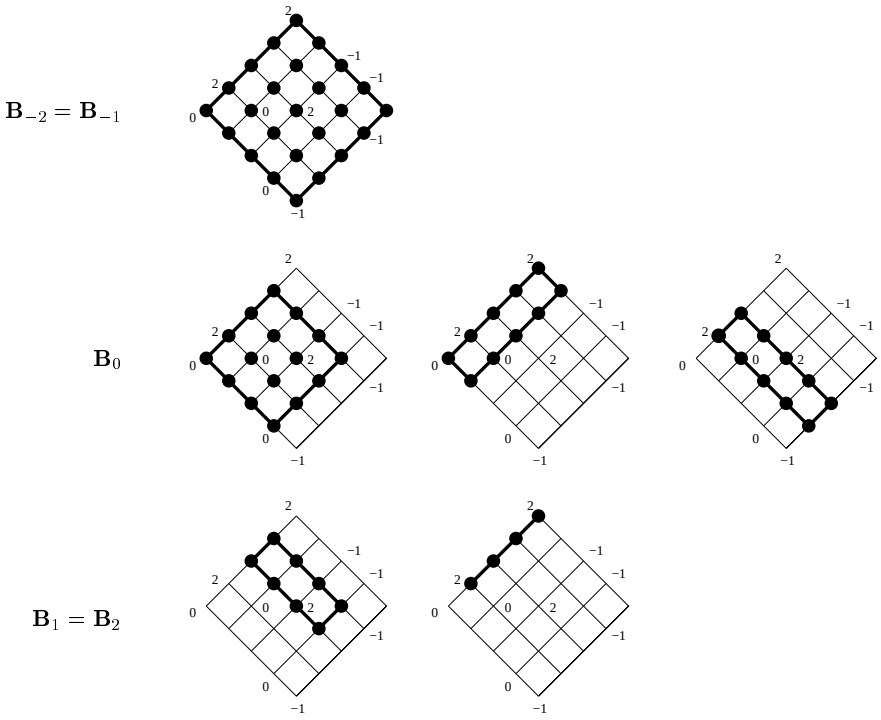


Figura 6.11: Base resultante da aplicação do ISI sobre os dados da tabela 6.1.

uma das representações: árvore de decisão paralela (PDT), árvore de decisão oblíqua (ODT) e ISI. Observe que todos eles são consistentes com a função inicial parcialmente definida por $\mathcal{M}_W$.

6.3 Algoritmo para a aplicação do operador

A aplicação de um W-operador binário para multiclassificação, ou de um operador "aperture", Ψ, quando representado pela sua base $\mathbf{B}_\Psi$, é muito simples. Dada uma imagem f onde o operador será aplicado, o algoritmo deve varrer todos os pontos da imagem[2] e observar a configuração vista por W centrada no ponto onde o operador está sendo aplicado. Dada a configuração observada, digamos $\mathbf{x}$, o algoritmo tem que verificar se $\mathbf{x} \in \mathbf{B}$, isto é, tem que verificar se $\mathbf{x} \in \mathcal{I}$ para algum $\mathcal{I} \in \mathbf{B}$. Como a base está ordenada do maior rótulo para o menor e o resultado do operador é o supremo dos rótulos associados aos intervalos em que ele pertence, então podemos parar a busca no primeiro intervalo que contenha a configuração observada. Descreveremos abaixo o algoritmo para aplicação do operador para os operadores multiclassificação. O algoritmo para a aplicação do operador níveis de cinza é semelhante.

[2]Todos os pontos da imagem em que W esteja totalmente contido no domínio da imagem.

X_1^*	X_2^*	X_3^*	$\mathcal{M}_W$	PDT	ODT	ISI
-2	0	-2	-1	-1	-1	-1
-2	0	-1	*	-1	-1	0
-2	0	0	*	0	-1	0
-2	0	1	-1	-1	-1	-1
-2	0	2	*	0	-1	-1
-1	0	-2	0	0	0	0
-1	0	-1	*	0	-1	0
-1	0	0	*	0	-1	2
-1	0	1	*	-1	-1	2
-1	0	2	-1	-1	-1	-1
0	0	-2	*	2	0	0
0	0	-1	*	2	0	0
0	0	0	2	2	2	2
0	0	1	*	-1	-1	2
0	0	2	-1	-1	-1	-1
1	0	-2	*	0	0	0
1	0	-1	0	0	0	0
1	0	0	*	0	2	2
1	0	1	*	0	2	2
1	0	2	*	0	2	0
2	0	-2	0	0	0	0
2	0	-1	2	2	2	2
2	0	0	*	2	2	2
2	0	1	*	2	2	2
2	0	2	2	2	2	2

Tabela 6.2: Comparação dos valores atribuídos.

6.3.1 Algoritmo para aplicação da base de um operador multiclassificação

Dada uma imagem $f \in \{0,1\}^E$ e um W-operador $\Psi : \{0,1\}^E \to [0,k]^E$ representado pela sua base $\mathbf{B} = \{\mathcal{I}_1, \ldots, \mathcal{I}_m\}$ (onde $\ell(\mathcal{I}_i) \geq \ell(\mathcal{I}_{i+1})$, $\forall i \in \{1, 2, \ldots, |\mathbf{B}| - 1\}$), o algoritmo que produz $g = \Psi(f)$ é dado por:

Algorithm 4 Algoritmo para aplicação do W-operador $\Psi : \{0,1\}^E \to [0,k]^E$.

```
1:  for all t ∈ E do
2:     j ← 0 ;
3:     not_found ← True ;
4:     X ← (f(t + w_1), f(t + w_2), ..., f(t + w_|W|)) ;
5:     while j ≤ |B| and not_found do
6:        if X ∈ I_j then
7:           g(t) ← ℓ(I_j) ;
8:           not_found ← False ;
9:        else
10:          j ← j + 1 ;
11:       end if
12:    end while
13: end for
```

Embora simples, este algoritmo não é eficiente pois, para cada configuração, no pior caso, todos os intervalos têm que ser examinados.

Uma alternativa para deixar a aplicação mais eficiente é, após encontrada a base, transformar a representação da função computacional em uma árvore de decisão, ou mesmo, em um diagrama de decisão [133, 134, 135, 136, 137, 138]. No caso binário existem vários algoritmos para essa transformação de representação, sendo que um deles foi implementado no sistema [77]. No caso da base de operadores níveis de cinza, pesquisas vêm sendo realizadas nesse sentido por H. M. F. Madeira [139, 140].

Capítulo 7

Representação e indução via multiresolução

A motivação principal para introduzir os operadores "aperture", como vimos anteriormente, foi que seu projeto é viável graças a restrição adequada do domínio espacial e do domínio dos níveis de cinza. A combinação destas restrições diminui a quantidade de configurações que precisam ser estimadas, e conseqüentemente a quantidade de exemplos necessária para que o erro de estimação seja aceitável. Outros tipos de restrições são possíveis e foram testadas [141, 78, 59, 11, 142, 60, 51, 61, 126, 143, 144, 145, 146, 147, 148, 149, 70, 150], e o objetivo também é tornar o projeto de operadores possível diminuindo a quantidade de exemplos necessária para uma boa estimação. Embora não exista uma classificação consagrada dos tipos de restrições possíveis, o projeto de operadores via multiresolução classifica-se como um projeto com restrição do tipo algébrico.

Neste capítulo, vamos mostrar como combinar adequadamente as informações de duas ou mais resoluções para construir um operador de imagens. A idéia, novamente, é diminuir o espaço de operadores para facilitar o projeto. Neste caso, a idéia motiva-se no fato que filtrar otimamente uma imagem com janelas grandes é melhor do que filtrar com janelas pequenas. Porém, o aumento da janela traz consigo um aumento no número de variáveis aleatórias e, conseqüentemente, um aumento no erro do operador estimado. A essa idéia dá-se o nome de projeto multiresolução piramidal. A palavra piramidal vem do fato que ela usa a restrição da resolução de forma hierárquica, da resolução mais alta para a mais baixa. A idéia foi inicialmente usada para projetar operadores binários [142] e mostrou-se experimentalmente que diminuía-se o erro do operador projetado em comparação com outras técnicas de projeto de operadores. Seguindo o sucesso do projeto binário e combinando a idéia com o projeto de operadores "aperture", estendemos o projeto multiresolução de operadores "aperture" restringindo a resolução no domínio espacial, no domínio dos níveis de cinza e em ambos os domínios. Um sistema que aplica estas idéias foi implementado e está descrito juntamente com a análise de sua complexidade de espaço e tempo. Os exemplos de aplicação se restringiram desembaçar sinais e imagens e foram comparados com os operadores "aperture" e com os operadores lineares.

7.1 Restrição no domínio do operador

Nesta seção mostraremos as idéias principais de se restringir por resolução o domínio dos operadores "aperture". Como o domínio desses operadores pode ser restrito tanto espacialmente, quanto nos

níveis de cinza, vamos analisá-los separadamente.

Sejam $W_0, W_1, \ldots, W_n$ uma conjunto de janelas tais que $W_i \subseteq W_{i-1} \subseteq E$, $0 \leq i \leq n$. Denominaremos este conjunto de *pirâmide espacial* de janelas. Seja $L = [0, \ldots, l-1]$ o domínio dos níveis de cinza das funções características de $D_i = L^{W_i}$ (o espaço de configurações sobre W_i) em L.

Seja $\rho_j : D_{j-1} \to D_j$, $1 \leq j \leq n$ uma *transformação de resolução* entre os dois espaços de configurações, D_{j-1} e D_j. Seja $w_{i,j} \in W_i$ o j-ésimo ponto da janela W_i na resolução i. Um exemplo de tal transformação entre L^{W_0} e L^{W_1}, onde $W_0 = \{w_{0,1}, w_{0,2}, w_{0,3}, w_{0,4}, w_{0,5}\}$ e $W_1 = \{w_{1,1}, w_{1,2}, w_{1,3}\}$, é dado por, se $\mathbf{x} \in D_0 = L^{W_0}$ e $\mathbf{z} \in D_1 = L^{W_1}$, $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5) \in D_0 = L^{W_0}$, então $\mathbf{z} = \rho(\mathbf{x}) = (z_1 = x_1, z_2 = x_3, z_3 = x_5)$. A figura 7.1 ilustra tal exemplo (os quadrados cinza são os pontos pertencentes às janelas W_0 e W_1).

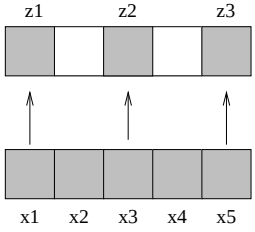


Figura 7.1: Mudança de resolução espacial

Note que o número de configurações em D_1 é $|D_1| = |L|^{|W_1|}$ e o número de configurações em D_0 é $|D_0| = |L|^{|W_0|}$, portanto, $|D_0| = |L|^{(|W_0|-|W_1|+|W_1|)} = |L|^{(|W_0|-|W_1|)}|D_1|$. Ou seja, o número de configurações em D_0 é $|L|^{|W_0|-|W_1|}$ vezes maior que em D_1.

Uma outra transformação que diminui o espaço de operadores é restringir os níveis de cinza do espaço de configurações. Sejam $L_0, L_1, \ldots, L_n$ um conjunto de intervalos dos inteiros tal que $L_i \subseteq L_{i-1}$, $1 \leq i \leq n$. Denominaremos este conjunto de *pirâmide níveis de cinza*. Seja $W \subset E$ uma janela e $D_i = L_i^W$ o espaço de configurações de W em L_i.

Seja $\rho'_j : D_{j-1} \to D_j$, $1 \leq j \leq n$ uma *transformação de resolução nos níveis de cinza* entre os dois espaços de configurações D_{j-1} e D_j. Um exemplo de tal transformação entre L_0^W e L_1^W, onde $L_0 = [0, 7]$ e $L_1 = [0, 3]$ são dois intervalos de Z, é dado por, se $\mathbf{x} \in D_0 = L_0^W$ e $\mathbf{z} \in D_1 = L_1^W$, então $\mathbf{z} = \rho(\mathbf{x}) = \lfloor \frac{\mathbf{x}}{2} \rfloor$. A figura 7.2 ilustra tal exemplo (os retângulos representam os níveis de cinza da janela).

Note que o número de configurações em D_1 no caso em que $\mathbf{z} = \rho(\mathbf{x}) = \lfloor \frac{\mathbf{x}}{2} \rfloor$ é $|D_1| = |L_1|^{|W|}$ e o número de configurações em D_0 é $|D_0| = |L_0|^{|W|} = |2L_1|^{|W|} = 2^{|W|}|L_1|^{|W|} = 2^{|W|}|D_1|$.

Ambas transformações ρ e ρ' podem ser combinadas em uma única transformação. Sejam $W_0, W_1, \ldots, W_n$ uma conjunto de janelas tais que $W_i \subseteq W_{i-1} \subseteq E$, e sejam $L_0, L_1, \ldots, L_n$ um conjunto de intervalos dos inteiros tal que $L_i \subseteq L_{i-1}$, $1 \leq i \leq n$. O conjunto formado por $W_i \times L_i$ será denominado de *pirâmide de "apertures"*. O espaço das configurações "aperture" de W_i em L_i será denotado $D_i = L_i^{W_i}$.

Seja $\rho'_j : D_{j-1} \to D_j$, $1 \leq j \leq n$ uma *transformação de resolução nos níveis de cinza* entre os dois espaços de configurações tal que $\mathbf{z} = \rho(\mathbf{x})$, $\mathbf{z} \in D_j$ e $\mathbf{x} \in D_{j-1}$. Um exemplo de tal transformação entre $L_0^{W_0}$ e $L_1^{W_1}$ é dado por $\mathbf{z} = \rho(\mathbf{x}) = (z_1 = \lfloor \frac{x_1}{2} \rfloor, z_2 = \lfloor \frac{x_3}{2} \rfloor, z_3 = \lfloor \frac{x_5}{2} \rfloor)$, onde $W_0 = \{w_{0,1}, w_{0,2}, w_{0,3}, w_{0,4}, w_{0,5}\}$, $W_1 = \{w_{1,1}, w_{1,2}, w_{1,3}\}$, $L_0 = [0, 7]$, $L_1 = [0, 3]$, $\mathbf{x} =$

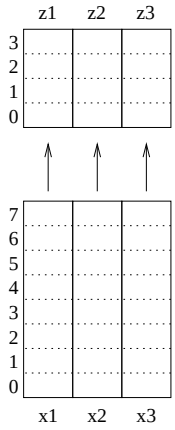


Figura 7.2: Mudança de resolução nos níveis de cinza

$(x_1, x_2, x_3, x_4, x_5) \in D_0 = L_0^{W_0}$ e $\mathbf{z} = (z_1, z_2, z_3) \in D_1 = L_1^{W_1}$. A figura 7.3 ilustra tal exemplo.

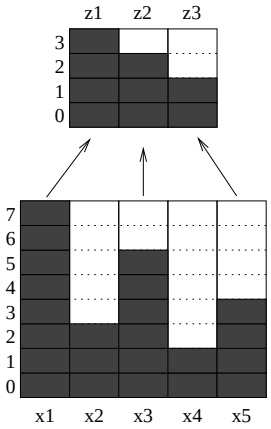


Figura 7.3: Mudança de resolução no espaço e nos níveis de cinza

7.1.1 Projeto multiresolução piramidal de operadores

Para os espaços $D_0 = L_0^{W_0}$ e $D_1 = L_1^{W_1}$, onde $W_1 \subseteq W_0$ e $L_1 \subseteq L_0$, a probabilidade de selecionar uma configuração particular em D_0, $P(\mathbf{x} \in D_0)$, assumindo que a distribuição das configurações é uniforme, é $P(\mathbf{x} \in D_0) = (|L_0|^{|W_0|})^{-1}$, e em D_1 é $P(\mathbf{x} \in D_1) = (|L_1|^{|W_1|})^{-1}$. Como $|W_1| \leq |W_0|$ e $|L_1| \leq |L_0|$, então $P(\mathbf{x} \in D_0) \leq P(\mathbf{x} \in D_1)$. No contexto de multiresolução, se $\mathbf{z} \in D_1$ e $\rho : D_0 \to D_1$ é a transformação de resolução, então o número de vezes que $\mathbf{z}$ é observada é determinado por $\rho^{-1}(\mathbf{z})$ da seguinte maneira:

$$N(\mathbf{z}) = \sum_{\mathbf{x}:\rho(\mathbf{x})=\mathbf{z}} N(\mathbf{x}) \tag{7.1}$$

Em outras palavras, como o número de configurações possíveis diminui, então o número de vezes que elas são observadas tende a aumentar. Assim, o filtro projetado em uma resolução mais baixa terá uma melhor estimativa das probabilidades condicionais que o filtro projetado em resoluções mais altas. Porém, não podemos esquecer que existe um compromisso entre a diminuição da resolução e o erro do operador.

A abordagem multiresolução produz uma função ψ_1 sobre D_1 e aplica-a sobre D_0 (onde D_1 é um espaço restrito de D_0) por $\psi_0(\mathbf{x}) = \psi_1(\mathbf{z})$, onde $\mathbf{z} = \rho(\mathbf{x})$, e $\rho : D_0 \to D_1$. Isto significa que, mesmo tendo dados para estimar os filtros relativamente a D_0, nós só usamos dados relativos a D_1 via ρ.

Uma abordagem alternativa é usar dados de ambos espaços. Se temos uma boa estimativa de $p_{\mathbf{x}}$ (a distribuição de probabilidade de $\mathbf{x}$) e $\psi_{0,N}(\mathbf{x}) \neq \psi_{1,N}(\mathbf{z})$, então é melhor usar $\psi_{0,N}(\mathbf{x})$ em vez de $\psi_{1,N}(\mathbf{z})$. Por outro lado, se temos uma estimativa ruim, ou nenhuma estimativa, de $p_{\mathbf{x}}$, mas uma estimativa melhor de $p_{\rho(\mathbf{x})}$, então é melhor usar $\psi_{1,N}(\mathbf{z})$. Assim, ao invés de aplicar apenas $\psi_{0,N}$, ou $\psi_{1,N}$, para todos os $\mathbf{x}$, a função é escolhida considerando-se a qualidade das estimativas de probabilidade. No caso mais simples, é necessário que $\mathbf{x}$ seja observado apenas uma vez e o estimador multiresolução é dado por,

$$\psi_{(0,1),N}(\mathbf{x}) = \begin{cases} \psi_{0,N}(\mathbf{x}), & \text{if} \quad N(\mathbf{x}) > 0 \\ \psi_{1,N}(\rho_1(\mathbf{x})), & \text{if} \quad N(\mathbf{x}) = 0 \end{cases} \tag{7.2}$$

Esta idéia pode ser repetida para qualquer número de restrições e aplicada iterativamente por uma seqüência de operadores restritos pela resolução, $\rho_1, \rho_2, \ldots, \rho_m$, até que o tamanho de W_i ($W_i \subset W_{i-1}$) seja 1, e o tamanho de L_i ($L_i \subset L_{i-1}$) seja 2. Neste caso, o estimador multiresolução é dado por,

$$\psi_{(0,\ldots,m),N}(\mathbf{x}) = \begin{cases} \psi_{0,N}(\mathbf{x}), & \text{if} \quad N(\mathbf{x}) > 0 \\ \psi_{1,N}(\rho_1(\mathbf{x}), & \text{if} \quad N(\mathbf{x}) = 0, N(\rho_1(\mathbf{x})) > 0 \\ & \vdots \\ \psi_{m-1,N}(\rho_{m-1}(\mathbf{x})), & \text{if} \quad N(\mathbf{x}) = 0, \ldots, N(\rho_{m-1}(\rho_{m-2} \ldots \rho_2(\rho_1(\mathbf{x})) > 0 \\ \psi_{m,N}(\rho_m(\mathbf{x})), & \text{if} \quad N(\mathbf{x}) = 0, \ldots, N(\rho_{m-1}(\rho_{m-2} \ldots \rho_2(\rho_1(\mathbf{x})) = 0 \end{cases} \tag{7.3}$$

A figura 7.4a mostra um exemplo em que ρ_0 mapeia cada quatro pixels da janela W_0 para um pixel em W_1 e ρ_1 mapeia cada dois pixels da janela W_1 para um pixel em W_2. Este é o caso típico de projeto de operadores multiresolução tanto no caso binário quanto no caso níveis de cinza.

Uma ligeira modificação desta abordagem em 7.3 pode ser benéfica se exigirmos que $N(\mathbf{x})$ seja suficientemente grande (i.e., $\psi_{(0,1),N}(\mathbf{x}) = \psi_{0,N}(\mathbf{x})$, se $N(\mathbf{x}) > \alpha$, onde $\alpha \in \mathbf{N}$), então podemos evitar o uso de uma estimação ruim de $\psi_{0,N}(\mathbf{x})$.

Uma outra modificação interessante em 7.3 é trocar $\psi_{j,N}$, onde $j \in [1, m]$, por uma outra função $\psi_{\text{des},N}$, que pode ser um filtro linear [8], ou um outro filtro qualquer projetado via métodos de aprendizado computacional [78, 54, 7]. A motivação para isto é que, se usarmos um método de aprendizado que tenha um bom poder de indução para o problema que está sendo resolvido, então o operador multiresolução híbrido, representado por $\psi_{(0,j),N}$, será melhor que o operador cuja função característica é $\psi_{\text{des},N}$ pois ele é igual ao valor observado para as janelas maiores que W_j e para as configurações de $L_j^{W_j}$ não observadas, ele é igual a $\psi_{\text{des},N}$.

A escolha do nível j onde será substituída a função $\psi_{\text{des},N}$ é feita encontrando-se a função característica $\psi_{W,N}$ cujo erro de projeto para as N amostras de treinamento seja o menor.

7.1.2 Implementação do "software"

A implementação do sistema usado para projetar e aplicar os operadores multiresolução é simples e usa parcialmente as funções que já foram explicadas anteriormente. Em termos de projeto, agora o sistema consiste de três partes separadas: (*i*) *observação* ou *estimação*, (*ii*) *decisão* e (*iii*) *aplicação*.

- O procedimento de estimação (*i*) consiste da mesma biblioteca de funções usada anteriormente para projetar o operador "aperture", mais algumas funções extras para tratar as transformações de restrição. Uma pequena diferença é que a leitura da especificação da "aperture" é substituída pela leitura da especificação de uma pirâmide de "apertures", ou janelas espaciais e de níveis de cinza. Assim, ao invés de uma tabela de exemplos e de níveis de cinza associados a cada exemplo, $\mathcal{X}$, estimam-se m tabelas, uma para cada resolução da pirâmide. O processo de construção dessas tabelas é simples e consiste em transladar a "aperture" $W_0 \times L_0$ sobre o conjunto de imagens de treinamento e montar a tabela $\mathcal{X}_0$ correspondente. A construção das outras tabelas faz-se a partir de $\mathcal{X}_0$ aplicando-se os operadores de restrição ρ_i sobre cada configuração $\mathbf{x}$ observada por $W_0 \times L_0$. O resultado deste procedimento é uma tabela $\mathcal{X}_j$ para cada função característica ψ_j.
- O procedimento de decisão (*ii*) é exatamente o mesmo que o explicado anteriormente, mas agora tem que ser aplicado sobre todas as tabelas $\mathcal{X}_i$, $1 \leq i \leq m$, resultando em $\mathcal{M}_i$.
- O procedimento de aplicação (*iii*) também é semelhante ao anterior, isto é, leitura da imagem onde será aplicado o operador multiresolução, leitura da especificação da pirâmide e leitura das tabelas $\mathcal{M}_i$. Depois, translada-se a "aperture", $W_0 \times L_0$, sobre a imagem e, para cada configuração observada, busca-se a primeira ocorrência dela nas tabelas $\mathcal{M}_i$, $0 \leq i \leq l_i$, nessa ordem. Em seguida, armazena-se esse rótulo associado àquela configuração na imagem de saída. Essa busca do rótulo é feita na ordem decrescente das janelas da pirâmide, isto é, se a configuração não foi achada na tabela i, ela é transformada por ρ_j, $1 \leq j \leq m$ e procurada na tabela seguinte, $i+1$. Se a pirâmide foi bem projetada, a configuração sempre vai terminar rotulada.

A complexidade de memória é linear no tamanho de cada imagem observada vezes o número de pares de imagens, vezes o número de janelas na pirâmide. Mesmo sendo linear, não quer dizer que não ocupe muito espaço, pois o número de pares de imagens pode ser grande e a pirâmide complexa. Note que, quanto maior a janela, maior serão as tabelas $\mathcal{X}_i$ e $\mathcal{M}_i$ (a menos de casos especiais de pouca utilidade prática) pois o número de configurações diferentes cresce e a probabilidade de achar configurações iguais decresce.

A complexidade de tempo da aplicação, no pior caso, é igual a soma dos logaritmos[1] do número de diferente configurações em cada nível da pirâmide vezes o número de pontos do domínio da imagem. Ou seja, no pior caso o valor da imagem no ponto em que o operador está sendo aplicado só é encontrado após varrer a árvore toda.

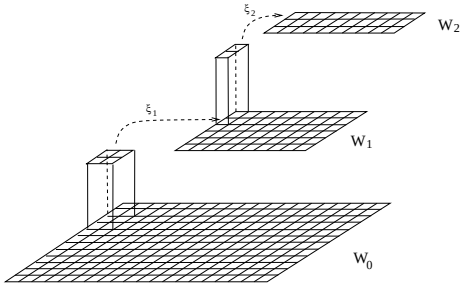
[1] Logaritmo na base 2 pois as tabelas estão ordenadas por construção e assim podemos usar busca binária

7.2 Exemplos ilustrativos

Para ilustrar a aplicação do projeto multiresolução híbrido, desembaçaremos imagens. Há três possibilidades para projetar os operadores "aperture" multiresolução: (*i*) restrição da janela espacial, (*ii*) restrição da janela dos níveis de cinza e (*iii*) restrição de ambas janelas. Como o número de possibilidades é muito grande, vamos nos limitar a apenas algumas combinações dessas três possibilidades.

7.2.1 Desembaçando imagens 2D

Esta aplicação envolve desembaçar imagens geradas por um modelo booleano [127] cuja função primária é piramidal, com no máximo 16 níveis de cinza. Foram geradas 20 imagens seguindo esse modelo, e as respectivas imagens embaçadas . O embaçamento foi feito com um kernel de convolução 3×3 não plano (veja fig. 7.4b). A figura 7.5 mostra parte de uma imagem original e de uma imagem embaçada.

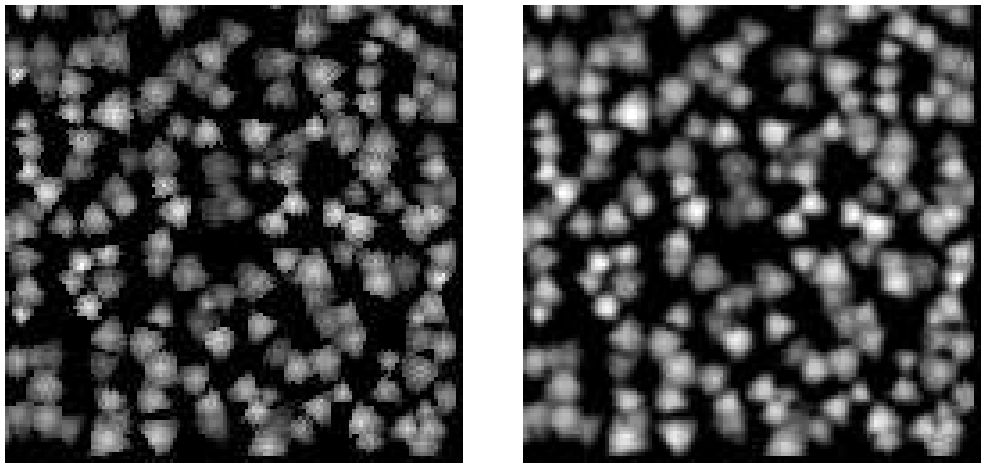


(a)

1	2	1
2	3	2
1	2	1

(b)

Figura 7.4: (a) Estrutura piramidal, (b) Kernel da convolução.



(a) (b)

Figura 7.5: Função aleatória booleana: parte de uma imagem normal (a) e embaçada (b)

O conjunto inicial foi dividido em 10 pares de imagens para treinamento e 10 pares para teste. Usando os pares de treinamento, projetamos operadores "aperture" multiresolução e testamos esses

operadores com o conjunto de teste. As comparações foram feitas com experimentos anteriores, onde treinamos e testamos operadores lineares restritos e operadores "aperture" não-multiresolução com esse mesmo conjunto de treinamento e teste. A figura 7.6 mostra o MSE para alguns dos melhores operadores projetados. Cada curva representa um experimento com uma certa restrição. Cada ponto da curva representa o MSE do operador (a média sobre 10 imagens de teste) após uma certa quantidade de exemplos de treinamento.

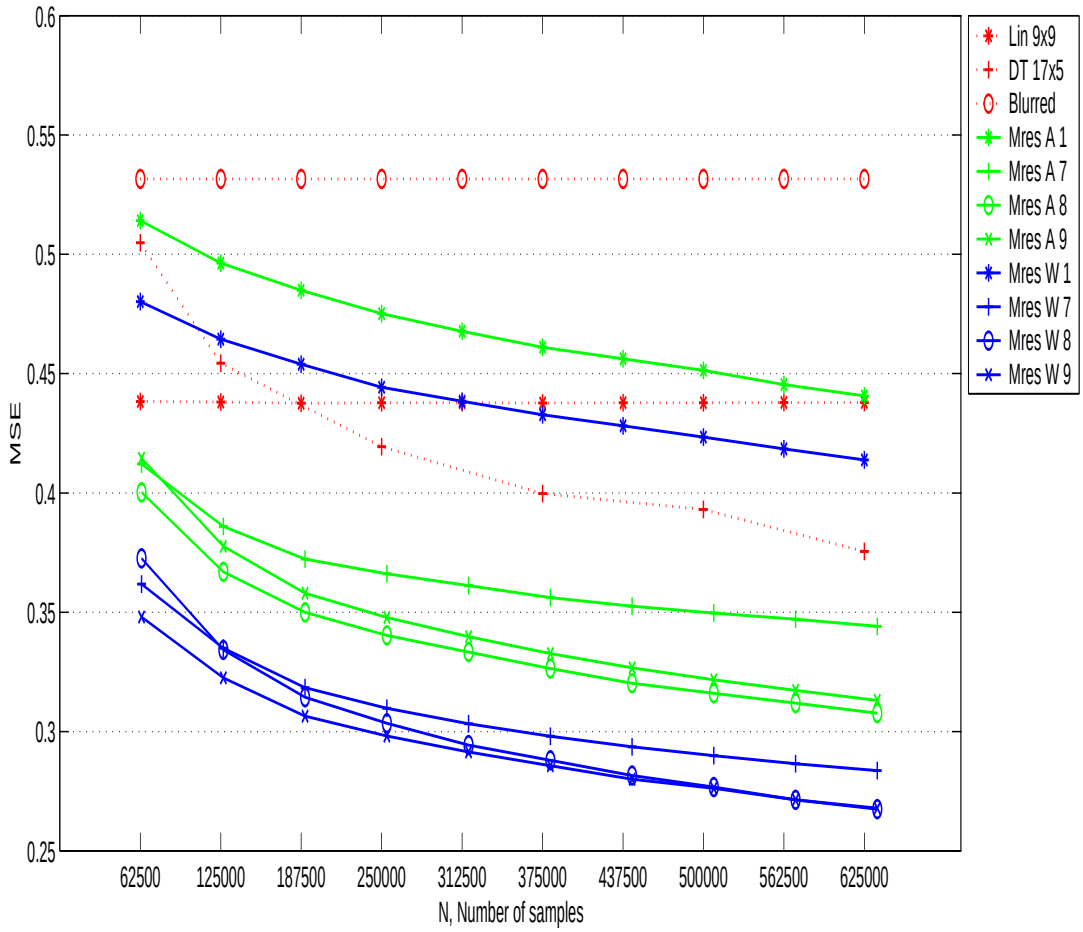
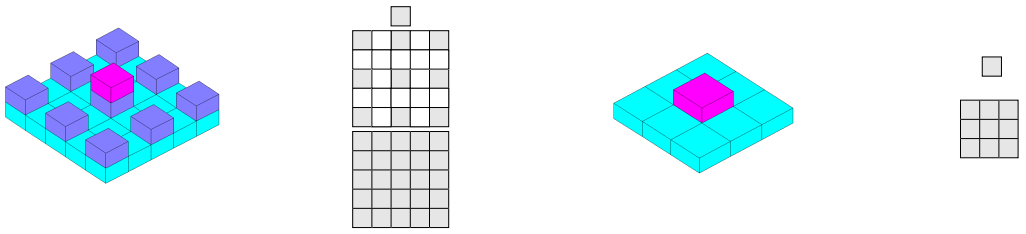


Figura 7.6: Comparação do MSE: pirâmides baixas

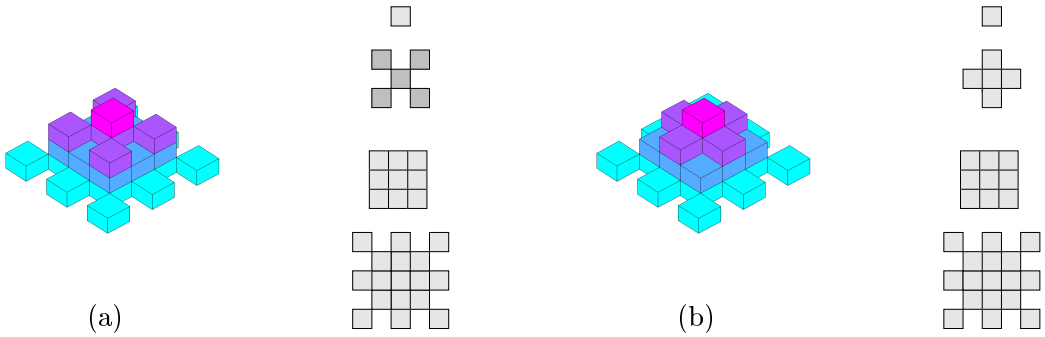
A curva rotulada "blurred" é o erro entre a imagem embaçada e a imagem ideal (média de 10 imagens). A curva rotulada "Lin 9x9" é o resultado do operador linear restrito ótimo [151] projetado usando uma janela de 9×9 pontos. As figuras 7.7a e b mostram, respectivamente, as pirâmides usadas para projetar os W-operadores rotulados "Mres W 1" e "Mres W 7". As figuras 7.8a e b mostram, respectivamente, as pirâmides usadas para projetar os W-operadores rotulados "Mres W 8" e "Mres W 9".

A curva rotulada "$DT17 \times 5$" é o operador "aperture" projetado usando uma janela de 17 pontos (nível 0 da pirâmide mostrada na figura 7.8a).



(a) (b)

Figura 7.7: Pirâmides para os experimentos “Mres W 1” e “Mres W 7”



(a) (b)

Figura 7.8: Pirâmides para os experimentos “Mres W 8” e “Mres W 9”

As curvas rotuladas “Mres A 1” , “Mres A 7”, “Mres A 8” e “Mres A 9” são os MSEs para operadores “aperture” multiresolução projetados usando as pirâmides mostradas nas figuras 7.7a, 7.7b, 7.8a e 7.8b, respectivamente. Cada “aperture” na pirâmide tem $K = [-5, 5]$.

Como esperávamos, a maioria dos operadores “aperture” são melhores que os operadores lineares, que estabilizam ao redor de 0.438. Para aproximadamente 62500 exemplos de treinamento, os “apertures” multiresolução “Mres A 1” e “Mres W 1” são piores que os que não são multiresolução ($DT17 \times 5$). Os melhores resultados são para os operadores rotulados “Mres W 8” e “Mres W 9” (MSE(“Mres W 8”) < MSE(“Mres W 9”)); seguidos pelos operadores rotulados “Mres W 7”, “Mres A 9”, “Mres A 8” e “Mres A 7” (MSE(“Mres W 7”) < MSE(“Mres A 9”) < MSE(“Mres A 8”) < MSE(“Mres A 7”)).

O modelo booleano usado (pirâmides baixas sobre um fundo de valor zero) favorecem os W-operadores sobre os operadores “aperture” com $k << l$ pois como o intervalo de níveis de cinza é pequeno, tem-se uma melhor estimação das probabilidades condicionais para cada padrão, assim como um menor número de padrões.

7.2.2 Desembaçando imagens 2D: intervalo grande

O exemplo anterior motivou-nos a este segundo exemplo com as imagens 2D. Ele é similar ao anterior, mas as imagens de treinamento foram modificadas para comprovar nossa hipótese sobre a boa estimação das configurações dos W-operadores. Desta vez, 20 superfícies suaves diferentes foram geradas e somadas a cada uma das 20 imagens anteriores do modelo booleano. Cada função foi gerada escolhendo-se aleatoriamente (usando uma distribuição uniforme entre 0 e 50) quatro níveis de cinza para os cantos da imagem, e interpolando-se uma função entre eles. Depois de somadas às imagens originais, cada imagem resultante será formada por pirâmides pequenas posicionadas em diferentes níveis de cinza. A figura 7.9 mostra uma projeção 3D de parte da imagem da superfície gerada. A figura 7.10 mostra parte de uma imagem original e da respectiva composição com a superfície gerada. A imagem embaçada foi regerada para cada imagem usando o mesmo kernel que antes e os novos pares de imagens (embaçada, ideal) foram usados para projetar novos operadores, como anteriormente. A figura 7.11 mostra uma projeção 3D de parte da imagem composta embaçada. A figura 7.12a mostra parte da imagem embaçada original e a figura 7.12b mostra parte da imagem composta embaçada (projeção 2D da figura 7.11).

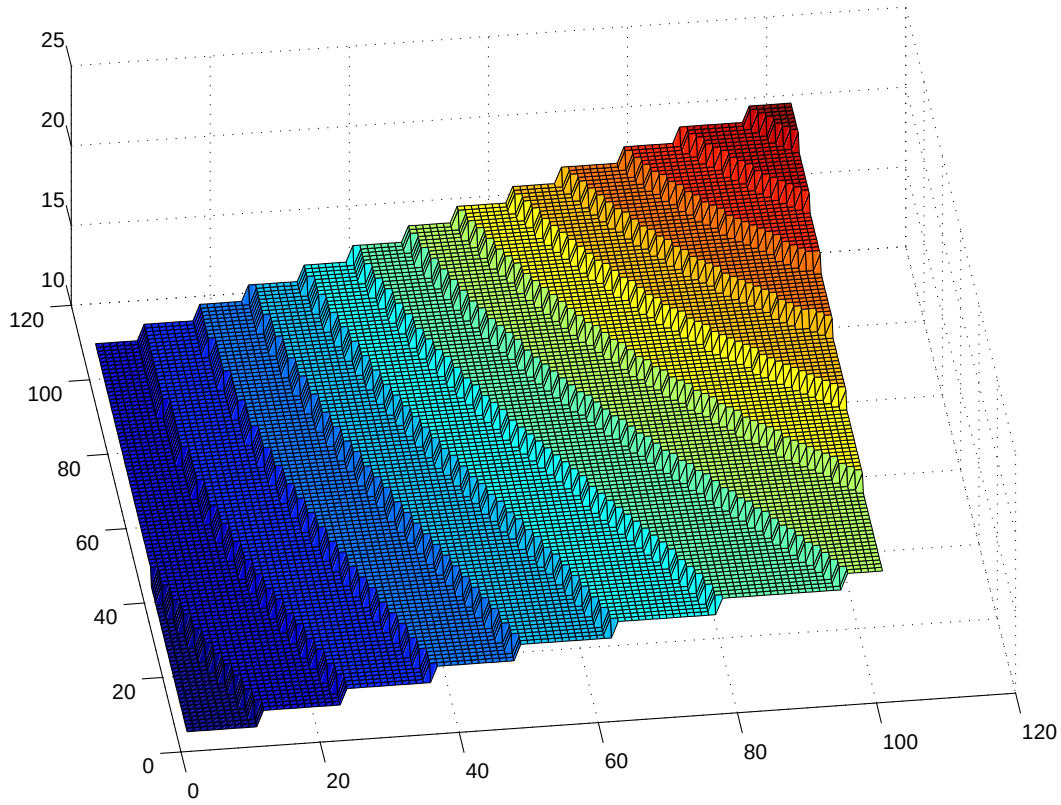
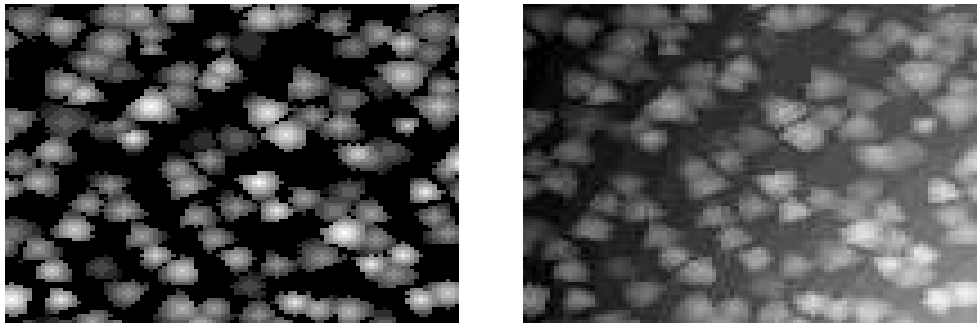


Figura 7.9: Superfície base.

A figura 7.13 mostra o MSE para os melhores operadores projetados em cada classe (linear, "aperture" e "aperture" multiresolução) Cada curva representa um experimento com uma certa restrição. Cada ponto da curva representa o MSE do operador (média sobre 10 imagens de teste)



(a) (b)

Figura 7.10: Modelo booleano: imagem original (a) e função composta (b).

quando treinados usando uma certa quantidade de exemplos de treinamento no projeto.

A curva rotulada "blurred" é o erro médio entre as imagens embaçadas e as imagens ideais. A curva rotulada "Lin 9×9" é o resultado do operador linear ótimo restrito para uma janela de 9×9 pontos, como anteriormente. As curvas "Mres A 6", "Mres A 8", "Mres W 6" e "Mres W 8" mostram o MSE para os "apertures" multiresolução, e W-operadores, respectivamente, projetados usando as pirâmides mostradas nas figuras 7.14a ("Mres A 6" e "Mres W 6"), e b ("Mres A 8" e "Mres W 8"). Cada "aperture" na pirâmide tem a mesma janela nos níveis de cinza, $K = [-5, 5]$. A curva rotulada "Dt_6" mostra o MSE para o operador "aperture" usando uma janela 3×3. A curva rotulada "Dt_{15}" mostra o MSE para o operador "aperture" projetado usando a janela de 17 pontos. Esta curva é mostrada aqui apenas para comparação com o operador multiresolução "Mres A 8".

Como era esperado, a maioria dos filtros "aperture" foram melhores que os lineares, cujo melhor resultado experimentado estabiliza próximo de 0.47. O melhor resultado é obtido pelo operador multiresolução "Mres A 8". O operador "Dt_{15} ("aperture") é pior que todos os multiresolução, mas melhor que todos os W-operadores multiresolução testados. O melhor operador "aperture" não-multiresolução é "Dt_6", que é similar a "Mres A 6".

A performance ruim dos W-operadores é devida ao intervalo grande de níveis de cinza, o que prejudica a estimação, como discutimos anteriormente. O melhor dos W-operadores multiresolução foi o "Mres W 8".

A figura 7.15 mostra o MSE dos operadores multiresolução e não-multiresolução definidos pelas pirâmides mostradas nas figuras 7.14b ("Mres A 8"), 7.16a ("Mres A 9"), para o filtro linear "Lin 9×9" e para o operador "aperture" não-multiresolução "Dt_6". O gráfico mostra o efeito de trocar a janela W_2 sobre a precisão do operador projetado. Em ambos os casos a janela W_2 tem 5 pontos mas, experimentalmente, a cruz diagonal usada no experimento 9 funciona melhor para o modelo de imagens usado aqui.

Um resultado esperado, mas mesmo assim interessante, é mostrado na figura 7.17. O gráfico mostra duas curvas, o MSE para os operadores "aperture" multiresolução e o MSE para os não-multiresolução com o mesmo número de amostras. A pirâmide usada para projetar o operador multiresolução indicado pelos rótulos W_i, $0 \leq i \leq 3$, no gráfico é composta pelas janelas W_i, $W_{i+1}, \ldots, W_4$ mostradas na figura 7.18. Por exemplo, a pirâmide indicada pelo rótulo W_2 é composta pelas janelas W_2, W_3 e W_4 mostradas na figura 7.18. A curva de erro mostra que o melhor operador "aperture" para as amostras de treinamento dadas é a rotulada W_2. O erro aumenta para janelas menores e

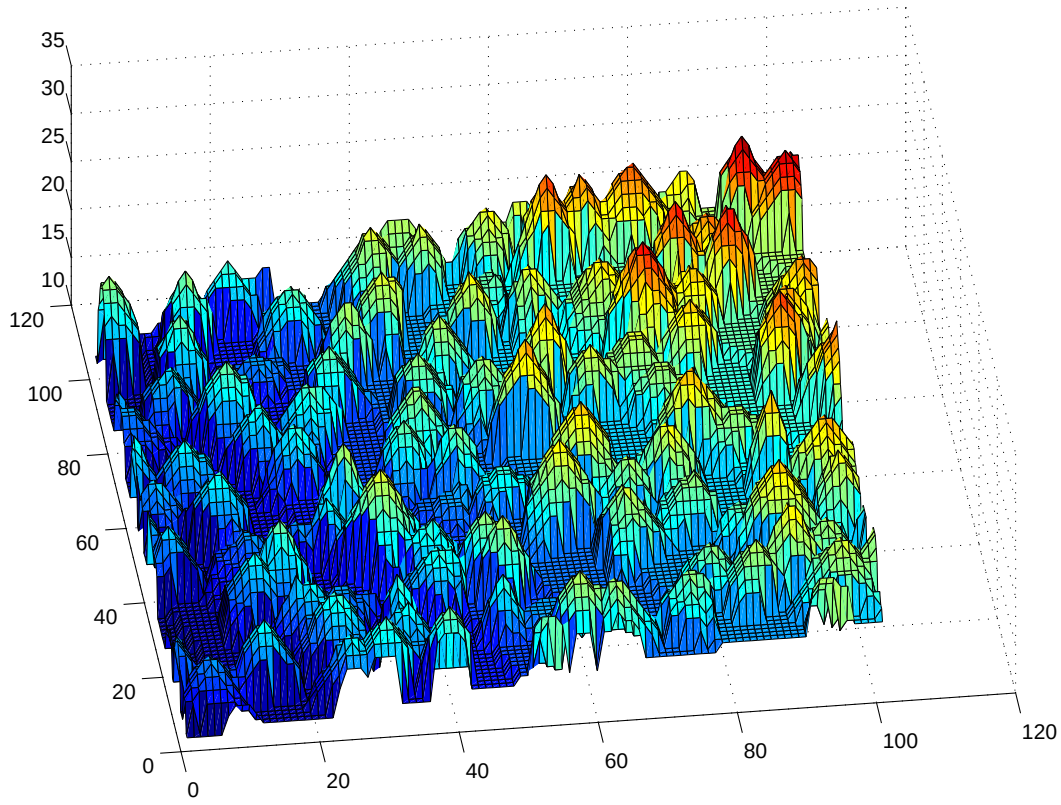
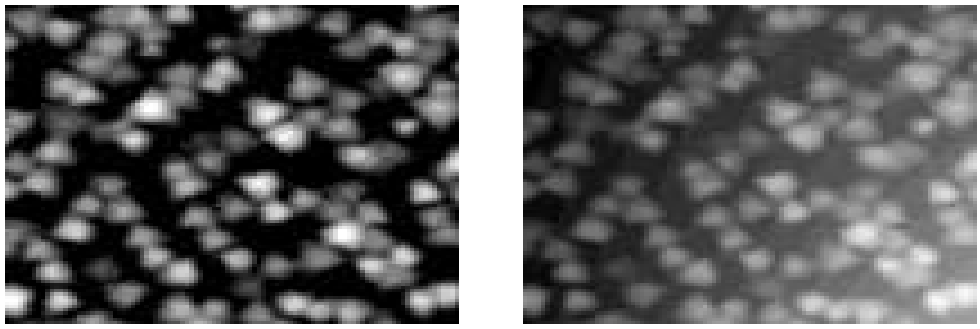


Figura 7.11: Modelo booleano composta com a superfície de base.

maiores. Para os operadores multiresolução, o erro aumenta conforme adicionamos mais níveis na pirâmide, ilustrando as vantagens de usar esta abordagem.

As figuras 7.19 a 7.24 mostram uma pequena região de uma imagem teste, da imagem embaçada e o resultado do melhor filtro para cada classe (linear, "aperture", "aperture" multiresolução). A figura 7.19 mostra uma região de 500 pontos da imagem original e a figura 7.20 mostra a mesma região da imagem embaçada. A figura tem 161 pontos com valores diferentes da imagem original (retângulos com arestas pretas), a maioria deles difere de 1 nível (129 pontos) e por 2 níveis (30 pontos). O MSE para a região é 0.534. A figura 7.21 mostra o resultado do filtro linear projetado sobre uma janela de 9×9 pontos; neste caso o MSE é de 0.434, mas o número de pontos errados não diminui. Este fato deve-se à diminuição dos pontos cuja diferença é maior do que 1 nível. A figura 7.22 mostra o resultado do melhor W-operador multiresolução, cujo MSE é 0.552, portanto maior que o inicial. Isso acontece pois, apesar de haver 130 pontos errados (contra 161 pontos na imagem embaçada) o erro cometido em cada ponto é grande. O número de pontos com diferença de 1 nível diminuiu para 102 e com diferenca maior que 2, 30 pontos. A figura 7.23 mostra o resultado do melhor "aperture" multiresolução (janela de 17 pontos). O MSE e o número de pontos errados cai para 0.342 e 89, respectivamente. O melhor resultado é mostrado na figura 7.24, obtido pelo "aperture" multiresolução cuja primeira janela é a de 17 pontos. O MSE neste caso é 0.222 e o número de pontos errados é 73.



(a) (b)

Figura 7.12: Modelo booleano: imagem embaçada (a) e o embaçamento da imagem composta com a superfície (b)

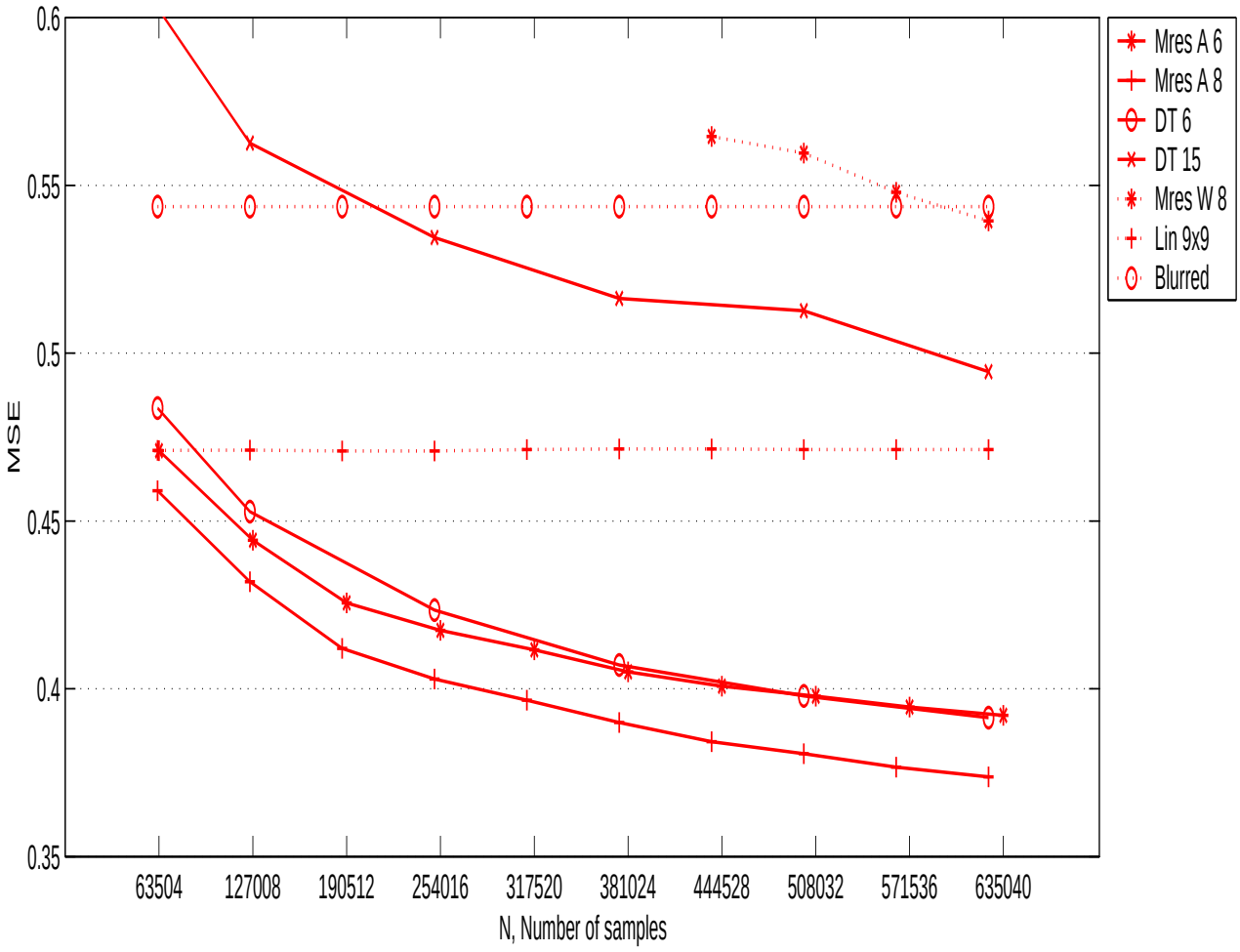


Figura 7.13: Comparação do MSE: intervalo grande

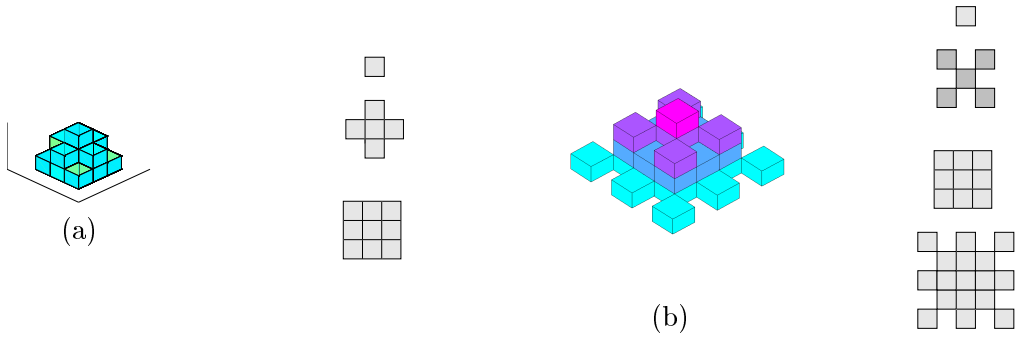


Figura 7.14: Pirâmides para os experimentos “Mres A 6” e “Mres A 8”

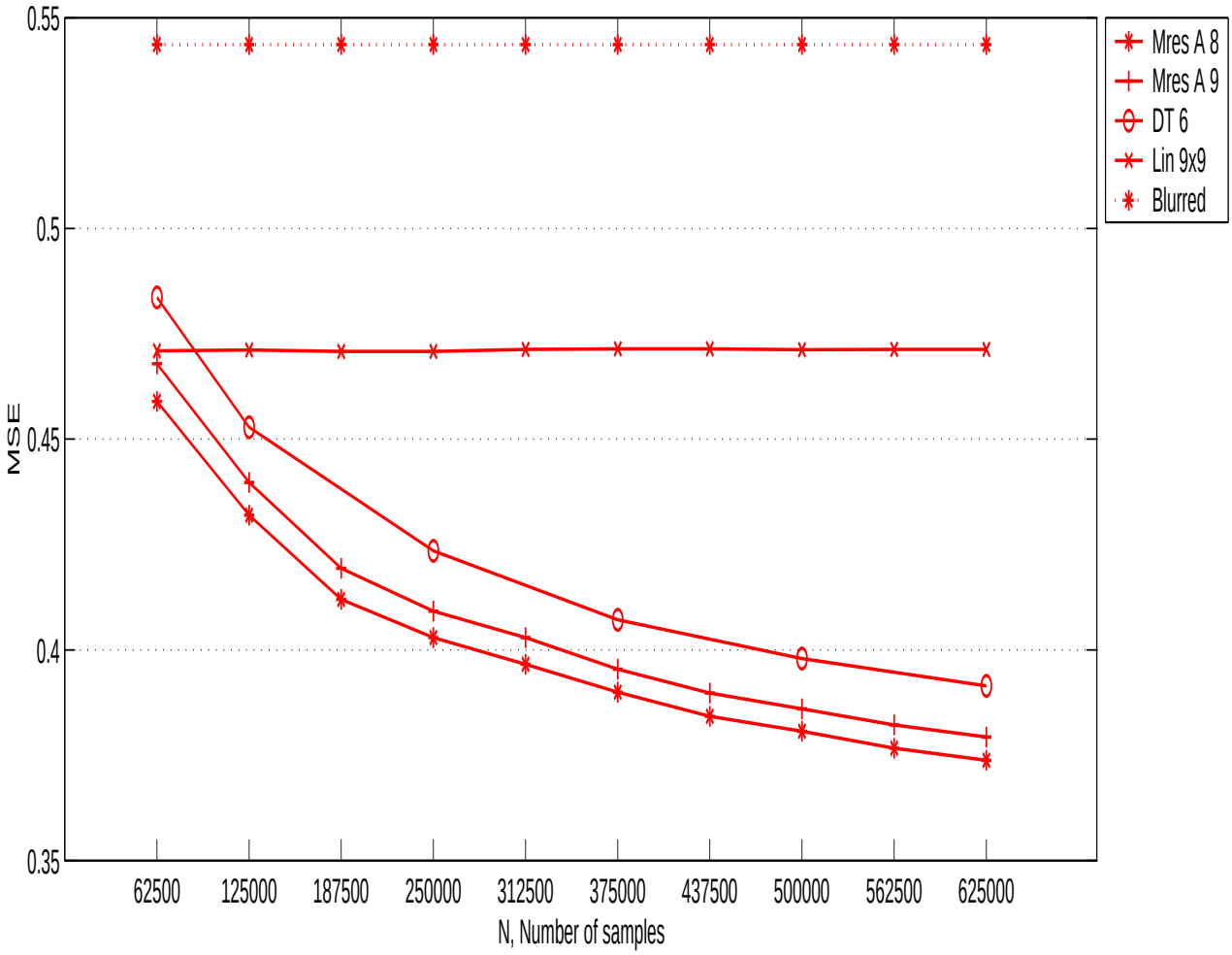


Figura 7.15: Comparação quando mudamos apenas uma janela da pirâmide

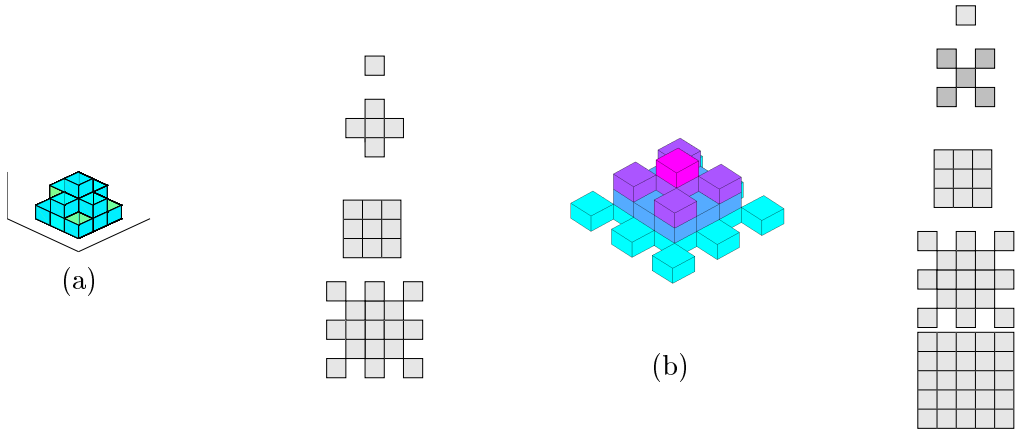


Figura 7.16: Pirâmides para os experimentos "Mres A 9" e "Mres A 11"

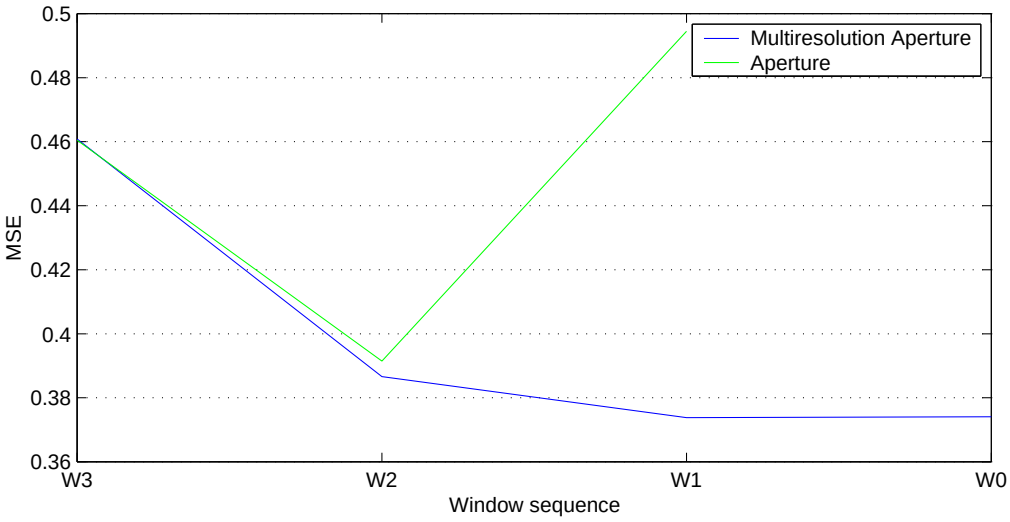


Figura 7.17: Comparação do MSE: multiresolução × não-multiresolução

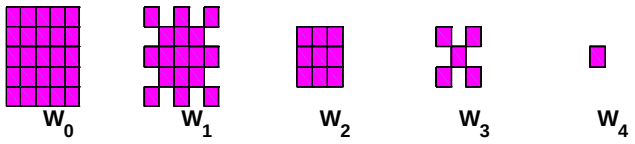


Figura 7.18: Pirâmides: $\{W_3, W_4\}, \{W_2, W_3, W_4\}, \ldots, \{W_0, W_1, W_2, W_3, W_4\}$

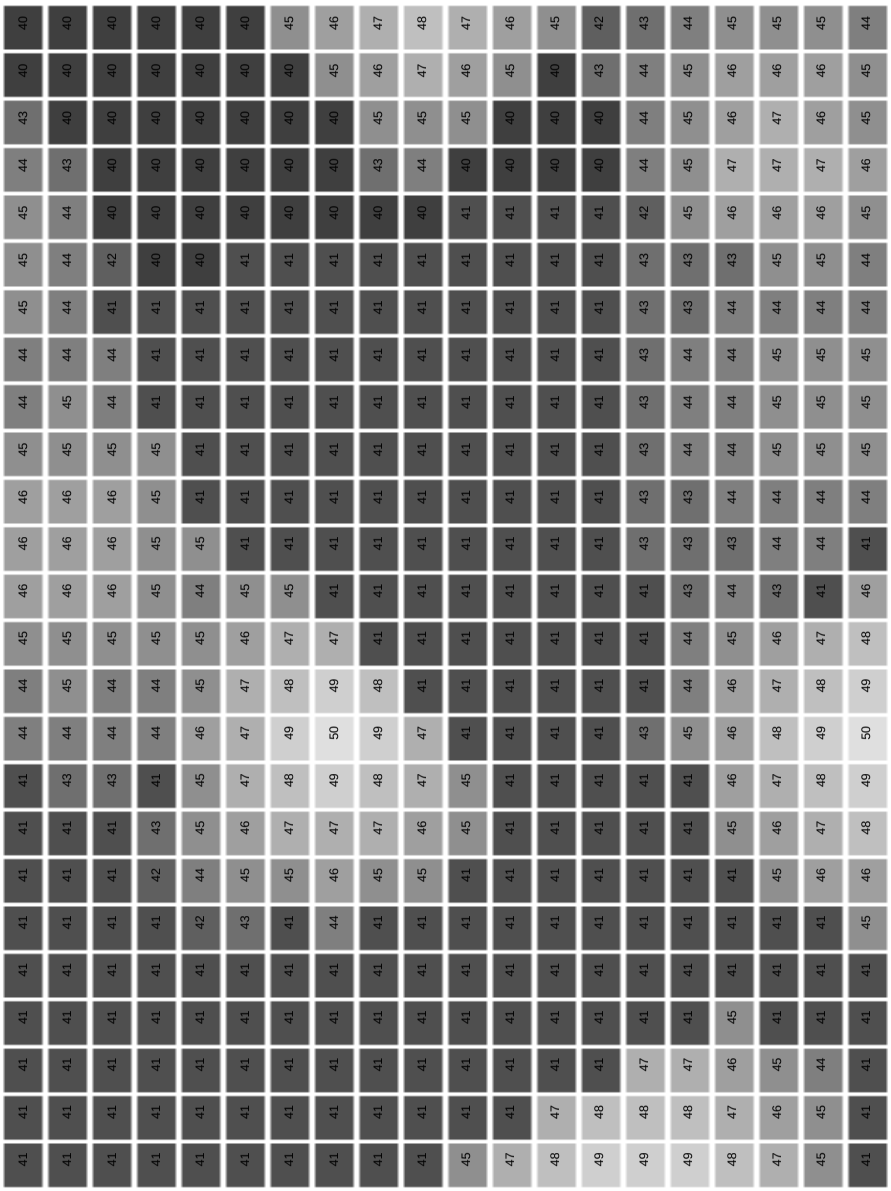


Figura 7.19: Parte da imagem original

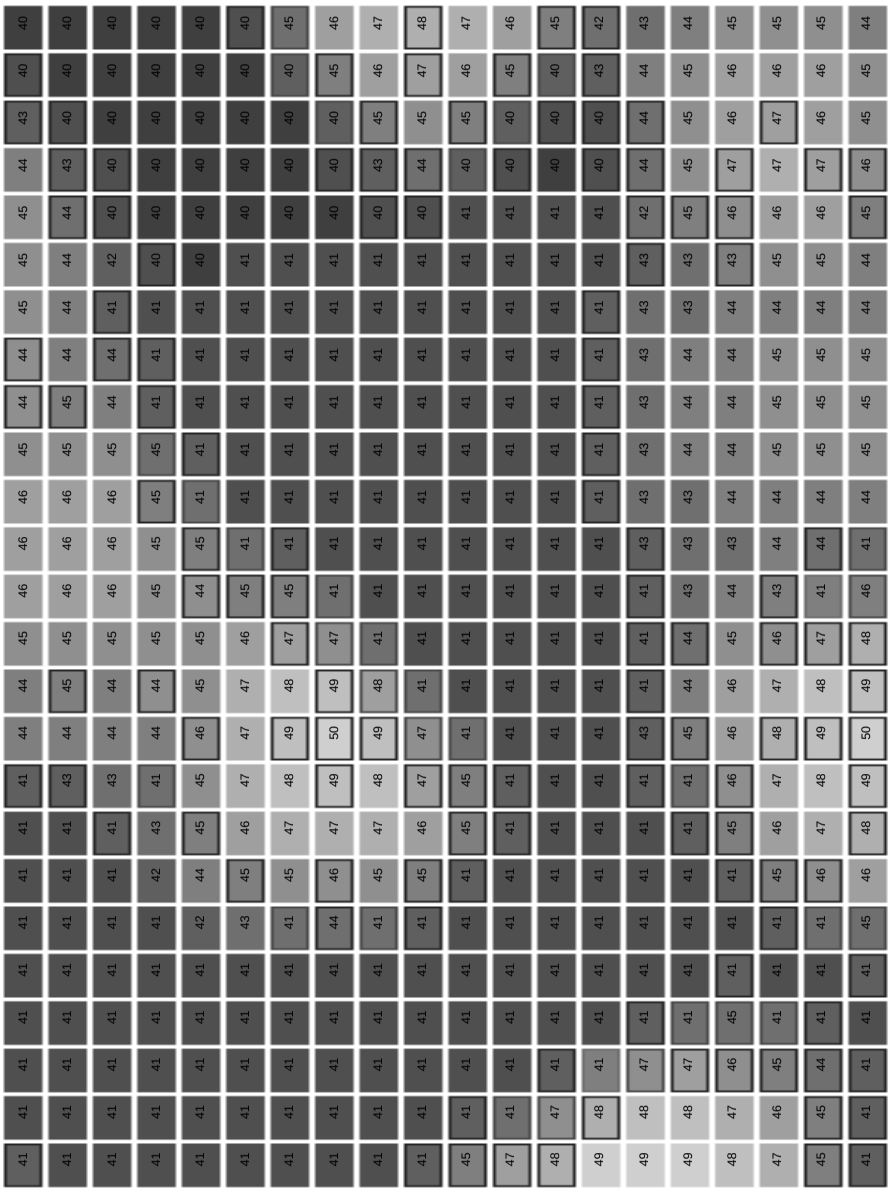


Figura 7.20: Imagem embaçada

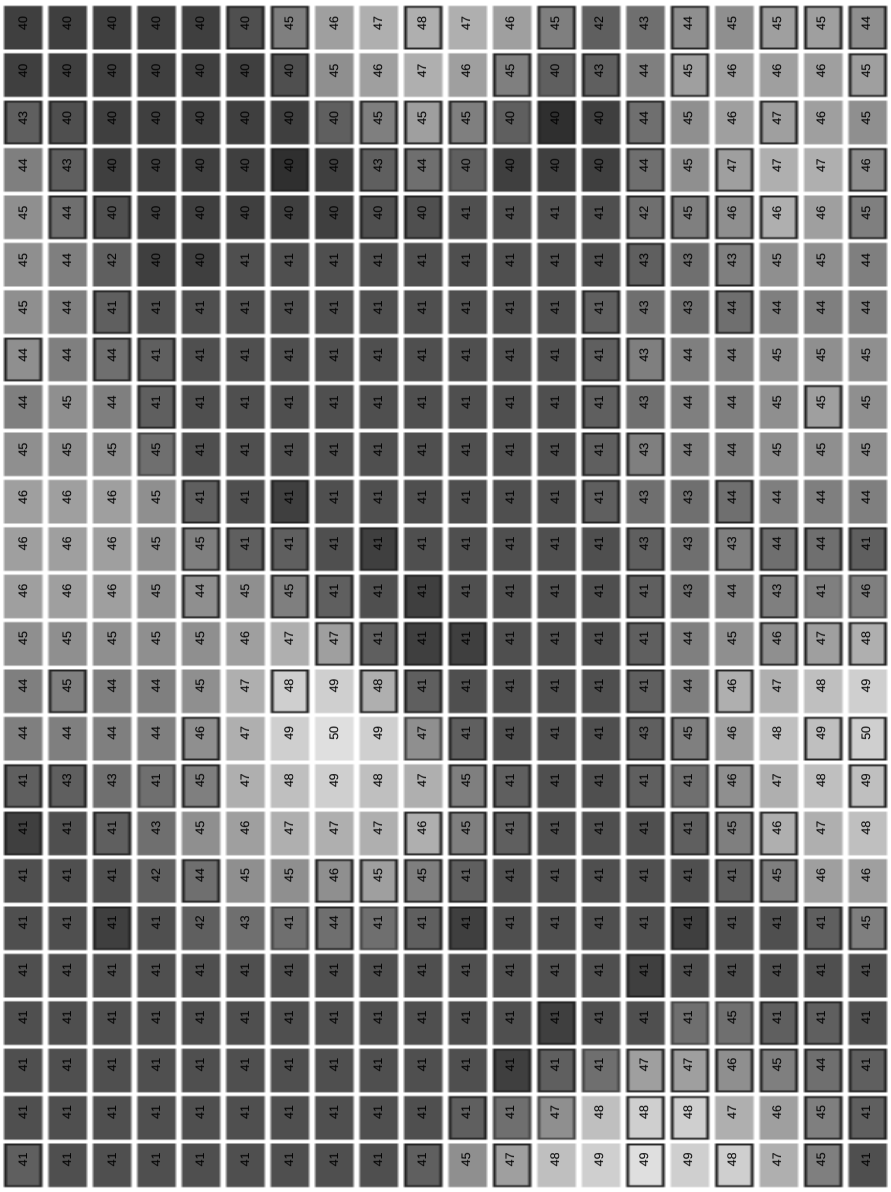


Figura 7.21: Resultado do melhor filtro linear

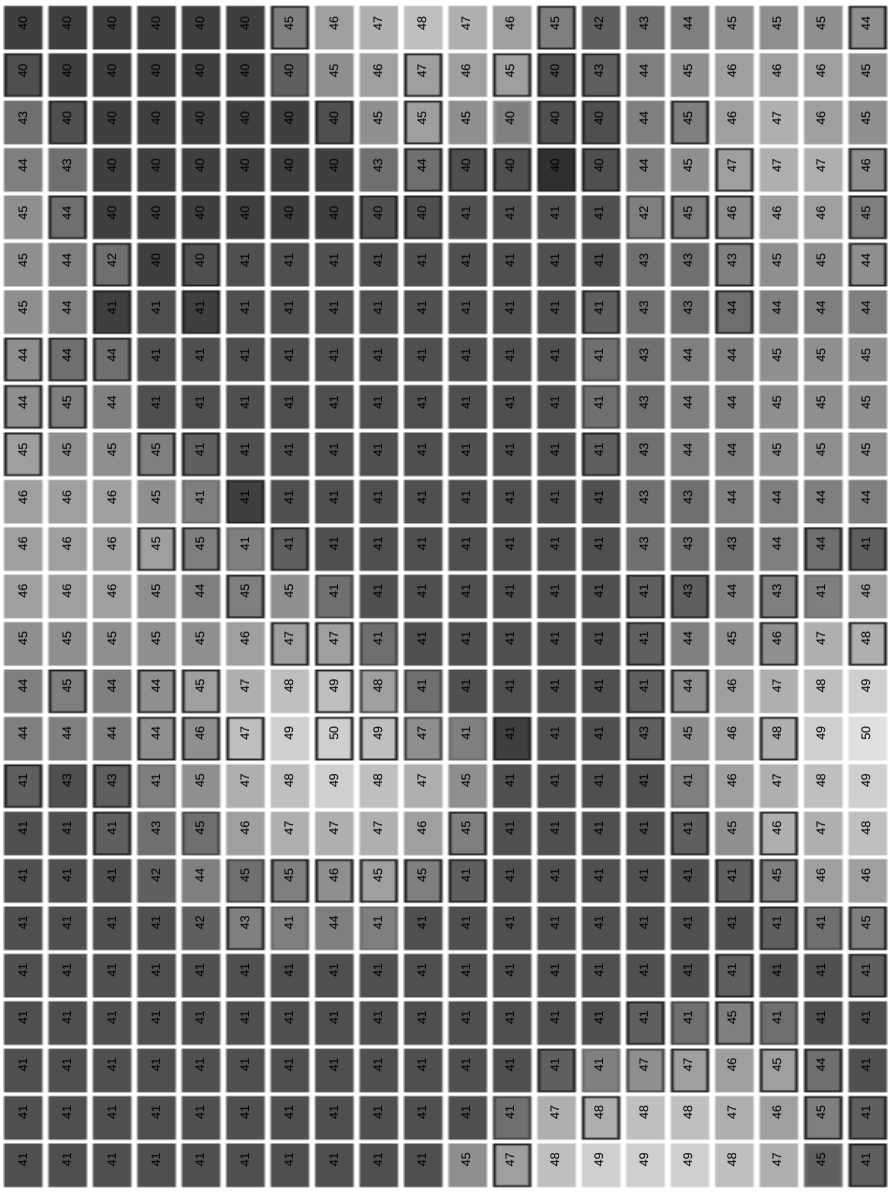


Figura 7.22: Resultado do melhor W-operador multiresolução

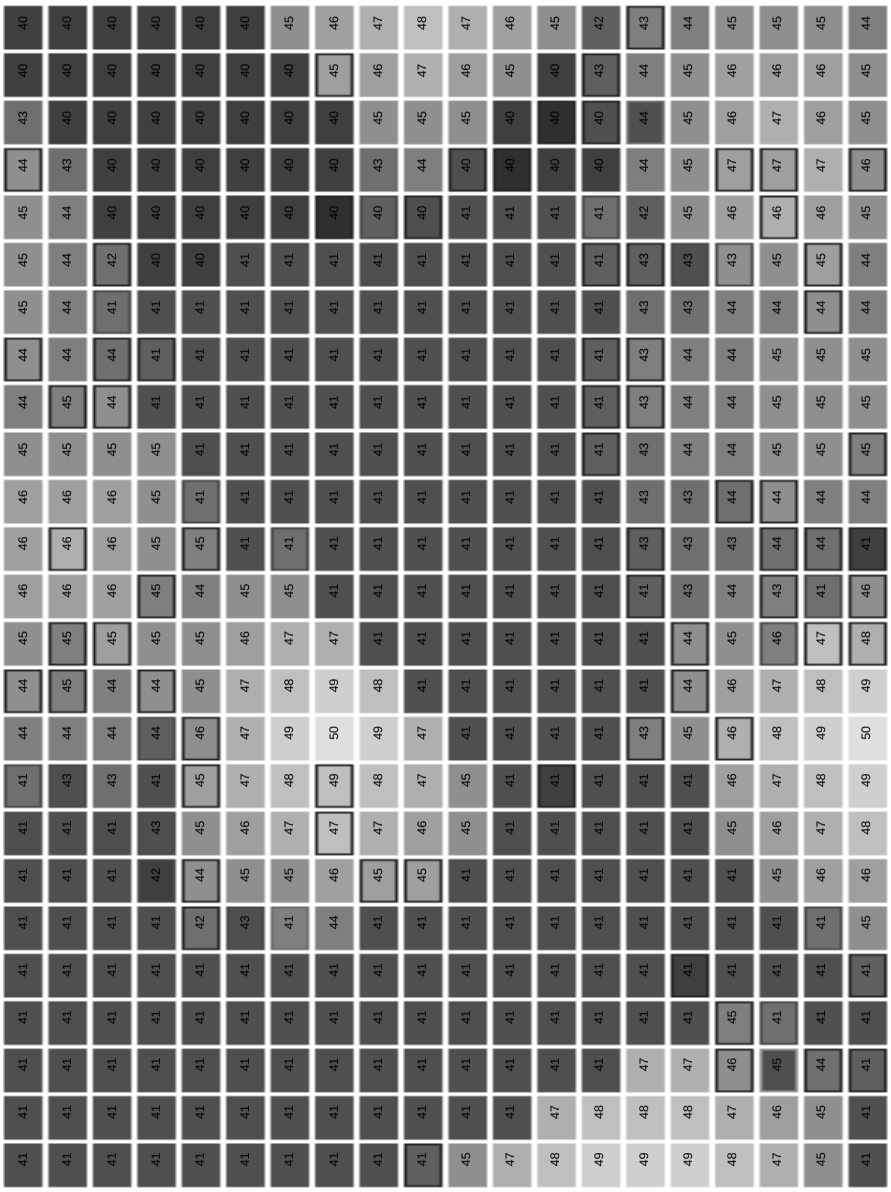


Figura 7.23: Resultado do melhor “aperture”

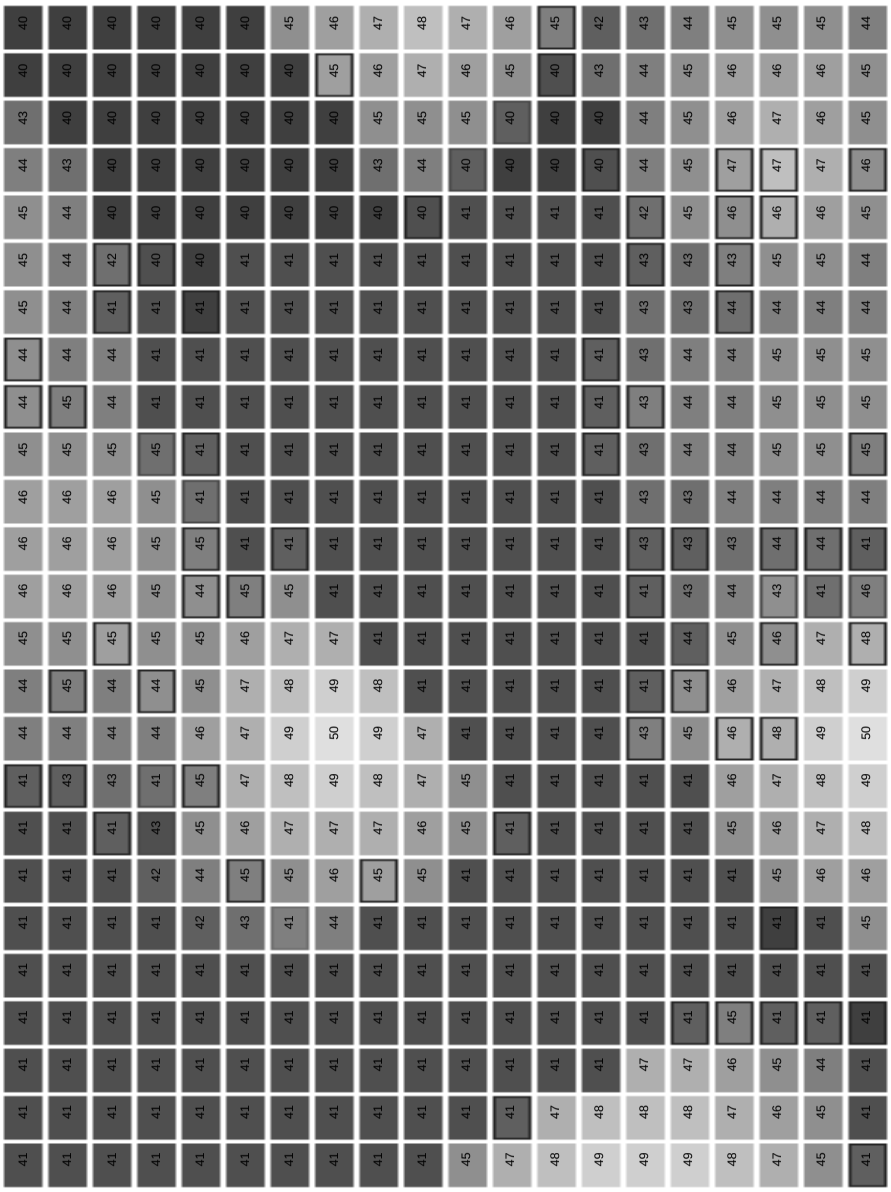


Figura 7.24: Resultado do melhor "aperture" multiresolução

Capítulo 8

Aplicações

Este capítulo apresenta diversas aplicações nas quais o projeto de operadores "aperture" foi usado para resolver problemas reais. O capítulo está dividido em três partes: Na primeira parte, mostramos dois exemplos de uso desses operadores para desembaçamento de imagens de satélites[1]. Na segunda parte, mostramos a aplicação dos operadores "aperture" para aumento de resolução de imagens. Na terceira parte, mostramos a aplicação dos operadores "aperture" como ferramentas auxiliares para encontro de marcadores. Apresentamos também a extensão desses operadores para espaço das imagens coloridas e alguns exemplos de aplicação nesse contexto.

8.1 Desembaçamento da banda 3 de imagens do SPOT

O objetivo desta aplicação é desembaçar uma imagem de satélite de 430×430 pontos (SPOT), degradada via convolução por um núcleo não plano de dimensões 7×7. Este núcleo simula a imagem adquirida pelo satélite CBERS (fruto de uma cooperação entre o Brasil e a China). A figura 8.1 mostra a imagem tirada pelo satélite SPOT e a figura 8.2 mostra a correspondente imagem convoluída.

Foram feitos 35 experimentos usando 7 "apertures" diferentes. As diferentes "apertures" testadas são denotadas por $(W \times k,\ m)$, onde W é a janela espacial e k e m são os limites da janela nos níveis de cinza. Foram testados: $(W \times 5,\ 20)$, $(W \times 5,\ 25)$, $(W \times 5,\ 30)$, $(W \times 10,\ 20)$, $(W \times 10,\ 25)$, $(W \times 10,\ 30)$, $(W \times 15,\ 15)$ e $(W \times 15,\ 30)$; onde W é uma janela de 13 pontos mostrada na figura 8.3.

O posicionamento da "aperture" para todos os experimentos foi no sinal. O núcleo de um operador linear ótimo de janela restrita também foi estimado usando a mesma janela espacial. Como havia apenas uma imagem, usamos um número crescente de pontos aleatoriamente escolhidos da imagem embaçada para treinar o operador (17629, 33566, 48130, 61101, e 72933). O restante da imagem foi usada para testar o operador. A figura 8.4 mostra o resultado do operador "aperture" $(W \times 15,\ 30)$ usando 72933 exemplos.

A figura 8.5 mostra o resultado do operador linear ótimo restrito de janela W os mesmos 72933 exemplos.

Como pode-se ter uma falsa impressão (positiva) dos resultados, é preferível analisar os gráfico de

[1]Agradecemos ao Prof. G. J. F. Banon e ao INPE pelo uso das duas imagens do SPOT neste trabalho.

Figura 8.1: Imagem do SPOT

Figura 8.2: Simulação do CBERS

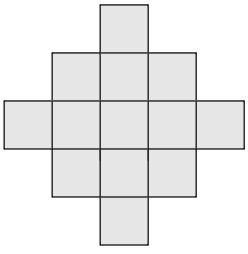


Figura 8.3: Janela de 13 pontos

Figura 8.4: Resultado do operador "aperture" ($W \times 15$, 30)

Figura 8.5: Resultado do operador linear ótimo

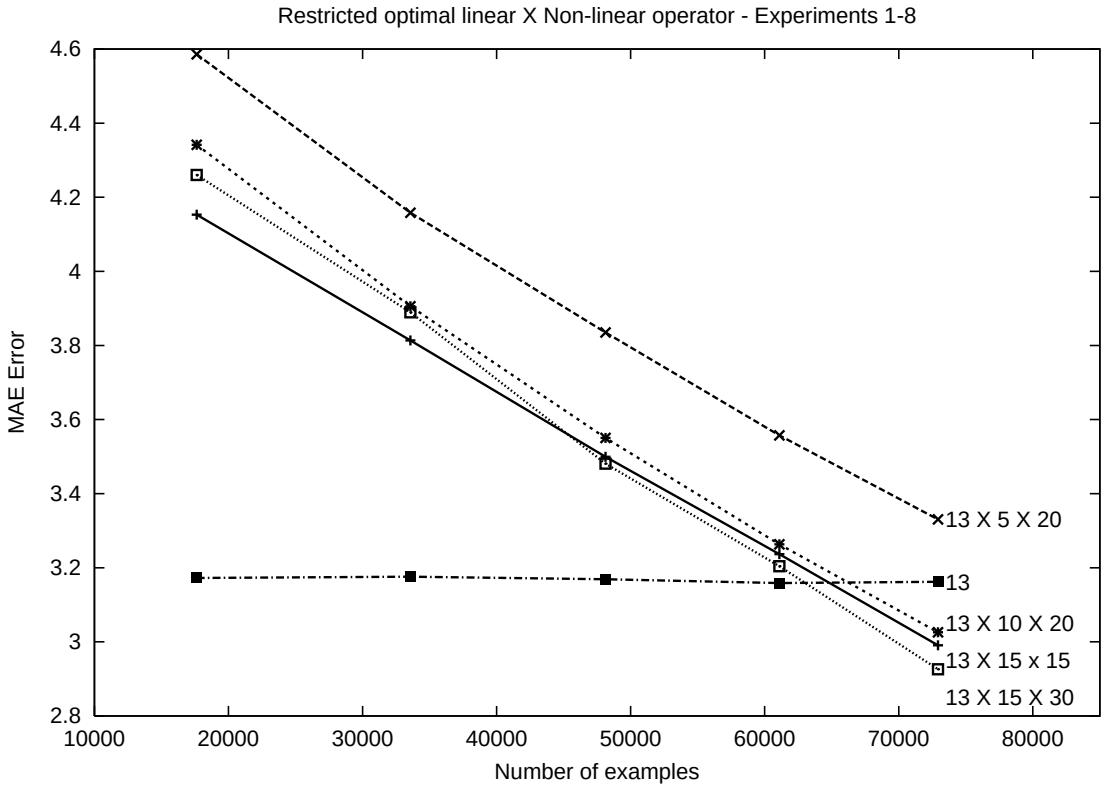


Figura 8.6: MAE dos operadores estimados

erro MAE e MSE para cada um dos Ψ_N. A figura 8.6 mostra o MAE medido entre a imagem resultante da aplicação de cinco operadores estimados (quatro "apertures" e um linear) e a imagem original do SPOT. O erro foi calculado apenas nos pontos de teste (que não foram usados no treinamento). A figura 8.7 mostra o MSE dos mesmos operadores. O MAE entre a imagem embaçada e a não-embaçada (erro inicial) é 5.379 e o MSE é 64.61. Esses pontos não foram mostrados no gráfico para não prejudicar a visualização das outras curvas.

O MAE do operador linear é aproximadamente 3.17[2] e ele é melhor que todos os operadores "aperture" estimados para até 61101 exemplos. Para 72933 exemplos de treinamento, cinco dos operadores "aperture" têm melhor MAE que o operador linear. Para essa quantidade de exemplos de treinamento, os operadores estão ordenados da seguinte maneira (do maior MAE para o menor): $(W \times 5,\ 30)$, $(W \times 5,\ 25)$, $\mathbf{(}W \times 5\mathbf{,\ 20)}$, $\mathbf{(}W \times 10\mathbf{,\ 20)}$, $(W \times 10,\ 25)$, $(W \times 10,\ 30)$, $\mathbf{(}W \times 15\mathbf{,\ 15)}$ e $\mathbf{(}W \times 15\mathbf{,\ 30)}$. No gráfico mostramos apenas os que estão em negrito. Restringir muito a quantidade de níveis de cinza vistos na imagem observada não deu bons resultados, mesmo relaxando bem a quantidade de níveis de cinza da saída (a diferença de MAE entre esses operadores com $k = 5$, para essa quantidade de exemplos de treinamento, não oferece muito subsídio para explicar a inversão observada na ordem dos erros desses operadores em relação à restrição em m).

O MSE do operador linear é aproximadamente 19.04 e, desta vez, ele é melhor que todos os operadores "aperture" estimados para até 72933 exemplos de treinamento. Ordenando os operadores

[2]Como vimos anteriormente, ele não muda muito com o aumento do treinamento.

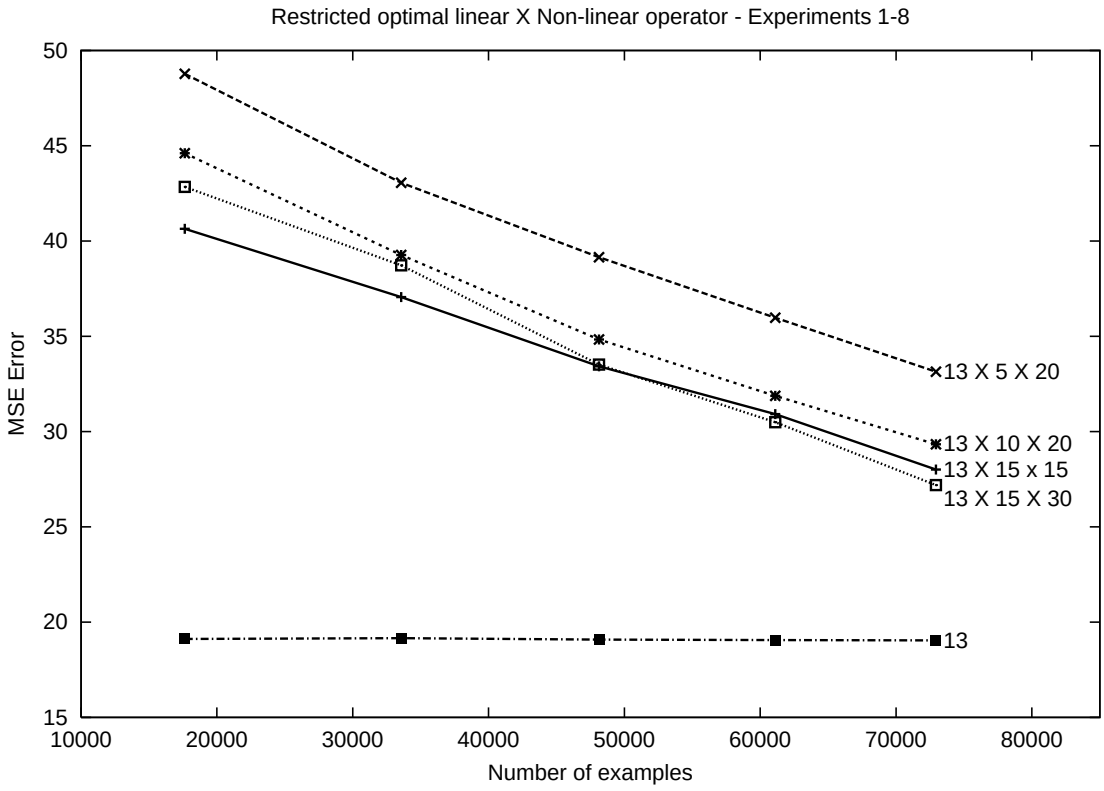


Figura 8.7: MSE dos operadores estimados

de acordo com o MSE para essa quantidade de exemplos de treinamento, obtemos a mesma ordem que anteriormente, ou seja: ($W \times 5$, 20), ($W \times 5$, 25), **($W \times 5$, 20)**, **($W \times 10$, 20)**, ($W \times 10$, 25), ($W \times 10$, 30), **($W \times 15$, 15)** e **($W \times 15$, 30)**. No gráfico mostramos apenas os que estão em negrito.

Há duas prováveis razões para o operador linear ter sido melhor que o operador "aperture": a quantidade de exemplos de treinamento dada para treiná-lo pode não ter sido suficiente, e a restrição nos níveis de cinza pode ter sido demasiada. A segunda razão parece não influenciar tanto pois o erro MSE para o operador "aperture" ($W \times 15$, 30) é apenas ligeiramente melhor que o operador de "aperture" ($W \times 15$, 15). Além disso, sabemos que os operadores lineares têm bom desempenho na correção dos picos da imagem, assim, de certa forma, é esperado que ele fosse mais baixo. Porém, é razoável supor, analisando o gráfico do MSE, que com mais exemplos de treinamento o operador "aperture" resultasse melhor pois a curva de erro está decrescendo em torno de 79000 exemplos.

8.1.1 Desembaçamento da segunda imagem do SPOT

Os resultados do experimento anterior com a imagem do SPOT mostraram que faltaram dados de treinamento para uma melhor avaliação dos operadores "aperture". Por isso, o experimento foi repetido usando uma imagem de 2985×4131 pontos (a resolução de cada ponto é de $20m \times 20m$). A figura 8.8 apresenta a segunda imagem do SPOT e a figura 8.9 apresenta a simulação do CBERS (obtida via convolução) para a imagem apresentada na figura 8.8. Ambas figuras foram normalizadas para melhor visualização. As imagens transformadas encontram-se em:

`http://www.vision.ime.usp.br/demos.html`

Foram feitos 33 experimentos usando 10 "apertures" diferentes ($W \times k$, m), onde W é a janela espacial e k e m os limites da janela nos níveis de cinza: ($W_1 \times 5$, 10), ($W_1 \times 10$, 30), ($W_2 \times 5$, 10), ($W_2 \times 10$, 30), ($W_3 \times 5$, 10), ($W_3 \times 10$, 30), ($W_4 \times 5$, 10), ($W_4 \times 10$, 30), ($W_5 \times 5$, 10), ($W_5 \times 10$, 30); onde W_1 é a janela 3×3, W_2 é a janela 3×3 dilatada por si mesma, W_3 é a janela 3×3 dilatada pela cruz, W_4 é a janela 5×5 e W_5 é a janela 3×3 dilatada duas vezes por ela mesma. O posicionamento da "aperture" para todos os experimentos foi no sinal. O núcleo de um operador linear também foi estimado usando a maior janela espacial, isto é, W_4.

Novamente havia apenas uma imagem (porém, bem maior que a anterior); assim usamos uma máscara onde escolhemos até 14 subimagens de 256×256 pontos e uma subimagem de 169×256 pontos. As subimagens da máscara foram escolhidas visualmente de forma a pegar diversas situações diferentes da imagem. A figura 8.11 ilustra a máscara utilizada. Os buracos em preto indicam quais subimagens foram usadas para o treinamento. O restante da imagem foi usada para testar o operador.

O erro inicial (erro medido sobre a imagem embaçada comparada com a imagem original nos pontos não usados no treinamento) é de 0.85 para o MAE e 4.0 para o MSE. O erro do operador linear é de 2.93 para o MAE e de 14.65 para o MSE.

O melhor operador treinado com até 15 imagens e janela com $m = 10$ foi o de janela W_1 (MAE=0.689, MSE=2.811), seguido pelos de janelas W_2 até W_5 (veja as tabelas 8.1 e 8.2). Novamente temos uma situação invertida para até esse número de exemplos. Aumentando m para 30, temos uma pequena melhora no MAE da janela W_1 com $k = 10$ (MAE = 0.667 e MSE = 1.944), porém o treinamento fica mais difícil computacionalmente e não conseguimos completá-lo para as janelas maiores que a janela W_3 pois cada treinamento estava durando mais que uma semana (o número de exemplos distintos com a janela W_3 para a máscara utilizada foi aproximadamente 670000 e o

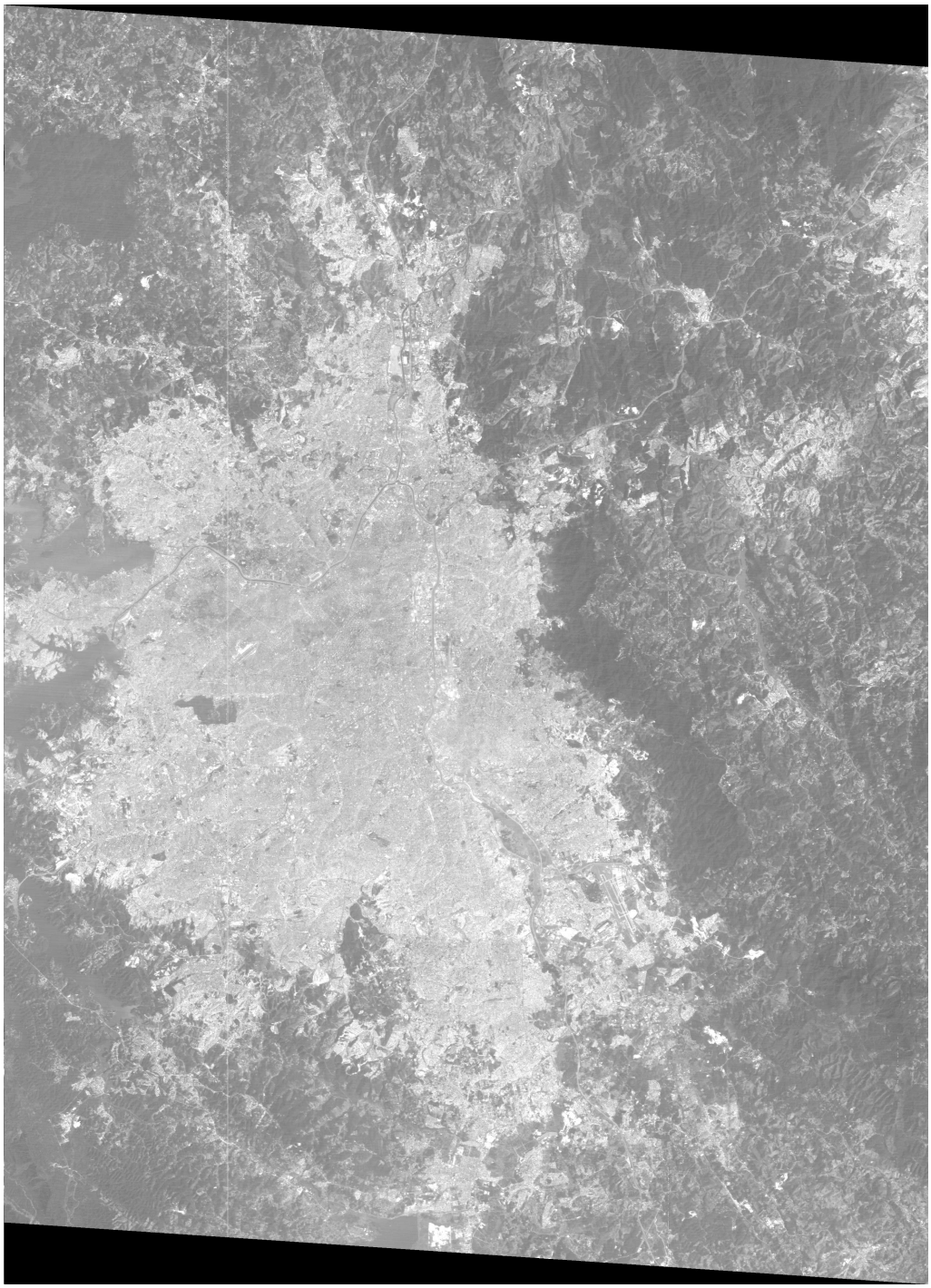


Figura 8.8: Segunda imagem do SPOT

Figura 8.9: Simulação do CBERS para a segunda imagem do SPOT

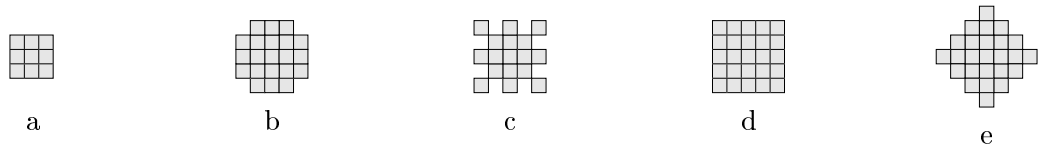


Figura 8.10: W_1 (a), W_2 (b), W_3 (c), W_4 (d), W_5 (e)

Número total de exemplos	194000	388000	582000	776000	970000
($W_4 \times 10$, 30)	0.903316	-	-	-	-
($W_5 \times 10$, 10)	0.898091	0.86184	0.85914	0.863035	0.863036
($W_5 \times 10$, 30)	0.897375	-	-	-	-
($W_4 \times 10$, 10)	0.893088	0.855303	0.849525	0.861773	0.860826
($W_3 \times 10$, 10)	0.880024	0.835442	0.830883	0.835502	0.839969
($W_3 \times 10$, 30)	0.875091	-	-	-	0.821903
($W_2 \times 10$, 10)	0.803851	0.764309	0.761308	0.745893	0.7433
($W_2 \times 10$, 30)	0.800578	-	-	-	0.737202
($W_1 \times 10$, 10)	0.721554	0.704591	0.696803	0.689504	0.689137
($W_1 \times 10$, 30)	0.703122	-	-	-	0.666967

Tabela 8.1: Tabela com MAE dos operadores

treinamento durou 6 dias).

8.2 Mudança de resolução

A mudança de resolução de uma imagem digital é uma aplicação importante para a indústria de impressoras pois os documentos podem ser gerados em uma resolução diferente daquela em que vão ser impressos [6]. Por exemplo, um documento digitalizado em resolução baixa por uma máquina de fax, que chega a um computador pessoal via uma placa de fax-modem e que irá ser impresso em uma impressora laser de 600dpi ("dots per inch"), não pode, ou pelo menos não deveria, ter suas

Número total de exemplos	194000	388000	582000	776000	970000
($W_4 \times 10$, 30)	4.106686	-	-	-	-
($W_3 \times 10$, 10)	3.850904	3.470446	3.420035	3.312936	3.539874
($W_5 \times 10$, 10)	3.920462	3.631362	3.678485	3.631642	3.395599
($W_4 \times 10$, 10)	3.62343	3.579734	3.475789	3.566246	3.385036
($W_2 \times 10$, 10)	3.307937	3.200714	3.202097	3.032141	3.221458
($W_2 \times 10$, 30)	3.431469	-	-	-	2.859283
($W_1 \times 10$, 10)	2.966808	2.916886	2.928555	2.806484	2.811432
($W_3 \times 10$, 30)	3.643239	-	-	-	2.661987
($W_5 \times 10$, 30)	3.624659	-	-	-	-
($W_2 \times 10$, 30)	2.199536	-	-	-	1.944209

Tabela 8.2: Tabela com o MSE dos operadores

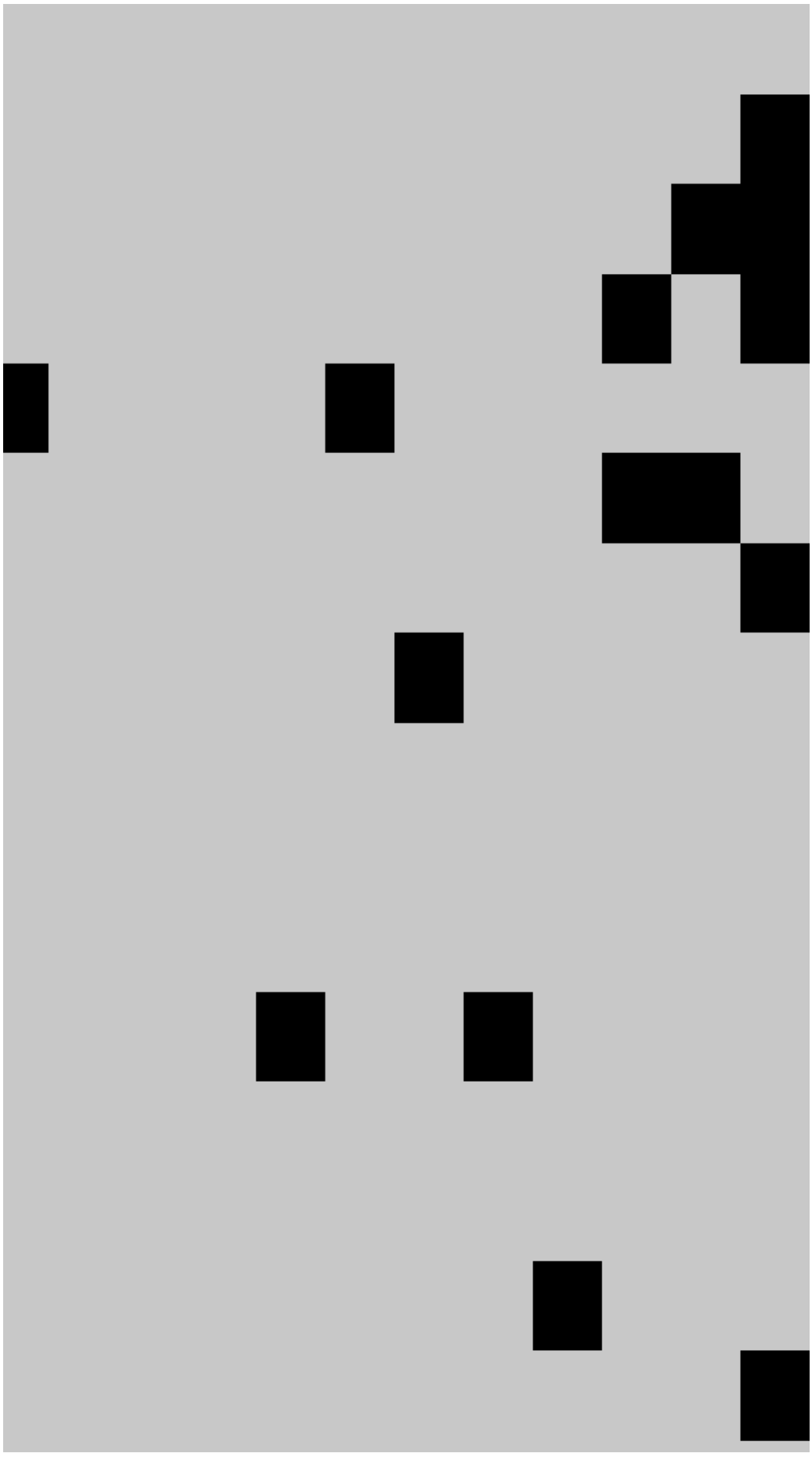


Figura 8.11: Máscara utilizada para restringir o treinamento

Figura 8.12: Imagem de um olho em 75 dpi

letras serrilhadas na impressão. Os programas da impressora devem e precisam adequar a resolução da imagem antes de imprimir.

Até há pouco tempo só existiam impressoras monocromáticas (seja em preto e branco, ou coloridas), isto é, impressoras que pintam, ou não, um ponto no papel[3]. Para essas impressoras, várias técnicas de melhoramento de resolução foram experimentadas e são, normalmente, implementadas fisicamente em "hardware" em função do desempenho [6].

O que motivou o estudo que vamos apresentar nesta seção foi o aparecimento recente de impressoras que podem imprimir mais que um tom de cinza[4], aliado ao fato que os filtros "aperture" deram bons resultados na restauração de arestas nos experimentos de desembaçamento.

O objetivo deste experimento é testar a aplicabilidade dos operadores "aperture", e da técnica de projeto automático, para projetar um operador para mudança de resolução (i.e., interpolação de pontos). Lembremos que, uma vez projetado o operador, ele servirá apenas para uma classe de imagens e um tipo de mudança de resolução. Além disso, um estudo aprofundado sobre o assunto seria necessário caso quiséssemos implementar a técnica industrialmente. No caso deste experimento, vamos partir de uma imagem digitalizada com resolução de 75 dpi e chegar a uma imagem de resolução de 150 dpi. A figura 8.12 mostra uma imagem digitalizada em 75 dpi e a figura 8.13 uma imagem digitalizada em 150 dpi.

Para explicar como projetar o operador nos casos de mudança de resolução inteira, vamos recordar primeiramente como diminuir a resolução de uma imagem. A maneira mais simples de diminuir a resolução original, digamos m dpi, por um fator n tal que $\frac{m}{n}$ é um número inteiro, por exemplo 2, como no caso da mudança de 150 dpi para 75 dpi, é escolher um a cada $n \times n$ pontos da imagem maior e descartar os outros pontos. Se este procedimento é aplicado consistentemente para cada um dos $n \times n$ pontos, então teremos $n \times n$ imagens (fases) de $\frac{m}{n}$ dpi. Estas fases podem ser reagrupadas (seguindo o caminho inverso) sempre que quisermos recompor a imagem de 150 dpi novamente. A figura 8.14 ilustra o procedimento completo.

Para mudar a resolução de uma imagem para outra resolução maior via operadores "aperture", vamos seguir a idéia inversa da diminuição de resolução e projetar um conjunto de operadores cujas entradas são imagens digitalizadas em baixa resolução e cujas saídas são outras imagens de baixa

[3] A impressão de imagens que possuem mais de um nível de cinza eram possíveis graças a uma técnica chamada "dithering" [152, 153]

[4] Na época que foi feito este trabalho, 4 níveis, hoje já existem impressoras que conseguem 256 níveis de cinza para cada cor.

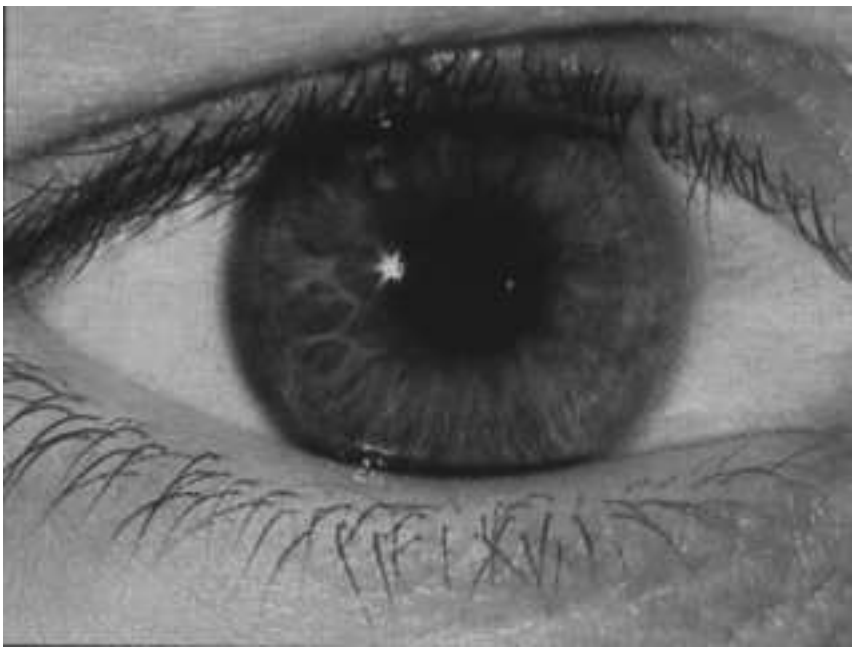


Figura 8.13: Mesma imagem em 150 dpi

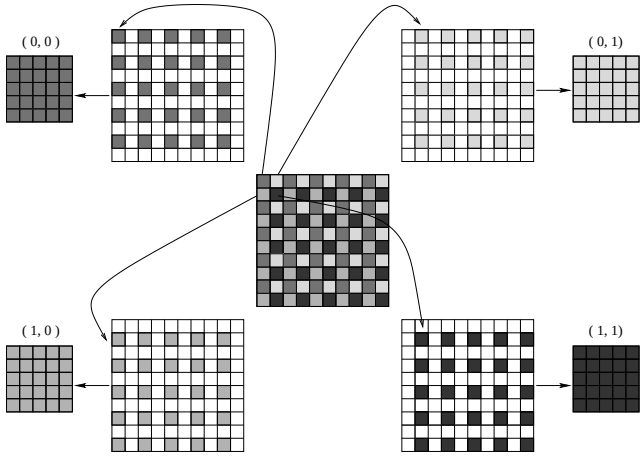


Figura 8.14: Amostragem uma imagem em quatro fases

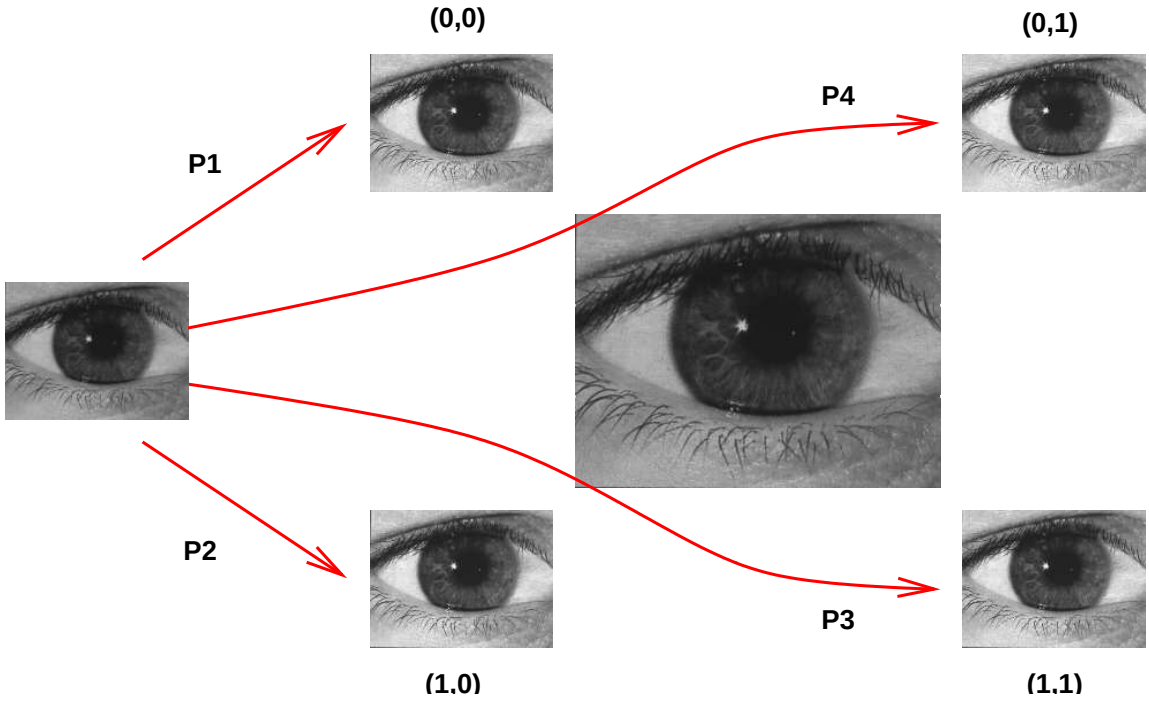


Figura 8.15: Projeto do operador

resolução (que são as fases das imagens de alta resolução). A figura 8.15 ilustra o processo. O projeto desses operadores é feito usando as imagens de baixa resolução como exemplos de treinamento (i.e., a imagem observada é uma imagem digitalizada em baixa resolução) em cada uma das fases da mesma imagem adquirida em uma resolução mais alta (i.e., cada imagem amostrada será a imagem ideal usada para estimar o operador "aperture" para aquela fase). No caso exemplificado, teremos quatro operadores, um para cada fase da imagem.

Cada um dos $n \times n$ operadores projetados transforma a imagem adquirida em baixa resolução (desde que seja similar à usada no treinamento) numa das $n \times n$ fases da imagem de alta resolução. Se o treinamento for feito com um número suficiente de imagens, a aplicação do operador de imagens em imagens semelhantes às usadas no treinamento deveria resultar em imagens semelhantes às que esperaríamos das fases da imagem se elas tivessem sido adquiridas em alta resolução. O processo termina reagrupando-se as imagens para montar uma imagem de alta resolução (seguindo as flechas da figura 8.14 ao contrário). Como as diferenças entre a imagem observada e as fases da imagem ideal normalmente não são muito grandes, os operadores "aperture" podem dar um bom resultado.

Como a razão entre as resoluções usadas é inteira, o método é chamado de *conversão inteira*. Uma idéia similar pode ser usada para conversões não inteiras [6], por exemplo, de 80 para 120 dpi. Porém, não tentamos nada neste sentido.

As figuras 8.16 e 8.17 mostram as fases da imagem da figura 8.13. As imagens são rotuladas pela origem escolhida onde a amostragem começa. Quatro operadores "aperture", Ψ_1, Ψ_2, Ψ_3 e Ψ_4, foram projetados usando a metade direita da imagem de 75 dpi (8.12) para o treinamento. O erro foi medido usando a imagem toda. Dez combinações de janelas foram testadas ($W \times k$, m): ($W_1 \times 10$, 10), ($W_1 \times 10$, 15), ($W_1 \times 10$, 20), ($W_1 \times 10$, 25), ($W_1 \times 10$, 30), ($W_2 \times 5$, 15), ($W_2 \times 5$, 20), ($W_2 \times 5$,

a

b

Figura 8.16: Fases (0,0)(a) e (0,1)(b) da imagem de 150 dpi

a

b

Figura 8.17: Fases (1,0)(a) e (1,1)(b) da imagem de 150 dpi

25), $(W_2 \times 5,\ 30)$, e $(W_2 \times 10,\ 30)$, onde W_1 é a janela 3×3 e W_2 é a janela de 13 pontos (cruz dilatada pela cruz) definida anteriormente.

O resultado reagrupado dos operadores projetados foram comparados com a imagem original, e também com dois algoritmos clássicos de aumento de resolução (replicação simples e média dos vizinhos mais próximos). A tabela 8.3 mostra o resultado dos operadores projetados e dos algoritmos clássicos. A primeira coluna mostra o operador, a segunda o MAE e a terceira o MSE dos pontos diferentes entre a parte de teste da imagem reagrupada e a imagem original.

O resultado dos métodos clássicos é muito ruim, tanto se compararmos o MAE, MSE, e o número de pontos diferentes, ou se compararmos visualmente as imagens resultantes. A imagem resultante de um dos operadores "aperture" $(W_1 \times 10,\ 25)$ é mostrada na figura 8.18. A figura 8.19 mostra o resultado do método de replicação simples e a figura 8.20 mostra o resultado da replicação bilinear. As principais diferenças, em termos visuais, entre os métodos estão nas texturas (elas são melhor reconstruídas pelo operador "aperture") e nas arestas pois os métodos lineares tendem a deixar as arestas embaçadas.

As figuras 8.21a, 8.21b, 8.22a e 8.22b mostram uma parte da imagem aumentada para a imagem original, resultado do operador "aperture", resultado da replicação simples e resultado da replicação bilinear.

As figuras 8.23a, 8.23b, 8.24a e 8.24b mostram outra parte da imagem aumentada para a

Método	MAE	MSE
Linear	5.973623	89.436554
Bilinear	5.270745	62.238235
($W_2 \times 5$, 15)	3.399889	49.103142
($W_2 \times 5$, 20)	3.386958	48.717712
($W_2 \times 5$, 25)	3.379018	49.683300
($W_2 \times 5$, 30)	3.368771	49.590046
($W_1 \times 10$, 10)	3.395259	48.861778
($W_1 \times 10$, 15)	3.185887	44.557217
($W_1 \times 10$, 20)	3.094494	42.284954
($W_2 \times 10$, 30)	3.083440	43.477432
($W_1 \times 10$, 30)	3.081368	43.083649
($W_1 \times 10$, 25)	3.076085	42.380814

Tabela 8.3: Tabela com os erros dos operadores

Figura 8.18: Resultado do operador "aperture" ($W_1 \times 10$, 25)

Figura 8.19: Resultado da replicação linear

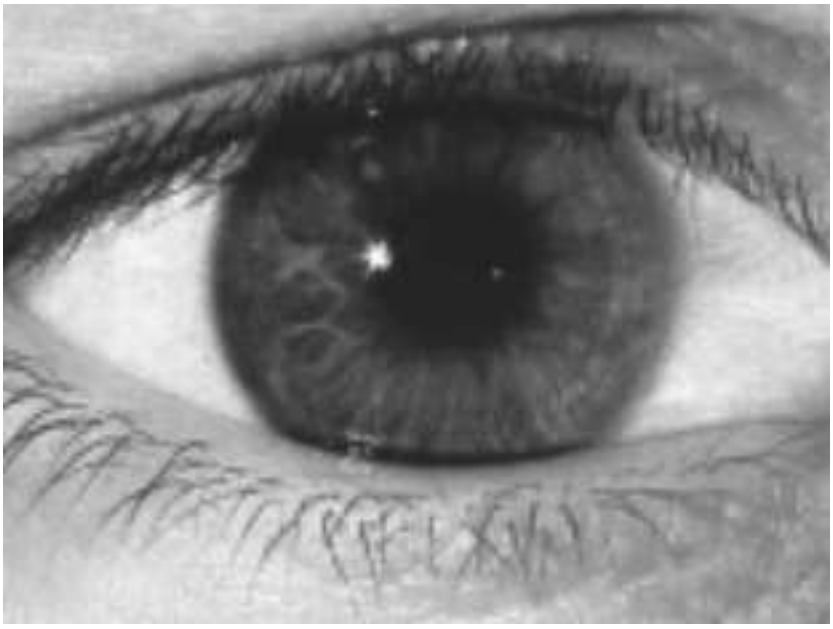
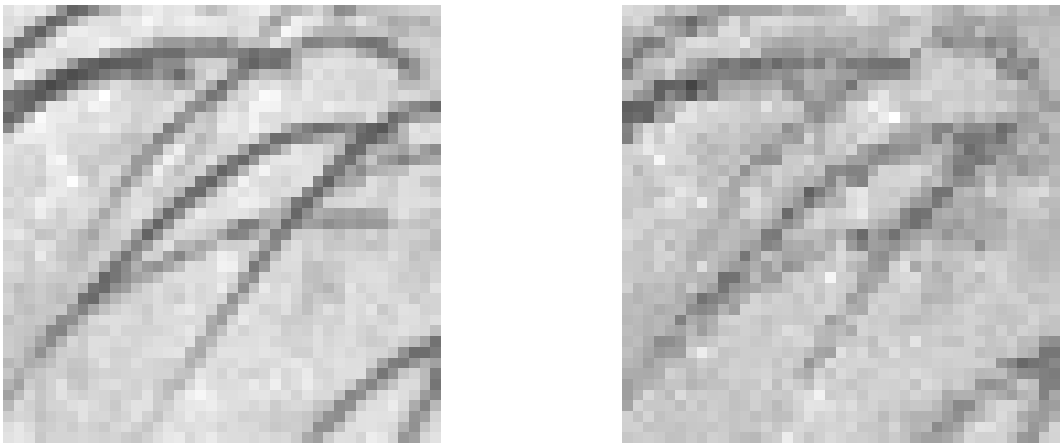
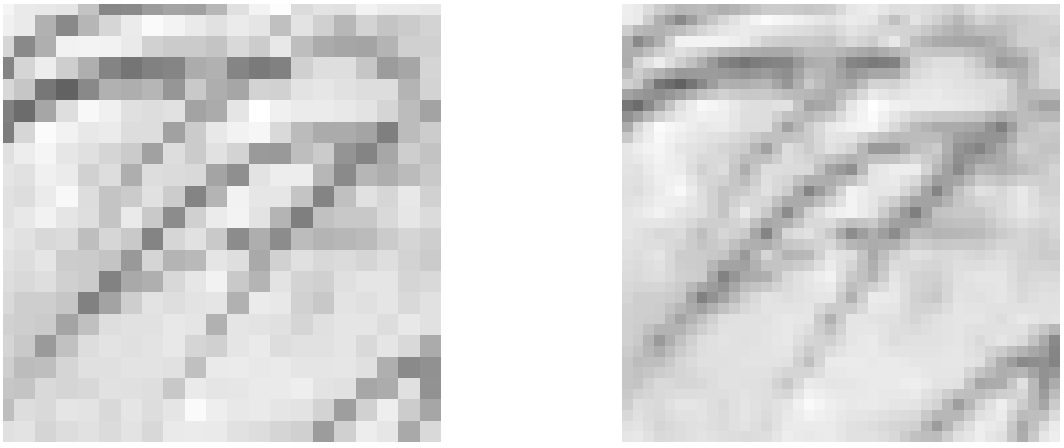


Figura 8.20: Resultado da replicação bilinear



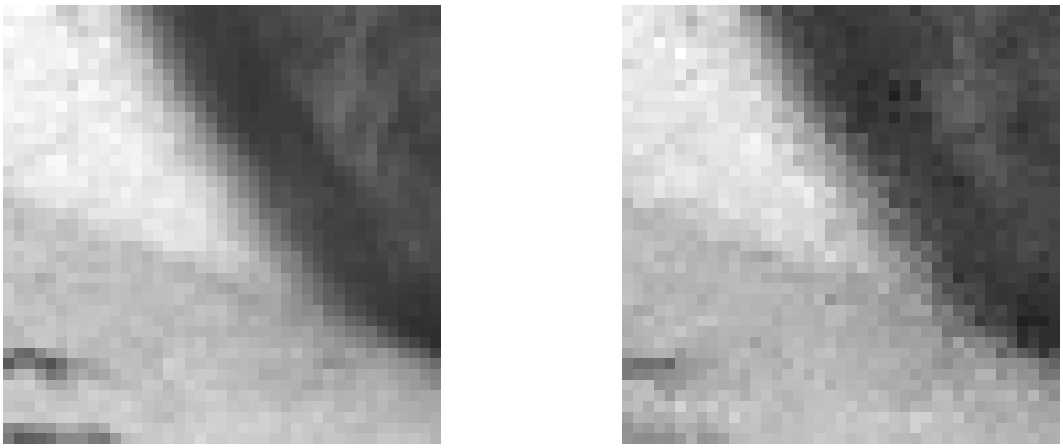
a b

Figura 8.21: Cílios - (a) imagem original e (b) operador "aperture" ($W_1 \times 10, 25$)



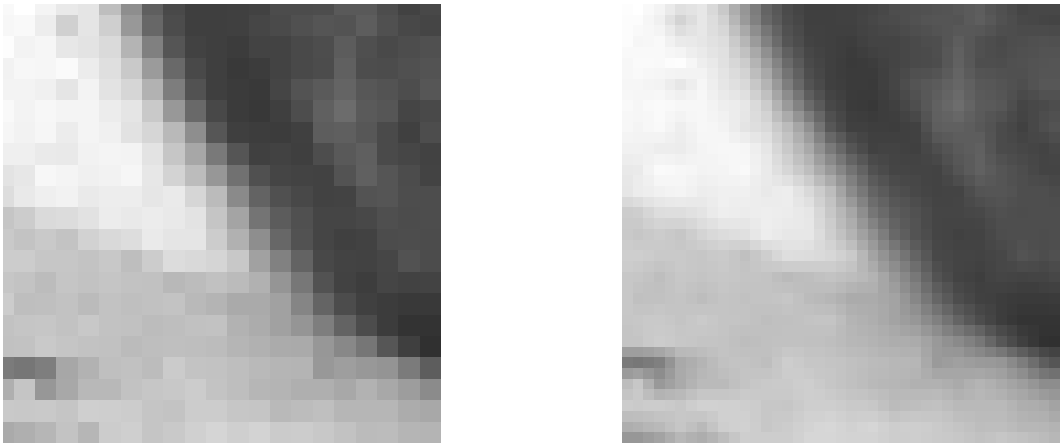
a b

Figura 8.22: Cílios - (a) replicação linear e (b) replicação bilinear



a b

Figura 8.23: Olho - (a) imagem original e (b) operador "aperture" ($W_1 \times 10, 25$)



a b

Figura 8.24: Olho - (a) replicação linear e (b) replicação bilinear

imagem original, resultado do operador "aperture", resultado da replicação simples e resultado da replicação bilinear.

8.3 Segmentação de seqüências de imagens

Um *vídeo digital* é uma seqüência de imagens digitais cuja ordem é definida pelos instantes em que as imagens foram adquiridas, por exemplo, a imagem 1 de uma seqüência é tomada no instante t_0, a imagem 2 no instante $t_0 + \Delta t$, a imagem 3 no instante $t_0 + 2\Delta t$, e assim por diante. A forma como elas são adquiridas, seja via câmeras digitais, ou via digitalização de sinais de TV, ou etc, não é assunto deste trabalho; vamos partir do fato que já temos um vídeo digital e um problema de segmentação a ser resolvido.

A tecnologia dos vídeos digitais está cada vez mais presente em produções profissionais pois os custos são mais baixos e, principalmente, por abrir um amplo espectro de novas possibilidades em termos de criação e produção. Por exemplo, as tarefas de edição de vídeo podem ser feitas usando ferramentas computacionais com muito menos esforço. Além disso, abre-se a possibilidade de novas técnicas de edição como. por exemplo, a *mistura* de diferentes imagens em uma só, a *substituição* de alguns objetos da imagem por outros, a deformação da imagem e etc.

Quando pensamos em substituição de objetos num vídeo, a técnica mais conhecida e usada é o *"chroma key"* [154]. Como a palavra indica, a técnica é baseada na substituição de partes de imagens que foram cobertas com uma máscara colorida (usualmente um tecido azul ou verde) por partes de imagens de outros vídeos. Um cuidado adicional nestes casos é que as cores das partes que serão substituídas não devem ser iguais ao da máscara pois senão podem ser confundidos com as partes que serão substituídas. A figura 8.25 ilustra um exemplo simples de "chroma key".

(a) (b)

Figura 8.25: Exemplo da técnica de "chroma key"

Quando não se usava computadores, o "chroma key" era feito analógicamente, baseado em filtros eletrônicos. Hoje em dia, em se optando pelo uso de vídeos digitais, pode-se usar técnicas de processamento de imagens para a substituição das máscaras de "chroma key". As técnicas de reconhecimento de padrões clássicas, usando as cores e intensidades dos pontos da imagem como atributos são simples e dão bons resultados para a substituição das máscaras.

Uma tarefa mais difícil é a introdução de partes de outros vídeos no lugar de objetos não cobertos pela máscara. Esta tarefa depende essencialmente de segmentar os objetos que serão substituídos em uma seqüência de imagens e substitui-los por objetos de outras seqüências. Automatizar a segmentação não é simples pois requer ferramentas computacionais que consigam rastrear e segmentar corretamente objetos, o que é muito difícil pois a segmentação é um problema mal posto [155].

Quando iniciamos este trabalho, não havia um produto comercial conhecido que fizesse bem esta tarefa e a literatura sobre a área era restrita. Essa alternativa era então normalmente evitada pelos produtores de vídeo, a menos que fosse inevitável; caso em que a solução manual era usada. Porém, a experiência com segmentação usando morfologia matemática e o sucesso inicial que obtivemos com os operadores "aperture" nos motivaram a abordar o problema.

A solução que implementamos é baseada no paradigma de Beucher-Meyer (veja o apêndice para uma introdução curta à segmentação de imagens morfológica), onde os marcadores são detectado pelos operadores "aperture". Estes, por sua vez, são projetados a partir de alguns pares de imagens que refletem as diferentes situações que os objetos aparecem no vídeo. Este último requisito é importante para garantirmos que os marcadores serão bem localizados em todos os quadros da seqüência.

Para simplificar a tarefa de rastreamento, o usuário tem que localizar e marcar manualmente, na primeira imagem da seqüência, os objetos que se deseja segmentar. Se eles não aparecem juntos na mesma imagem, então o usuário tem que marcar cada objeto separadamente em cada um dos quadros que o objeto aparece pela primeira vez. A partir dessa segmentação, e da informação da velocidade aproximada de cada objeto a ser rastreado, o sistema continua a localização e segmentação do(s) objeto(s) nas outras imagens até que o(s) objeto(s) desapareça da cena, ou que o operador "aperture" não consiga marcar o objeto (por falha no treinamento ou porquê as condições de luminosidade, distância e etc, não são as mesmas do treinamento). No caso que um objeto que está sendo perseguido desapareça da cena e venha a aparecer posteriormente de novo, o usuário deve marcá-lo manualmente mais uma vez.

Uma condição para a aplicação desta técnica é que as imagens de treinamento sejam estatisticamente similares às outras imagens da seqüência na região onde os objetos aparecem. Quando há mudanças devido à iluminação, resolução, posição da câmera e etc, um novo treinamento pode ser necessário para os próximas imagens da seqüência. Uma alternativa que também pode ser usada para simplificar o treinamento é a filtragem da seqüência com o uso de filtros conexos. Isso é útil, principalmente, quando estamos segmentando objetos com texturas complexas. Porém, o cuidado de não misturar padrões de objetos diferentes tem que ser tomado. Em outras palavras, se após a filtragem a maioria dos padrões "aperture" do objeto de interesse já não forem mais discriminantes, então os parâmetros do filtro conexo têm que ser reajustados. Além disso, o filtro tem que ser robusto, isto é, servir para toda a seqüência, ou pelo menos para a maior parte dela.

8.3.1 Método

O método proposto é simples pois um dos objetivos do projeto automático de operadores é que ele possa ser feito por não especialistas em processamento de imagens. Os passos que devem ser seguidos são apresentados na figura 8.26 e descritos abaixo:

1. Treinar um operador Ψ dando pares de imagens, $\{f_i, g_i\}$, onde f_i é a i-ésima imagem da seqüência (original, ou após a filtragem conexa) e g_i é a correspondente imagem com os objetos marcados, ou até segmentados. Estes pares devem refletir estatísticamente as diferentes situações que o objeto de interesse aparece. Isto significa que os índices i não formam necessariamente uma seqüência com passo uniforme, como $\{f_i, g_i\}$, $\{f_{i+1}, g_{i+1}\}$, etc. Além disso, como o interesse é segmentar objetos que estão sendo rastreados e não a imagem toda, é natural que o operador não precise ser treinado usando toda a imagem, mas apenas uma certa vizinhança do objeto de interesse. Essa vizinhança é definida por uma imagem máscara m_i que

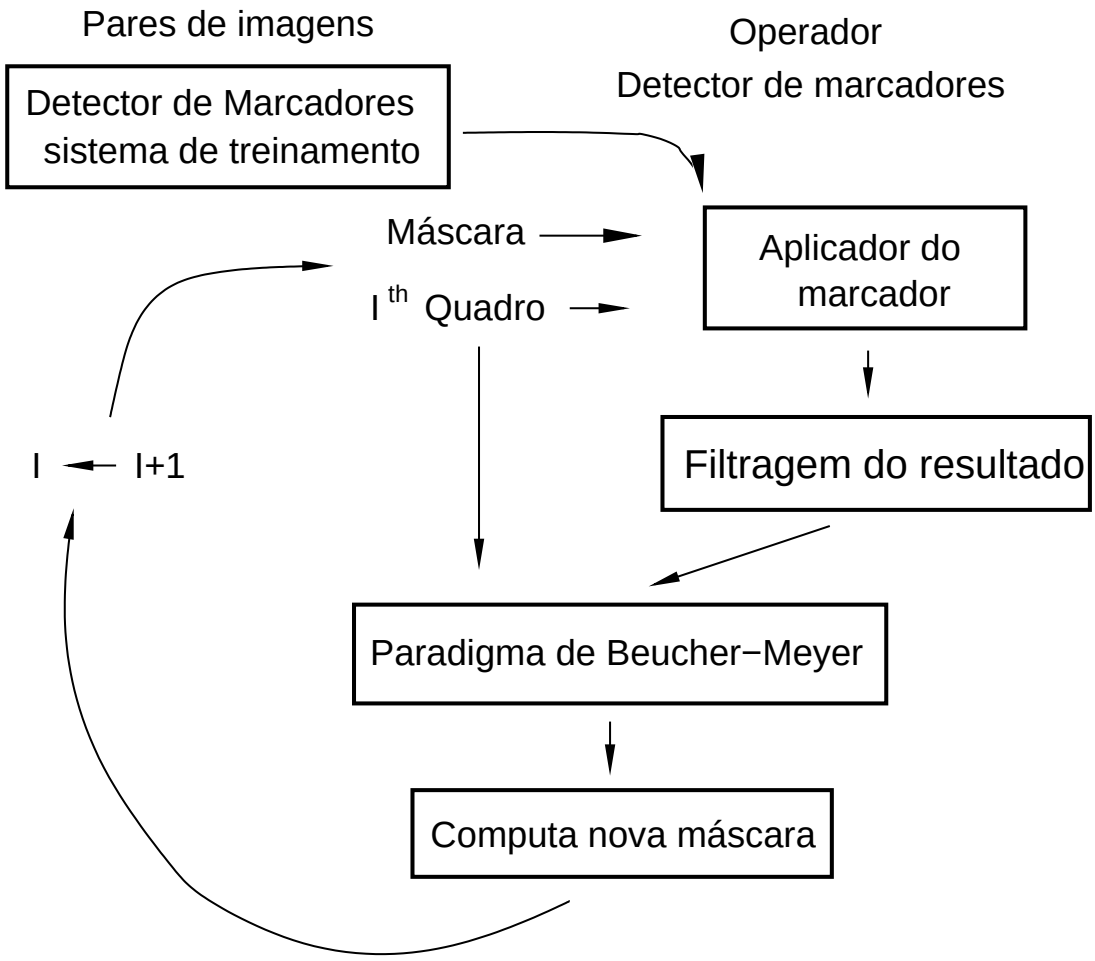


Figura 8.26: Fluxograma do método

é dada ao sistema junto com os pares de treinamento $\{f_i, g_i\}$ (na aplicação é diferente pois a imagem máscara é criada dinamicamente). A máscara restringe o domínio onde os pares de treinamento são coletados para dentro dela. A figura 8.27 mostra um objeto de interesse na i-ésima imagem, f_i, e uma máscara, m_i, usada para restringir o treinamento. O raio do círculo que forma a máscara é baseado no deslocamento aproximado do objeto na seqüência. Como o sistema permite a classificação de vários objetos, a imagem máscara é composta por vários desses círculos, tantos quantos forem os objetos que estão sendo perseguidos. Como ainda não temos uma maneira de projetar a "aperture" mais adequada, o usuário pode ter que tentar várias combinações de janela e quantidade de níveis de cinza, W e K, antes de conseguir uma boa segmentação. A quantidade de níveis de cinza de saída, M, é igual ao número de objetos que estão sendo perseguidos, mais um (para distinguir o fundo).

2. Aplicar o operador nas outras imagens da seqüência. O usuário localiza o(s) objeto(s) na primeira imagem da seqüência e fornece ao sistema o maior deslocamento aproximado do(s) objeto(s). O sistema filtra a imagem (se isso foi feito no treinamento), calcula a máscara de aplicação do operador na próxima imagem da seqüência, m_{i+1}, $i = [1 \ldots n[$, a partir da segmentação obtida na i-ésima imagem f_i e do parâmetro deslocamento (por exemplo, a dilatação com um elemento estruturante maior do que $\Delta S = V_0 \Delta T$ pontos). A figura 8.27 mostra um objeto na imagem $(i+1)$ e a respectiva máscara de aplicação.

3. Filtrar a imagem resultante pois o resultado de Ψ pode não ser perfeito (por diversas razões, principalmente por causa de falhas no treinamento). O filtro é necessário para eliminar os marcadores com baixa confiança estatística (usualmente pontos isolados ou pequenas regiões conexas). Este filtro é usualmente uma composição de filtros conexos, onde cada um deles é parametrizado por um elemento estruturante B (que estabelece a conectividade) e elimina as componentes conexas com área menor do que um limiar especificado pelo usuário. Um outro processo de eliminar componentes conexas que pode ser usado é feito atribuindo-se o nível de cinza da maior componente conexa vizinha às componentes menores com níveis de cinza diferentes da maior componente em que eles estão imersos. Desta forma, apenas as componentes conexas que forem maiores que o limiar serão os marcadores. Se o marcador for, em geral, maior que o objeto, o operador de poda ("prunning") deve ser aplicado para diminuir o tamanho do marcador.

4. - Aplicar o paradigma de Beucher-Meyer.

8.3.2 Exemplo de aplicação

No artigo "Automatic Design of Morphological Operators for Motion Segmentation" [84] e na dissertação "Segmentação de seqüências de imagens por morfologia matemática" por F. C. Flores [156], podem-se encontrar diversos exemplos de aplicação do método e, inclusive, uma comparação com outros métodos. Nesta seção mostraremos apenas um exemplo de aplicação em uma seqüência de imagens bem conhecida, o jogador de tênis de mesa [157]. A figura 8.28 mostra parte da primeira imagem da seqüência. O objeto que será rastreado e segmentado é a bola.

A seqüência mostra o jogador de tênis de mesa fazendo duas jogadas em 120 imagens em níveis de cinza com dimensões 480×421, cada. A bola é relativamente fácil de segmentar nas primeiras 8 imagens da seqüência pois aparece isolada. Nas próximas 3 imagens ela aparece ocluindo parte da

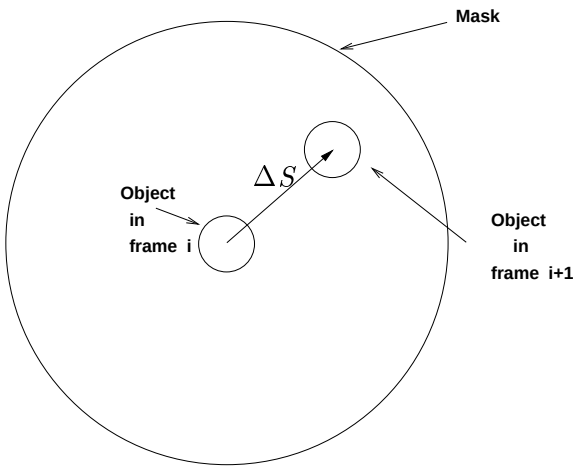


Figura 8.27: Máscara de restrição

Figura 8.28: Parte de uma imagem da seqüência do jogador de tênis de mesa

raquete e nas 5 imagens seguintes, seu formato não é mais um disco, mas um quase-cilindro. Isso acontece pela falta de resolução temporal (quantidade de imagens tomadas a cada segundo) do filme.

Para rastrear a bola nessas 16 imagens, usamos 5 imagens para treinar o operador, as 3 primeiras imagens da seqüência, 1 imagem de quando a bola oclui parte da raquete e 1 imagem de quando a bola deforma para um quase cilindro. A janela usada nesta parte foi uma janela espacial de nove pontos (3×3) com $k = 10$, $m = 3$ (um rótulo para o fundo, um para a bola e outro para a raquete) e posicionada nos níveis de cinza da imagem.

As figuras 8.29 a 8.31 mostram o resultado da aplicação do método para as primeiras 16 imagens. As linhas de partição de águas ao redor da bola está colorida de branco para diferenciar do fundo.

Outros rastreamentos dessa seqüência foram feitos (bola e raquete, ou apenas a raquete) e podem ser vistos juntamente com a descrição em:

`http://www.vision.ime.usp.br/demos.html`

8.4 Segmentação de seqüências de imagens coloridas

Esta aplicação é semelhante à anterior, mas há dois avanços importantes: (*i*) extensão do projeto de operadores "aperture" para imagens coloridas, que é feito concatenando-se as janelas de cada banda da imagem colorida em uma só janela e aplicando o projeto como anteriormente; e (*ii*) a aplicação do paradigma de Beucher-Meyer para imagens coloridas, que é feito via a aplicação de um gradiente colorido que realça as arestas das imagens coloridas e retorna essa informação em uma imagem em níveis de cinza, voltando assim ao escopo usual do paradigma. Grande parte dos conceitos básicos necessários para o entendimento desta parte do trabalho é apresentada nesta seção.

8.4.1 Operadores de imagens coloridas

Sejam L_1, L_2. ..., L_m reticulados completos totalmente ordenados. Seja L o produto cartesiano de L_1, L_2 ..., L_m, i.e., se $l \in L$, então $l = (l_1, l_2, \ldots, l_m)$, $l_i \in L_i$, $i = \{1, \ldots, m\}$.

Um mapeamento f de E em L, tal que $f(z) = (f_1(z), f_2(z), \ldots, f_m(z))$, onde $f_i : E \to L_i$ e $z \in E$, é chamado uma imagem *multivalorada* ou *multiespectral*. Os mapeamentos f_i são chamadas *bandas* da imagem multivalorada. Uma imagem colorida é um exemplo de uma função multivalorada onde $L = L_1 \times L_2 \times L_3$ é uma representação das cores sob um certo modelo de cores [151]. De agora em diante, vamos tratar apenas das imagens coloridas.

Três modelos de cores foram testados neste trabalho, RGB (*"Red"*, *"Green* e *"Blue"*), IHS (*"Intensity"*, *"Hue"* e *"Saturation"*) e YIQ (o sistema de cores usado em transmissões N.T.S.C.). Todos eles podem ser vistos como um sistema de coordenadas de cores. No sistema RGB, todos os eixos têm a mesma escala e cada eixo está relacionado a uma cor primária (i.e., vermelho, verde ou azul). Além disso, cada um dos eixos têm valores num intervalo finito (os valores associados a cada cor podem ir de 0 até o máximo valor que pode ser representado para cada cor na imagem). Assim, pode-se pensar no sistema RGB como um cubo com vértices[5] $(0,0,0)$ (preto), $(255,0,0)$ (vermelho), $(0,255,0)$ (verde), $(0,0,255)$ (azul), $(255,255,0)$ (magenta), $(255,0,255)$ (cyan), $(0,255,255)$ (amarelo) e $(255,255,255)$ (branco). Ou seja, uma cor está representada neste sistema por um ponto no cubo. O modelo IHS

[5]Considerando 255 como o valor máximo para cada cor pura.

Figura 8.29: Parte dos primeiros 3 quadros

Figura 8.30: Parte dos quadros 4 a 6

Figura 8.31: Parte dos quadros 7 a 9

é um sistema de coordenadas que pode ser definido a partir de uma transformação de coordenadas do sistema RGB [3]. Este espaço de cores é composto por três atributos: a intensidade representa a luminosidade da imagem, a cor descreve a cor na sua forma pura (cada valor no eixo cor representa uma cor diferente) e a saturação mede a quantidade de branco que está misturado com a cor. O sistema YIQ é usado no padrão N.T.S.C. de televisão colorida. O valor Y representa a luminância da cor e os valores I e Q descrevem juntamente a cor e a saturação [6].

Seja L como definido acima e M uma cadeia. O conjunto de todos os mapeamentos de E a L (i.e., o conjunto de todas as imagens coloridas sobre E com valores em L) e, respectivamente, de E a M (i.e., o conjunto de todas as imagens coloridas sobre E com valores em M), será denotado por Fun$[E, L]$ e Fun$[E, M]$. Um *operador de imagens coloridas para classificação* é um operador de Fun$[E, L]$ a Fun$[E, M]$. Este operador é usado para rotular objetos coloridos e será chamado por simplicidade de *detector de marcadores*.

Seja $W = \{W_1, W_2, W_3\}$, onde $W_i \subset E$, $i = \{1, 2, 3\}$. Para uma imagem colorida qualquer, f, uma *restrição* de f em W é configuração colorida $f|W$ dada por, para qualquer $x \in W$, $f(x)|W = (f_1(x)|W_1, f_2(x)|W_2, f_3(x)|W_3) = (f_1(x), f_2(x), f_3(x))$. Para qualquer $f \in Fun[E, L]$ e $x \in E$, a translação de f por t é a função f_t dada por, para qualquer $x \in E$, $f_t(x) = f(x - t) = (f_1(x - t), f_2(x - t), f_3(x - t))$.

A *classe de restrição* de f em W, denotada $\mathcal{F}_{f|W}$, é a família de imagens cuja restrição a W dá $f|W$, isto é, $\mathcal{F}_{f|W} = \{g \in Fun[E, L] : f|W = g|W\}$.

Um operador de imagem colorida Ψ é chamado *localmente definido no espaço* pela janela W sse, para qualquer $f \in Fun[E, L]$ e $x \in E$, $\Psi(f)(x) = \Psi(g)(x), \forall g \in \mathcal{F}_{f_{-x}|W}$. Um operador de imagens Ψ é chamado *espacialmente invariante por translação* sse, para qualquer $x \in E$, $\Psi(f_x) = \Psi(f)_x$. Um operador de imagens que é localmente definido no espaço por uma janela W e espacialmente invariante por translação é chamado de *W-operador*.

O projeto automático de operadores de imagens coloridas para classificação faz-se estimando a distribuição conjunta de cada configuração colorida e dos rótulos observados. Cada configuração colorida é transformada em uma configuração níveis de cinza concatenando-se as configurações observadas em cada banda (sem perda de informação). O resto do processo é idêntico ao projeto de operadores níveis de cinza. A aplicação do operador em uma dada configuração colorida também é feita concatenando-se as configurações de cada banda e obtendo-se a configuração níveis de cinza. Tendo-se essa configuração, procura-se a configuração nos intervalos do núcleo obtido na fase de treinamento.

8.4.2 Gradientes para imagens coloridas

Nesta seção, apresentamos duas definições de gradientes que foram usadas para testar o método de Beucher-Meyer em imagens coloridas.

Seja f uma imagem colorida representada em RGB, ou YIQ, então $f \in Fun[E, L]$, onde E é definido como anteriormente e $L = [0, 255] \times [0, 255] \times [0, 255]$. Se f está representada no modelo IHS, então $L = [0, 255] \times [0, 359] \times [0, 255]$.

Seja $x \in E$. A translação de $B \subset E$ por x, denotada B_x, é dada por $B_x = \{y \in E : (y - x) \in B\}$.

[6] As televisões P&B (preto e branco) funcionam pois usam apenas o atributo Y da transmissão.

Das definições clássicas de gradientes morfológicos, podemos definir vários gradientes para imagens coloridas [87]. Primeiramente recordamos as definições clássicas.

Definição 8.1 *Dada uma imagem $g \in Fun[E, L]$, onde E é um subconjunto do plano discreto e L um intervalo fechado dos inteiros, por exemplo $[0, 255]$, o gradiente de g, $\nabla_B(g)$, é dado por:*

$$\nabla_B(g) = \delta_B(g) - \varepsilon_B(g), \tag{8.1}$$

onde δ e ε são, respectivamente, a dilatação e a erosão morfológicas [16, 14], e $B \subset E$ é um elemento estruturante.

O resultado deste operador é uma imagem onde cada ponto é a diferença máxima entre os níveis de cinza dentro de B.

Definição 8.2 *Dada uma imagem $g \in Fun[E, L]$, onde E e L são definidos da mesma forma que anteriormente, o gradiente interno de g, $\nabla_B^i(g)$, é dado por:*

$$\nabla_B^i(g) = g - \varepsilon_B(g), \tag{8.2}$$

onde ε é a erosão morfológica e $B \subset E$ é um elemento estruturante.

A extensão para as imagens coloridas é dada pelas seguintes definições.

Para imagens no sistema de cores IHS, uma vez que a intensidade e a saturação são atributos com valores em L, os gradientes nessas componentes podem ser computados por $\nabla_B(\bullet)$. Entretanto, este não é o caso para o atributo "hue" cujos valores são ângulos ($h \in Fun[E, \Theta]$). Assim, uma outra métrica será definida para o cálculo do gradiente nessa componente.

Definição 8.3 *Sejam $\theta_1, \theta_2 \in \Theta$. A distância de cor entre θ_1 e θ_2 é dada por:*

$$d_h(\theta_1, \theta_2) = \lfloor \bigwedge \{|\theta_1 - \theta_2|, 360 - |\theta_1 - \theta_2|\} \rfloor. \tag{8.3}$$

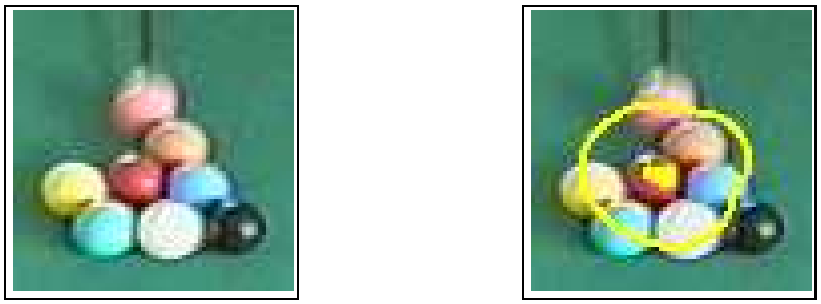
A distância introduzida acima retorna um valor inteiro entre 0 e 180. Para computar o gradiente proposto, estes valores são normalizados para a mesma escala dos outros dois atributos, isto é, para o intervalo K. Seja $m_K : [0, 180] \to [0, k-1]$ a função de normalização.

Definição 8.4 *Dada uma imagem $h \in Fun[E, \Theta]$, o gradiente angular $\nabla_\Theta(h)$, $\nabla_\Theta : Fun[E, \Theta] \to Fun[E, K]$, é dada por, para todo $x \in E$,*

$$\begin{aligned} \nabla_\Theta(h)(x) = \; & m_K\Big(\bigvee_{y \in \{B_x - \{x\}\}} d_h(h(x), h(y)) \\ & - \bigwedge_{y \in \{B_x - \{x\}\}} d_h(h(x), h(y)) \Big). \end{aligned} \tag{8.4}$$

onde $B \subset E$ é um elemento estruturante centrado na origem.

Seja μ_I, μ_H e μ_S três números inteiros. O próximo gradiente é uma combinação linear dos gradientes morfológicos para I e S, com o gradiente angular para H. Os coeficientes da combinação linear são usados para realçar ou diminuir a contribuição de cada subespaço no cálculo do gradiente.



(a) (b)

Figura 8.32: (a) Imagem original. (b) Marcadores.

Definição 8.5 *Dada uma imagem colorida $f \in Fun[E, L]$ representada no modelo de cores IHS, o gradiente colorido V, $\nabla^V(f)$, $\nabla^V : Fun[E, L] \to Fun[E, Z]$, é dado por:*

$$\nabla^V(f) = \mu_I \nabla_B(f_1) + \mu_H \nabla_\Theta(f_2) + \mu_S \nabla_B(f_3), \tag{8.5}$$

onde f_1, f_2 e f_3 são as bandas I, H, e S, respectivamente, e $B \subset E$ é um elemento estruturante centrado na origem.

Transformando as coordenadas de RGB para YIQ e aplicando um gradiente morfológico apenas sobre a componente Y da imagem, obtemos uma outra forma de conseguir uma imagem em níveis de cinza cuja informação são as bordas da imagem realçada.

Definição 8.6 *Dada uma imagem colorida $f \in Fun[E, L]$ representada no modelo de cores YIQ, o gradiente ∇^{VI}, $\nabla^{VI} : Fun[E, L] \to Fun[E, K]$, de f é dado por:*

$$\nabla^{VI}(f) = \nabla_B(f_1) \tag{8.6}$$

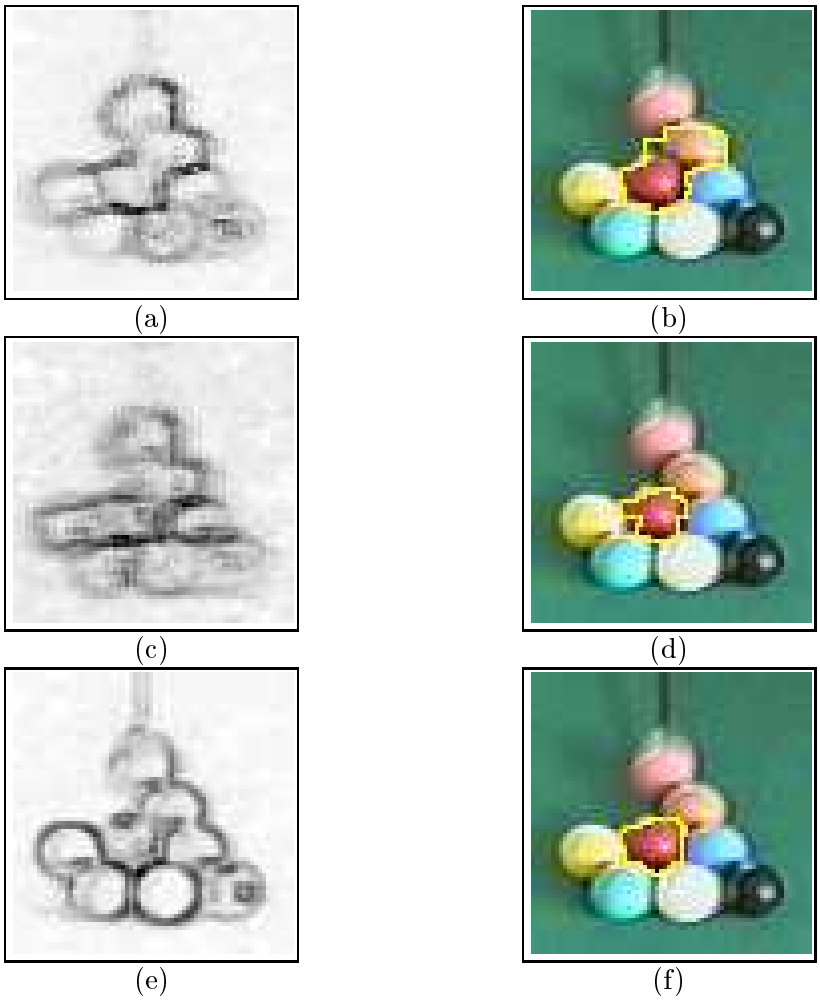
onde $B \subset E$ é o elemento estruturante centrado na origem de E.

Usando o paradigma de Beucher-Meyer, aplicamos a idéia para segmentar a bola vermelha apresentada na figura 8.32a). Um marcador interno é associado a bola, assim como um marcador externo (ilustrado na figura 8.32b). Há várias bolas coloridas na imagem e a borda delas é realçada diferentemente para cada gradiente. O gradiente V é aplicado sobre imagens IHS e o gradiente VI sobre imagens YIQ.

O paradigma aplicado ao gradiente V (ilustrado na figura 8.33a) apresenta alguns defeitos na segmentação, mas isso ocorre também com outros gradiente. O paradigma aplicado ao gradiente 8.33b é melhor para esta imagem, mas erra em outras, também. Entre todos os gradientes testados para dois conjuntos de imagens, o gradiente V deu os melhores resultados, embora tenhamos que escolher os pesos nesse caso. O gradiente VI foi o segundo melhor e neste caso não temos que escolher pesos.

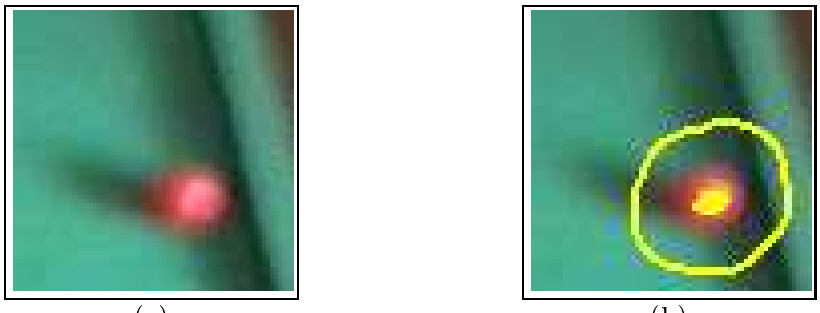
A figura 8.34a ilustra uma outra imagem onde aplicamos os mesmos gradientes e o paradigma. Desta vez há apenas uma bola, mas a imagem tem duas características interessantes. Além da existência das duas regiões sombreadas, ela está embaçada. A figura 8.34b ilustra os marcadores interno e externos.

Todos os gradientes coloridos testados funcionam bem neste caso, mesmo a bola estando localizada ao redor de uma região embaçada de alta luminosidade. As figuras 8.35a, c, e, g, i, k, e m ilustram



(a) (b) (c) (d) (e) (f)

Figura 8.33: Resultados dos diferentes operadores gradiente e das segmentações resultantes. (a-b) gradiente V, (c-d) gradiente V (com pesos), (e-f) gradiente VI.



(a) (b)

Figura 8.34: Outro exemplo de marcadores (a) Imagem original. (b) Marcadores.

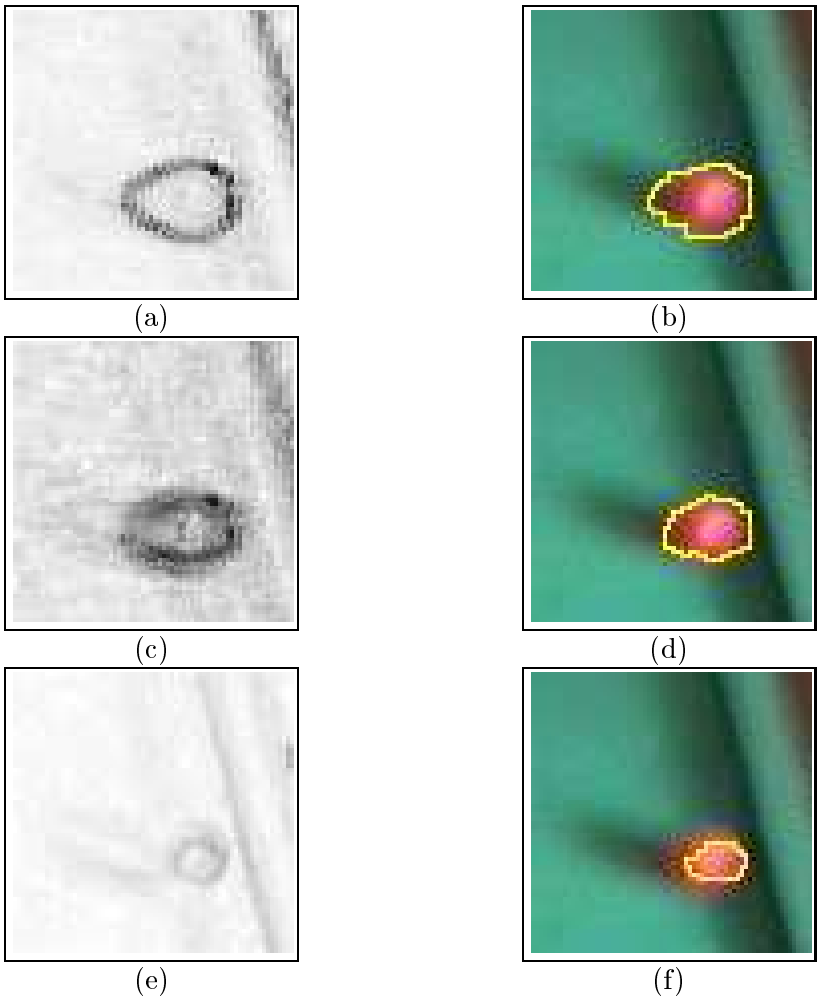


Figura 8.35: Gradientes coloridos e o resultado da segmentação. (a-b) gradiente V. (c-d) gradiente V ($\mu_S = 10$). (e-f) gradiente VI.

os resultados dos gradientes, e as figuras 8.35 b, d, f, h, j, l, e n ilustram o resultado da segmentação. O resultado do gradiente V com pesos iguais para todas as bandas é ilustrado na figura 8.34i, e com peso diferente para a banda de saturação ($\mu_S = 10$) é ilustrado na figura 8.34k. O resultado da segmentação são ilustrados nas figuras 8.34j e 8.34l, respectivamente. Visualmente estes são os melhores. Para trinta e três imagens testadas, o gradiente V foi o que melhor na média.

A figura 8.36 mostra uma outra imagem, de melhor qualidade, e a figura 8.37 mostra os marcadores usados para segmentar as gomas verdes.

As figuras 8.38, 8.39 e 8.40 mostram, respectivamente, o resultado da segmentação para os gradientes coloridos V com pesos iguais, V considerando apenas a componente H, e VI. Todos os gradientes erraram em alguma das gomas.

Figura 8.36: Imagem das balas de goma coloridas.

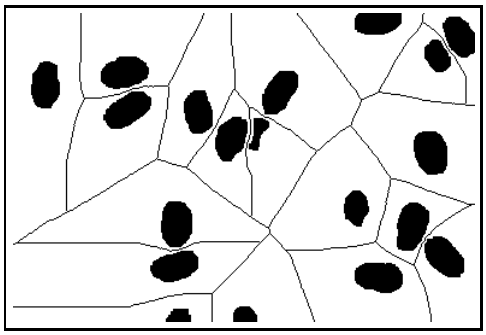


Figura 8.37: Marcadores para as gomas verdes.

Figura 8.38: Gradiente V.

Figura 8.39: Gradiente V - apenas a componente H.

Figura 8.40: Gradiente VI.

8.4.3 Segmentação do vídeo "Bolas de sinuca"

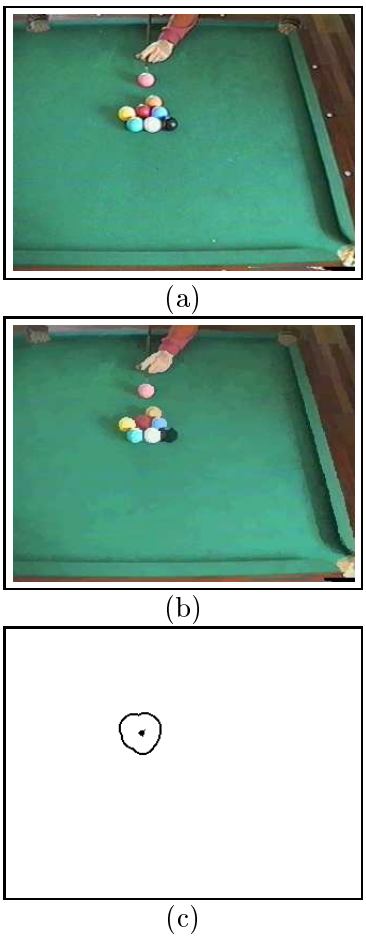
Este é um exemplo da aplicação das idéias apresentadas nas seções anteriores para a segmentação de vídeos coloridos. O objetivo deste exemplo é segmentar uma bola vermelha num vídeo de uma mesa de sinuca. A figura 8.32a mostra um exemplo de um dos quadros da seqüência. Há oito bolas que se movem durante alguns segundos de cena e cada uma delas têm uma cor distinta. A seqüência foi feita com uma câmera de vídeo amadora e tem variação de luminosidade durante a cena, além de desfocagem nos últimos quadros do vídeo.

Para garantir um bom acerto por parte do operador, uma pré-filtragem da seqüência foi feita com filtros conexos aplicados a cada banda da imagem. Essa filtragem reduz as zonas planas ("flat zones") da imagem e, também, o número de cores observadas. O filtro usado é uma composição de um "area open" e um "area closing", seguido por um operador que junta zonas planas pequenas (com área abaixo de um limiar) a zonas vizinhas maiores. A figura 8.41b mostra o resultado do filtro quando aplicado à imagem apresentada na figura 8.41a.

Dez pares de imagens (imagens filtradas e as respectivas segmentações rotuladas com rótulos diferentes para cada bola) foram usados para treinar o operador. A janela usada foi de apenas um ponto em cada banda (graças às imagens coloridas terem mais informações, não foi necessário usar uma janela espacial maior).

Após o treinamento do operador, a aplicação é feita localizando-se a bola de interesse (no caso a vermelha) na primeiro quadro. O operador treinado é aplicado dentro da máscara imposta sobre cada quadro filtrado (usando o mesmo filtro conexo usado no treinamento) e depois filtrado, para eliminar componentes conexas pequenas, e depois diminuído para não ultrapassar o tamanho do objeto de interesse. O marcador externo é a borda externa da máscara usada para restringir a aplicação do operador. A figura 8.41c apresenta os marcadores usado para segmentar a bola vermelha na imagem apresentada na figura 8.41a.

Finalmente, o paradigma de Beucher-Meyer (usando o gradiente V com pesos $\mu_I = 1, \mu_H = 1$ e $\mu_S = 10$) é aplicado sobre o quadro original usando os marcadores internos e externos encontrados anteriormente. A figura 8.42 mostra o resultado da segmentação, isto é, a linha de partição de águas composta com a bola vermelha. A informação da banda H ("hue") é a mais importante nesta seqüência pois todas as formas são semelhantes a menos da cor. A figura 8.43 mostra a bola vermelha segmentada em seis quadros consecutivos. A coluna à esquerda mostra uma parte do quadro e a coluna à direta as linhas de partição de águas.



(a)

(b)

(c)

Figura 8.41: Parte da seqüência das bolas de sinuca. (a) um dos quadros da seqüência. (b) o resultado da filtragem. (c) os marcadores.

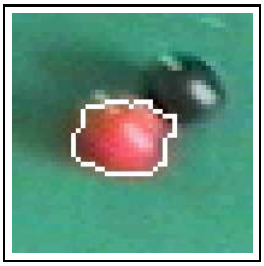


Figura 8.42: Um dos quadros da seqüência segmentado.

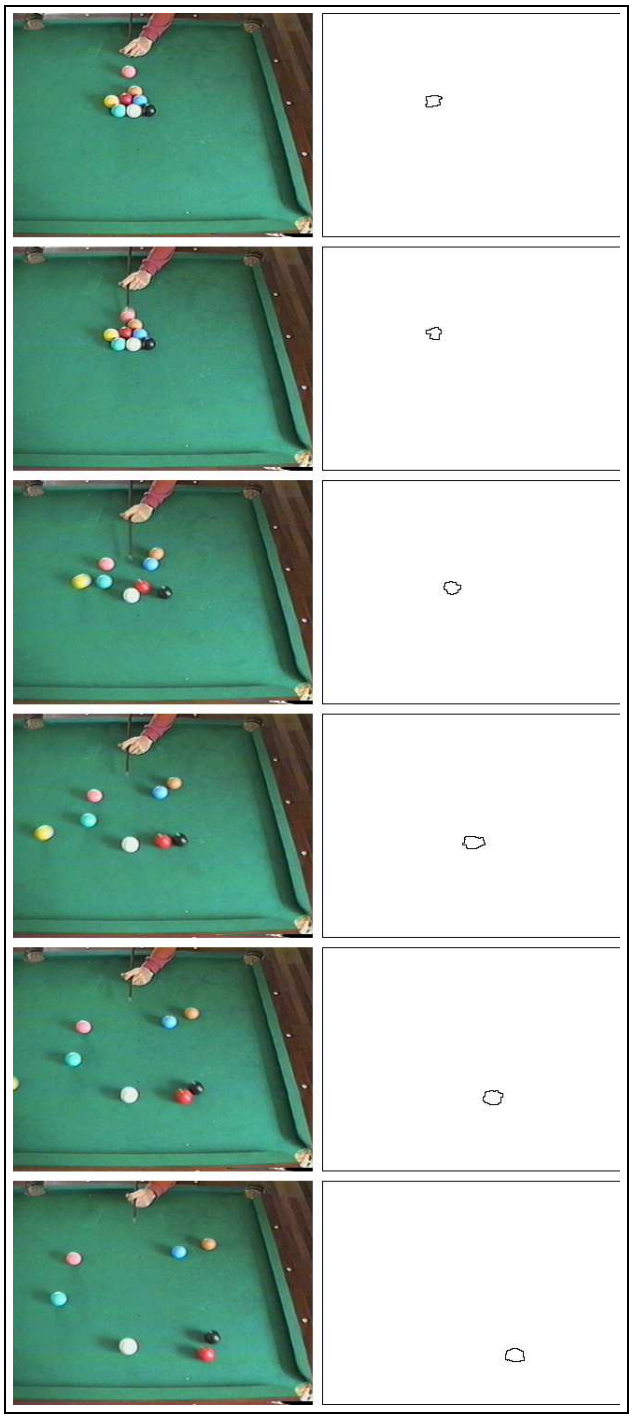
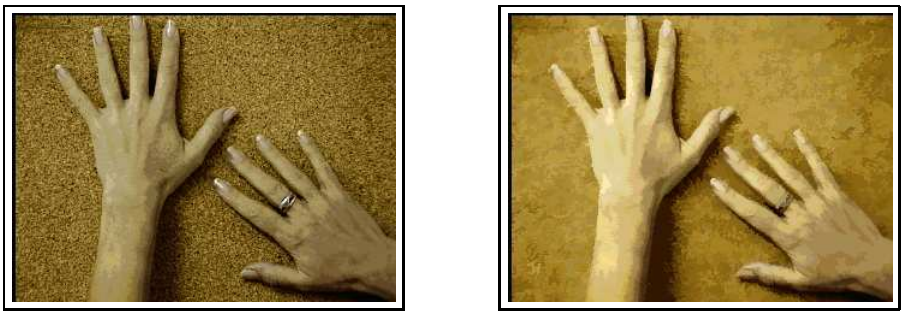


Figura 8.43: Resultado da segmentação.



(a) (b)

Figura 8.44: (a) Um quadro da seqüência. (b) Resultado da fltragem.

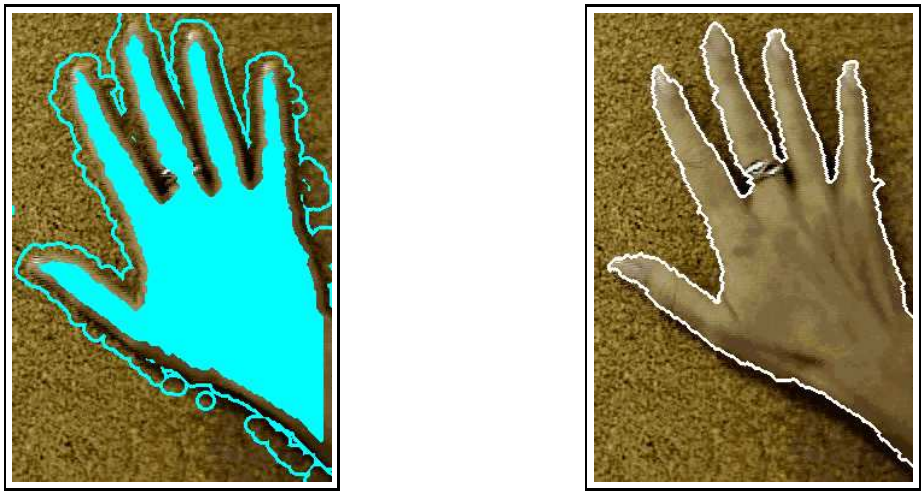
8.4.4 Segmentação do video "Mãos"

A seqüência "Mãos" mostra duas mãos movendo-se sobre uma placa de cortiça. Esta seqüência é composta de 60 quadros em RGB. Um exemplo de um dos quadros é mostrado na figura 8.44a. A aplicação neste exemplo é segmentar ambas as mãos. Este problema é mais difícil que o anterior por há dois objetos para serem seguidos e cada mão é composta de várias partes que precisam ser segmentadas, tais como as unhas e o anel na mão direita.

Nesta aplicação também foi usada a estratégia de pré-filtrar a imagem usando filtros conexos para melhorar a quantidade de acertos do operador projetado. A seqüência pré-filtrada é obtida pela composição de um "area open" e um "area closing", seguido por um operador que junta zonas planas pequenas (com área abaixo de um limiar) a zonas vizinhas maiores. A figura 8.44b ilustra o resultado da pré-filtragem na imagem da figura 8.44a. Dez pares de imagens são usados para projetar o operador. Cada par é composto pela imagem pré-filtrada e pela segmentação manual das mãos na imagem original (as mãos têm rótulo 1 e o fundo o rótulo 0).

O operador projetado é aplicado da mesma forma que anteriormente e os marcadores também são filtrados para eliminar pequenas componentes conexas e diminuído para não ser maior que as mãos. A figura 8.45a mostra os marcadores externos e internos para a imagem mostrada na figura 8.44a. O paradigma é aplicado usando o gradiente colorido V (com $\mu_I = 1, \mu_H = 0$ e $\mu_S = 1$). A figura 8.45b mostra o resultado da segmentação composto com a mão.

A figura 8.46 ilustra o resultado do método aplicado a alguns dos quadros da seqüência. A coluna à esquerda mostra os quadros originais e a coluna da direita mostra o resultado da segmentação.



(a) (b)

Figura 8.45: Resultado. (a) Marcadores. (b) Linhas de partição de águas.

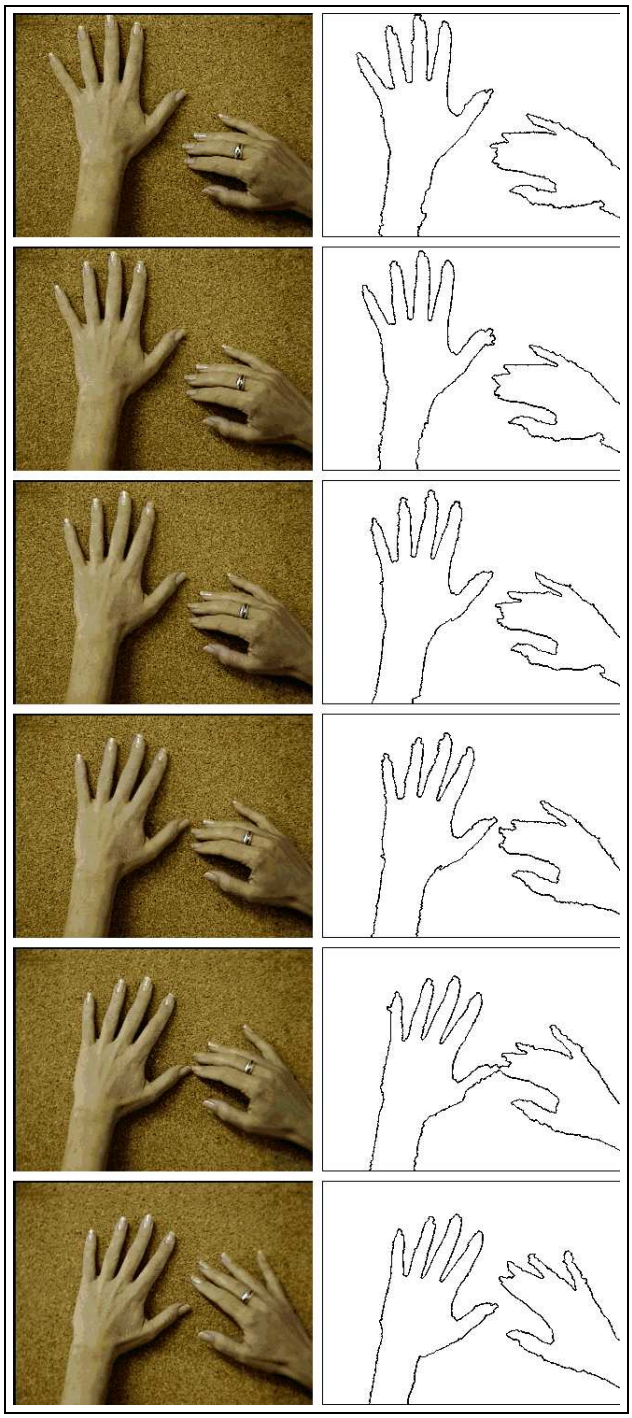


Figura 8.46: Alguns quadros da seqüência das mãos.

Capítulo 9

Conclusão

Neste capítulo analisamos os principais resultados deste trabalho e propomos algumas idéias para trabalhos futuros.

O projeto automático de W-operadores para imagens em níveis de cinza foi relegado a um segundo plano por muitos anos. Vimos diversas razões para esse fato e mostramos como foi possível tratar o problema via *reticulados discretos finitos*. Isso impôs novas restrições ao espaço de operadores e o preço é a possível subotimalidade do operador projetado.

A primeira contribuição teórica deste trabalho foi a caracterização dos operadores "aperture". Estes operadores são invariantes por translação no espaço, localmente definidos por uma janela espacial W e por uma janela nos níveis de cinza K.

A partir da caracterização desses operadores, um sistema para o seu projeto automático foi especificado e implementado. Esta é uma contribuição prática não apenas para o desenvolvimento deste trabalho mas, também, para outros pesquisadores do grupo que vierem a desenvolver pesquisas relacionadas ao assunto.

Cuidados foram tomados no projeto e na implantação do sistema ("software") para que pudéssemos testar não apenas com os parâmetros de janelas (espacial e níveis de cinza) como também com o posicionamento da janela, com funções de perda e representações diferentes.

A primeira representação testada foi via árvores de decisão. Isso aconteceu pois sua implementação é simples e seu poder de "generalização" é conhecidamente bom.

Essa montagem foi suficiente para fazermos centenas de experimentos nos quais comprovamos a superioridade dos operadores "aperture" sobre os filtros lineares ótimos de janela restrita (que é uma aproximação do filtro de Wiener) tanto para sinais, quanto para imagens, usando o critério MAE ou o MSE.

Fizemos experimentos de filtragem de ruído gaussiano de baixa variância sobre uma porcentagem dos pontos da imagem, desembaçamento de sinais ou imagens convoluídos por núcleos de convolução planos, ou não (como no caso das imagens) e restauração de sinais que foram degradados por ruído gaussiano e embaçamento. Em todos os casos, os operadores "aperture" foram melhores que o filtro ótimo linear de janela restrita para ambos MAE e MSE.

No caso das imagens, o número de exemplos de treinamento não foi suficiente para que os operadores "aperture" fossem melhores que o filtro linear ótimo de janela restrita para o MSE, mas

apenas para o MAE. Ainda assim, visualmente, o resultado é melhor pois os operadores aperture erram menos nas bordas, embora não corrijam os picos tão bem. Já os operadores lineares são bons para corrigir esses picos, mas péssimos com bordas. Porém, ficou evidente pelas curvas de erro que os operadores "aperture" ganhariam dos operadores lineares se mais exemplos fossem fornecidos para o treinamento.

A pesquisa com os operadores "aperture" representados por árvores de decisão sobre esses modelos de sinais e imagens serviu, também, para explorarmos diversos parâmetros dos quais os operadores dependem (tamanho e forma da janela espacial, tamanho da janela níveis de cinza e posicionamento da janela) e como eles influenciam no erro do operador, por exemplo, confirmamos a relação do tamanho da janela espacial sobre o erro do operador que já conhecíamos do projeto de operadores binários, e também estudamos a relação do tamanho da janela níveis de cinza sobre o erro do operador. Em ambos os casos vale que quanto maior a janela, menor o erro desde que haja exemplos suficientes para treinar o operador.

Vimos também que não há uma relação simples quando variamos esses dois parâmetros pois as relações de inclusão que há quando variamos apenas um desses parâmetros não existem mais.

As árvores de decisão têm pelo menos uma desvantagem no contexto que estamos estudando: a representação do operador se faz via uma partição do espaço das possíveis configurações do domínio da função característica do operador e cada elemento da partição está associado a um rótulo. Como a representação morfológica de um operador se faz via intervalos maximais do núcleo do operador, isso caracteriza uma desvantagem pois dificulta a integração do conhecimento e conseqüentemente do "software" como um todo.

Embora seja possível, não é simples passar da representação por árvores para intervalos[1].

Por causa disso, o algoritmo ISI binário foi estendido para o caso níveis de cinza. Essa extensão implicou em uma contribuição teórica pequena, mas original: a regra para a quebra de um intervalo níveis de cinza. O algoritmo foi implementado e algumas comparações simples foram feitas para verificar a sua indução sobre as configurações não treinadas. Porém, essas comparações não são conclusivas.

As duas abordagens já comentadas, árvores de decisão e ISI não esgotam as possíveis formas de projetar um operador de imagens. Outras abordagens poderiam ter sido testadas, por exemplo: redes neurais, "supporting vector machines" e etc. Neste sentido, como os resultados do projeto de operadores com representação via multiresolução para o caso binário eram bons, resolvemos testá-lo também para o caso dos operadores "aperture".

A caracterização dos operadores "aperture" multiresolução é a última contribuição teórica deste trabalho. Ao sistema foi adicionada esta possibilidade e os experimentos de desembaçamento de sinais e imagens foram repetidos usando a abordagem multiresolução. Os resultados desses experimentos foram melhores que os filtros lineares ótimos de janela restrita e, também, melhores que os resultados com as árvores de decisão.

[1] A transformação de intervalos para árvores no caso binário ou multiclassificação foi resolvida e implementada no sistema.

9.1 Aplicações

Demonstramos a utilidade dos operadores "aperture" para desembaçamento em duas imagens reais de satélite. Na primeira imagem, embora tenha faltado exemplos de treinamento, há uma vantagem no MAE do operador "aperture" em relação ao operador linear ótimo de janela restrita. Para a segunda imagem, em que tínhamos mais exemplos de treinamento, tanto o MAE quanto o MSE foram melhores que o melhor operador linear de janela restrita.

Apresentamos um experimento de aumento de resolução de imagens em níveis de cinza usando os operadores "aperture". Embora o resultado tenha sido melhor que dois operadores lineares clássicos de aumento de resolução de imagens, não fizemos mais experimentos pois há métodos não-lineares muito bons para isso.

Demonstramos a utilidade dos operadores "aperture" para localização de marcadores de objetos e auxílio em segmentação de seqüências de imagens (vídeos) níveis de cinza e coloridas. Para estas últimas, estendemos a teoria dos operadores "aperture" para imagens coloridas.

9.2 Trabalhos futuros

Acreditamos que ainda exista muito para se estudar tanto em termos teóricos, quanto práticos, sobre os operadores "aperture" e seu projeto.

Em termos de projeto, várias são as frentes de pesquisa que podem surgir. Por exemplo:

- faz-se necessário um aprofundamento do estudo teórico da precisão do operador "aperture". Isto poderia ser feito inicialmente sobre modelos de imagens simples, cuja comprovação prática seria possível.
- Ainda não foi feito um estudo da robustez do projeto desses operadores.
- Embora saibamos que o melhor posicionamento da "aperture", em termos do quanto do sinal é deixado de ser observado, é na mediana do sinal observado, não sabemos muito sobre a relação desse posicionamento com o erro do operador. Como constatamos experimentalmente, o posicionamento influencia o erro de uma forma não-trivial.
- Embora tenhamos feito experimentos com o projeto iterado de operadores "aperture", certamente há muito que ser estudado tanto em termos teóricos, quanto práticos.

O estudo do projeto automático de operadores morfológicos binários já tem mais de 10 anos e ainda não está esgotado. Da mesma maneira, acreditamos que o projeto dos operadores "aperture" esteja apenas no começo e que haja muito para ser pesquisado. Nossa esperança é que a pequena contribuição que demos aqui possa ser útil para outras pessoas que queiram fazer projeto automático de operadores em níveis de cinza.

Apêndice A

Segmentação de imagens

A segmentação de imagens é uma etapa importante e, em geral, muito difícil. Ela é importante pois é necessária na maior parte das soluções de análise de imagens [2, 3]. Ela é difícil pois é um problema mal-posto [155].

Um método de segmentação muito robusto e simples que vem da morfologia matemática é o paradigma de Beucher-Meyer [10]. Através dele podemos achar as fronteiras de objetos previamente especificados via marcadores, i.e, via regiões conexas de E na mesma posição dos objetos que queremos segmentar. O processo de segmentação reduz-se assim ao problema de encontrar esses marcadores [83].

A segmentação de imagens via o paradigma o encontro das bordas dos objetos que queremos segmentar é uma das duas mais importantes abordagens conhecidas de segmentação de imagens; a outra é agrupar os pontos da imagem que são similares. O paradigma de Beucher-Meyer aplica-se via o operador *LPE*, ou "*watershed*" [158, 159, 24], composto com alguns operadores morfológicos que são úteis para preparar a imagem a ser segmentada. Esta preparação é muitas vezes necessária para a eliminação das bordas dos objetos que não estamos interessados, e também as bordas que aparecem devido a ruído na imagem.

Bordas são, em geral, descontinuidades na imagem e podem ser detectadas por operadores de diferenciação, como o *gradiente morfológico* [16]. Este operador é, entretanto, muito sensível a ruídos na imagem, i.e., ele realça as transições devido às bordas, mas também devido a ruído. Assim, a imagem resultante da aplicação do gradiente morfológico é usualmente uma imagem ruidosa no sentido que ela tem mais informação do que seria necessário. A aplicação resultante da aplicação do operador LPE na imagem do gradiente produz, usualmente, uma imagem super-segmentada [24, 83]. A solução deste problema é já bastante conhecida e consiste em eliminar as bordas que não pertencem aos objetos que serão segmentados através de um operador que muda a *homotopia* [16] do gradiente da imagem [10, 83].

Definição A.1 *Seja $f \in L^E$ o resultado da aplicação do gradiente morfológico a imagem $h \in L^E$. Seja $\{E_i\}$, $E_i \subset E$, uma família de marcadores para os objetos que vão ser segmentados em h. Seja $g \in L^E$ uma imagem produzida atribuindo o valor k_M, onde k_M é o maior nível de cinza de h, a todos os pontos de $\{E_i\}$ e k_m, onde k_m é o menor nível de cinza em h, para todos os outros pontos. O operador que muda a homotopia do gradiente é definido por:*

$$\Theta(f) = \rho^*_{f \cap \nu(g)}(\nu(g)) \tag{A.1}$$

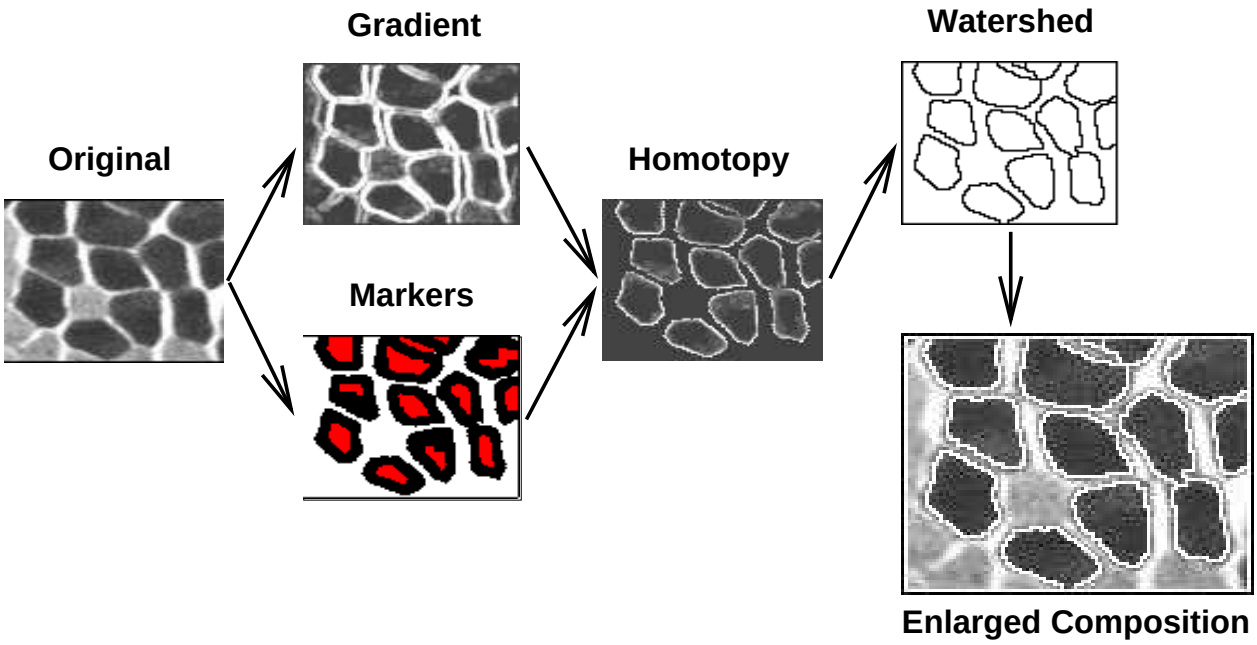


Figura A.1: Paradigma de Beucher-Meyer

onde ρ^* é o operador de reconstrução dual [16, 25, 83] e ν é a operação de negação [16, 83].

Dada uma imagem f, o operador Θ é um filtro [16, 17] que muda somente os mínimos regionais [160, 16, 17] de f (não há modificação dos máximos regionais [160, 16, 17]). Portanto, podemos dizer que ele muda a homotopia baixa de f [160]. Esta propriedade é importante pois garante que as linhas de partição de águas encontradas após a filtragem são um subconjunto das linhas de partição de águas achadas na imagem filtrada, i.e., não é adicionado nenhum máximo regional, apenas mínimos regionais são excluídos.

A figura A.1 ilustra a aplicação do paradigma para a segmentação de uma imagem de fibras musculares. Da esquerda para a direita, e de cima para baixo, a figura mostra a imagem a ser segmentada, o resultado do gradiente, os marcadores escolhidos para as fibras mais escuras (objetos conexos claros dentro das fibras), o resultado da filtragem homotópica do gradiente (invertido para melhor visualização), o resultado do operador LPE, e a imagem original composta com as bordas encontradas (aumentada para melhor visualização).

O paradigma de Beucher-Meyer é uma das mais poderosas técnicas conhecidas para segmentação de imagens. A grande qualidade desta estratégia é que o problema de detecção de contornos que, em geral, é muito complicado, é substituído por um problema mais simples, que é o de achar marcadores para os objetos a serem segmentados na imagem.

Os marcadores podem ser desenhados a mão, via uma interface de auxílio a segmentação; ou podem ser achados pela combinação de operadores de imagens projetados de forma "ad-hoc"; ou podem ser achados por operadores projetados automaticamente via um sistema de projeto que utiliza exemplos de treinamento fornecidos pelo usuário. Algumas vezes, no entanto, é difícil evitar usar alguma heurística no processo já que a segmentação envolve um conhecimento a priori dos objetos a serem segmentados.

Apêndice B

Publicações relacionadas ao tema da tese

Listamos em seguida os artigos publicados e submetidos durante o período do doutorado, além de um pequeno resumo de seu conteúdo.

B.1 Artigos de revista.

1. Ref. [161] Multiresolution Design of Aperture Operators

   O projeto de um operador "aperture" é baseado em restringir adequadamente o domínio espacial e o domínio dos níveis de cinza para diminuir o espaço de operadores e, conseqüentemente, o erro de estimação. O projeto de um operador restrito por resolução é baseado em combinar adequadamente as informações de duas ou mais resoluções e tem a mesma motivação que antes, isto é, diminuir o espaço de operadores para imagens em níveis de cinza. A restrição no domínio espacial, no domínio dos níveis de cinza e em ambas é caracterizada neste artigo e o aumento de erro por usar um filtro restrito em lugar de um filtro ótimo é analisado. A projeto multiresolução piramidal envolve aplicar a abordagem hierarquicamente, do espaço de resolução maior para o de resolução menor; no artigo caracterizamos essa abordagem e analisamos o erro do projeto. O sistema que faz o projeto multiresolução piramidal é descrito e sua complexidade analisada. Várias simulações para desembaçamento de imagens foram feitas e comparadas com filtros ótimos lineares de janela restrita e os resultados confirmam a utilidade da abordagem.

2. Ref. [78] Aperture Filters.

   Exceto no caso de sinais (ou imagens) com muito poucos níveis de cinza, o projeto de filtros MSE ótimos localmente definidos por uma janela é prejudicado pela inabilidade de conseguir dados de treinamento em quantidade suficiente para fazer uma estimativa precisa aceitável de um grande número de esperanças condicionais requeridas para o projeto do filtro. Mesmo usando restrições típicas como impor que o operador seja crescente, ou que haja uma decomposição iterativa, as estimativas permanecem intratáveis. Este artigo usa a restrição nos níveis de cinza para tratar este problema de estimação. Em cada ponto, o sinal é visto através de uma abertura, que é o produto entre uma janela espacial e uma janela nos níveis de cinza posicionada de acordo com os valores observados. Os valores que ultrapassam os níveis de cinza possíveis

vistos pela janela são projetados para dentro da abertura, tanto de cima para baixo, como de baixo para cima. Esta projeção comprime a massa de probabilidades do sinal observado em um conjunto de variáveis menor de forma a não alterar a massa de observações dentro da abertura (que se bem escolhida deve carregar a maior parte da massa) e altera minimamente a massa fora da abertura. Os experimentos mostram que os filtros "aperture" dão resultados melhores que os filtros lineares para restauração de embaçamento (desembaçamento), especialmente na restauração de arestas. Este paper trata de vários assuntos, como o posicionamento da abertura, o efeito da restrição nos níveis de cinza sobre a massa de probabilidade, o tamanho da abertura em relação à precisão do operador e ao tamanho das amostras de treinamento, a estimação das probabilidades condicionais e a representação do operador por árvores de decisão. Uma amostra dos experimentos feitos para estudar os efeitos dos filtros "aperture" é apresentada para os casos de filtragem de ruído aditivo e restauração de embaçamento.

3. Ref. [68] Automatic Programming of Morphological Machines by PAC Learning.

   Um aspecto importante da morfologia matemática é a descrição de operadores entre reticulados completos por uma linguagem formal, a Linguagem Morfológica (ML), cujo vocabulário é composto das operações de ínfimo e supremo, e dos operadores dilatação, erosão, anti-dilatação e anti-erosão. Esta linguagem é completa (isto é, ela pode representar qualquer operador entre reticulados completos) e expressiva (isto é, muitos operadores úteis podem ser representados por frases com, relativamente, poucas palavras). Desde os anos sessenta, as máquinas morfológicas (MMachs), têm sido construídas para implementar a linguagem morfológica restrita ao reticulado das imagens binárias e de níveis de cinza. Entretanto, projetar programas da MMach úteis não é uma tarefa elementar. Recentemente, muito esforco têm sido feito para automatizar a programação das MMachs. O objetivo das diferentes abordagens para este problema é achar representações formais convenientes do conhecimento para descrever transformações sobre estruturas geométricas e traduzí-las automaticamente em programas da MMach por sistemas computacionais. Nós apresentamos neste artigo as idéias centrais de uma abordagem baseada na representação das transformações por coleções de pares de imagens observada-ideal e a estimação de operadores a partir desses dados. Nesta abordagem, a estimação dos operadores é baseada em otimização estatística or, equivalentemente, em um ramo da teoria de aprendizado computacional conhecida por aprendizado PAC. Estes operadores são gerados na forma padrão dos operadores morfológicos e podem ser simplificados (isto é, transformados em operadores morfológicos equivalentes que usam menos palavras do vocabulário) por transformações sintáticas.

B.2 Artigos em conferência.

1. Ref. [81] Image restoration by multiresolution aperture filters.

   Este artigo estuda o projeto de operadores "aperture" via multiresolução piramidal. A abordagem de multiresolução já havia sido estudada para o projeto de operadores binários e neste trabalho estendemos a teoria para níveis de cinza. Os resultados experimentais mostraram que a abordagem pode ser vantajosa tanto para sinais, quanto para imagens.

2. Ref. [162] Design of Gray-scale Nonlinear Filters via Multiresolution Apertures

   Este artigo aplica a idéia do projeto piramidal multiresolução, que foi usado para projetar filtros binários, para projetar operadores "aperture", que são operadores níveis de cinza não lineares

que operam sobre dados conseguidos via uma janela espacial e nos níveis de cinza ("aperture"). Os operadores projetados são usados para achar marcadores para uma região dos olhos em um repositório de imagens de faces.

3. Ref. [163] Some Applications of Aperture Filters

   Filtros "aperture" formam uma classe de operadores em níveis de cinza que são invariantes por translação e localmente definidos por uma janela espacial e uma janela nos níveis de cinza. O projeto ótimo de um filtro desta classe é possível praticamente porque pode-se impor o tamanho de ambas as janelas espacial e níveis de cinza, e portanto reduzir grandemente o número de parâmetros a serem estimados no projeto do filtro. Isto é conseguido sem afetar muito a massa de probabilidade que será usada para a otimização completa. Este artigo mostra várias aplicações do mundo real dos filtros "aperture", tais como a extração automática de marcadores, desembaçamento e melhoramento de resolução.

4. Ref. [86] Morphological Operators for Segmentation of Color Sequences

   Este artigo apresenta uma técnica para a segmentação de objetos em uma seqüência de imagens coloridas. A técnica é baseada no paradigma de Beucher-Meyer, com marcadores detectados por um operador morfológico projetado por aprendizado computacional (or, equivalentemente, otimização estatística). Os objetos de interesse são marcados em alguns quadros do vídeo manualmente e usados para treinar um detector de marcadores. Então, o operador projetado é usado para marcar os objetos em outros quadros e o paradigma é aplicado sobre todos os quadros marcados pelo detector. Dois exemplos do mundo real ilustram a aplicação da técnica proposta.

5. Ref. [87] Color Image Gradients for Morphological Segmentation

   Este artigo propõe uma abordagem para a segmentação colorida que é uma alternativa para a busca de uma ordem total no espaço de cores. Ao invés de procurar por uma ordem total, nós procuramos por uma métrica adequada que defina um gradiente para imagens coloridas, que é um passo fundamental para o paradigma de segmentação morfológica.

6. Ref. [84] Automatic Design of Morphological Operators for Motion Segmentation.

   Um problema de interesse na edição de vídeos digitais é a eliminação de objetos em um vídeo e a introdução de pedaços de outros vídeos no lugar daqueles eliminados. Um problema fundamental para construir ferramentas para este propósito é a segmentação de objetos que se movem. Este artigo aborda este problema por uma técnica nova, baseada no paradigma de Beucher-Meyer, com marcadores detectados por operadores morfológicos projetados por aprendizado computacional (or, equivalentemente, por otimização estatística). Os objetos nos primeiros quadros do vídeo são marcados manualmente e usados para treinar o detector de marcadores. Então, o operador projetado é usado para marcar os objetos nos quadros seguintes e o paradigma é aplicado a todos os quadros marcado pelo detector. Alguns exemplos sintéticos e do mundo real ilustram a aplicação da técnica proposta. Situações complexas, como exclusão, são examinadas.

7. Ref. [79] Optimal Range-Domain Window Filters.

   O projeto não restrito de operadores digitais ótimos baseados em janela a partir de amostras de sinais é muito difícil por causa da inabilidade de obter dados suficientes para fazer boas estimativas dos parâmetros dos filtros. Este artigo estuda o problema de estimação por janelamento

nos níveis de cinza, assim como no espaço. Em cada ponto, o sinal é visto através de uma "aperture", que é o produto entre a janela no domínio e uma janela nos níveis de cinza. Os valores fora da "aperture" são projetados para o limite dos valores dentro da "aperture". Os experimentos mostram que os filtros "aperture" podem ser melhores que os filtros lineares para desembaçamento, especialmente na restauração das arestas. Uma amostra dos diversos experimentos que foram feitos para estudar estes efeitos dos filtros "aperture" para desembaçamento é apresentada.

8. Ref. [12] An OCR based on Mathematical Morphology

   Apresentamos neste artigo um protótipo de um OCR que foi projetado e implementado no Instituto de Matemática e Estatística da Universidade de São Paulo. A característica principal deste sistema é que todas as tarefas de processamento de imagens é feita por operadores da morfologia matemática (os chamados operadores morfológicos). Desta maneira, desenvolvemos operadores morfológicos para segmentar as imagens digitalizadas (isto é, identificar objetos como letras, palavras e parágrafos), e reconhecer o tipo de fonte e as letras. Os operadores morfológicos que fazem a segmentação foram projetados por técnicas heurísticas clássicas, enquanto os operadores que reconhecem os tipos de fontes e as letras foram projetados por técnicas de aprendizado computacional. A idéia fundamental por trás destas técnicas é a estimação dos operadores morfológicos a partir da observação de pares de imagens de entrada-saída, que descrevem sua performance ideal. Os operadores morfológicos projetados foram integrados em um sistema que traduza imagens de texto digitalizadas em textos no formato RTF, com razoável corretude e performance em termos de tempo. Este sistema foi desenvolvido com ajuda da plataforma KHOROS, usando as caixas de ferramentas MMach (para os operadores morfológicos projetados heuristicamente) e a PAC (para os operadores projetados por aprendizado).

Apêndice C

Lista complementar de artigos e relatórios

Apresentamos abaixo um lista complementar dos artigos (publicados ou submetidos) e relatórios nos quais participamos durante o período de doutorado compreendido de agosto de 1996 até a data da entrega desta tese.

C.1 Artigos de revista.

1. Ref. [164] Segmentation of Microarray Images by Mathematical Morphology

C.2 Artigos em conferência.

1. Ref. [165] Microarray Gridding by Mathematical Morphology
2. Ref. [166] Hybrid Human-machine Non-linear Filter Design Using Envelopes

C.3 Artigos submetidos.

1. Ref. [82] Nonlinear Filter Design Using Envelopes

C.4 Posters

1. Ref. [167] A role for YakA, cAMP and PKA in the nitrosoative/oxidative stress response of Dictyostelium discoideum cells.

C.5 Relatórios técnicos.

1. Ref. [168] An Integrated Environment for Storage and Analysis of Genetic Data.

Referências Bibliográficas

[1] N. D. A. Mascarenhas and F. R. D. Velasco. *Processamento Digital de Imagens.* Escola de Computação, IME-USP, 1984.

[2] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison-Wesley Publishing Company, 1992.

[3] W. K. Pratt. *Digital Image Processing.* John Wiley and Sons, 1991.

[4] R. Chellapa, C. L. Wilson, S.Sirohey, and C. S. Barnes. Human and Machine Recogntion of Faces: A Survey. Technical Report CAR-TR-731, CS-TR-3339, DACA76-92-C-0009, University of Maryland and NIST, August 1994.

[5] B. Laurel and S. J. Mountford, editors. *Art of Human-Computer Interface Design.* Addison-Wesley Pub Co, July 1990.

[6] R. P. Loce and E. R. Dougherty. *Enhancement and Restoration of Digital Documents: Statistical Design of Nonlinear Algorithms.* SPIE - The International Society for Optical Engineering, Bellingham, 1997.

[7] T. M. Mitchell. *Machine Learning.* McGraw-Hill Series in Computer Science. McGraw-Hill, March 1997.

[8] E. R. Dougherty. *Random Processes for Image and Signal Processing.* SPIE and IEEE Presses, Bellingham, 1999.

[9] S. Beucher and F. Meyer. *Mathematical Morphology in Image Processing*, chapter 12. The Morphological Approach to Segmentation: The Watershed Transformation, pages 433–481. Marcel Dekker, 1992.

[10] F. Meyer and S. Beucher. Morphological Segmentation. *Journal of Visual Communication and Image Representation*, 1(1):21–46, September 1990.

[11] J. Barrera, E. R. Dougherty, and M. Brun. Hybrid human-machine binary morphological operator design. An independent constraint approach. *Signal Processing*, 80(8):1469–1487, August 2000.

[12] J. Barrera, R. Terada, R. A. Lotufo, N. S. T. Hirata, R. Hirata Jr., and F. A. Zampirolli. An OCR based on Mathematical Morphology. In *Nonlinear Image Processing IX*, volume 3304 of *Proceedings of SPIE*, pages 197–208, San Jose, CA, January 1998.

[13] G. J. F. Banon and J. Barrera. Bases da Morfologia Matemática para Análise de Imagens Binárias. IX Escola de Computação, Pernambuco, Julho 1994.

[14] H. J. A. M. Heijmans. *Morphological Image Operators*. Academic Press, Boston, 1994.

[15] G. Matheron. *Random Sets and Integral Geometry*. John Wiley, 1975.

[16] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982.

[17] J. Serra. *Image Analysis and Mathematical Morphology. Volume 2: Theoretical Advances*. Academic Press, 1988.

[18] G. J. F. Banon and J. Barrera. Minimal Representations for Translation-Invariant Set Mappings by Mathematical Morphology. *SIAM J. Applied Mathematics*, 51(6):1782–1798, December 1991.

[19] H. J. A. M. Heijmans and C. Ronse. The Algebraic Basis of Mathematical Morphology – Part I: Dilations and Erosions. *Computer Vision, Graphics and Image Processing*, 50:245–295, 1990.

[20] C. Ronse and H. J. A. M. Heijmans. The Algebraic Basis of Mathematical Morphology – Part II: Openings and Closings. *Computer Vision, Graphics and Image Processing: Image Understanding*, 54:74–97, 1991.

[21] G. J. F. Banon and J. Barrera. Decomposition of Mappings between Complete Lattices by Mathematical Morphology, Part I. General Lattices. *Signal Processing*, 30:299–327, 1993.

[22] R. M. Haralick, S. R. Sternberg, and X. Zhuang. Image Analysis Using Mathematical Morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(4):532–550, July 1987.

[23] J. Serra and L. Vincent. An Overview of Morphological Filtering. *Circuits Systems Signal Process*, 11(1):47–108, 1992.

[24] L. Vincent and P. Soille. Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, June 1991.

[25] L. Vincent. Morphological Grayscale Reconstruction in Image Analysis: Applications and Efficient Algorithms. *IEEE Transactions on Image Processing*, 2(2):176–201, April 1993.

[26] I. Molchanov. *Statistics of the Boolean Model for Practioners and Mathematicians*. John Wiley and Sons, 1997.

[27] J. Goutsias. Morphological Analysis of Discrete Random Shapes. *Journal of Mathematical Imaging and Vision*, 2:193–215, 1992.

[28] L. Devroye. Automatic Pattern Recognition: A Study of the Probability of Error. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):530–543, July 1988.

[29] K. Fukunaga and R. R. Hayes. Effects of Sample Size in Classifier Design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(8), August 1989.

[30] S. J. Raudys and A. K. Jain. Small Sample Size Effects in Statistical Pattern Recognition : Recommendations for Practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):252–264, March 1991.

[31] L. Devroye, L. Györffi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer-Verlag, 1996.

[32] E. R. Dougherty. *Probability and Statistics for the Engineering, Computing and Physical Sciences.* Prentice-Hall, 1990.

[33] A. F. Kohn. *Reconhecimento de Padrões - Uma Abordagem Estatística.* EPUSP, 1999.

[34] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, 1973.

[35] L. G. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[36] D. Haussler. Decision Theoretic Generalizations of the PAC Model for Neural Net and Other Learning Applications. *Information and Computation*, 100:78–150, 1992.

[37] M. Anthony and N. Biggs. *Computational Learning Theory - An Introduction.* Cambridge University Press, 1992.

[38] H. Drucker, R. Schapire, and P. Simard. Boosting Performance in Neural Networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):705–719, 1993.

[39] D. H. Wolpert, editor. *The Mathematics of Generalization.* Addison-Wesley, 1995. The Proceedings of the SFI/CNLS Workshop on Formal Approaches to Supervised Learning.

[40] M. Perkowski, L. Jozwiak, and S. Mohamed. New Approach to Learning Noisy Boolean Functions. In *Proc. ICCIMA'98 Conference*, pages 693–706, Australia, February 1998. World Scientific.

[41] D. E. Knuth. *The Art of Computer Programming*, volume 3 / Sorting and Searching. Addison-Wesley, 1973.

[42] B. Carré. *Graphs and Networks.* Oxford University Press, 1979.

[43] M. R. Garey and D. S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, San Francisco, 1979.

[44] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* McGraw-Hill, 1990.

[45] R. S. Pressman. *Software Engineering : A Practitioner's Approach.* Series in Computer Science. McGraw-Hill, 1994.

[46] E. F. Harding and D. G. Kendall, editors. *Stochastic Geometry.* Wiley, 1974.

[47] J. Barrera, G. J. F. Banon, R. A. Lotufo, and R. Hirata Jr. MMach: a Mathematical Morphology Toolbox for the Khoros System. *Electronic Imaging*, 7(1):174–210, 1998.

[48] P. Maragos and R. W. Schafer. Morphological Filters: Part I: Their Set-Theoretic Analysis and Relations to Linear Shift-Invariant Filters. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35:1153–1169, August 1987.

[49] P. Maragos and R. W. Schafer. Morphological Filters: Part II: Their Relations to Median, Order Statistic, and Stack-Filters. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35:1170–1184, August 1987. Corrections in Vol. ASSP-37, April 1989, p. 597.

[50] E. R. Dougherty and C. R. Giardina. *Morphological Methods in Image and Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1988.

[51] E. R. Dougherty. Optimal Mean-Square N-Observation Digital Morphological Filters I. Optimal Binary Filters. *CVGIP: Image Understanding*, 55(1):36–54, January 1992.

[52] E. R. Dougherty and M. Haralick. Unification of Nonlinear Filtering in the Context of Binary Logical Calculus, Part I: Binary Filters. *J. Math. Imaging Vision*, 2(2):173–183, 1992.

[53] E. R. Dougherty and R. P. Loce. Optimal Mean-Absolute-Error Hit-or-Miss Filters: Morphological Representation and Estimation of the Binary Conditional Expectation. *Optical Engineering*, 32(4):815–827, April 1993.

[54] J. Barrera, E. R. Dougherty, and N. S. Tomita. Automatic Programming of Binary Morphological Machines by Design of Statistically Optimal Operators in the Context of Computational Learning Theory. *Electronic Imaging*, 6(1):54–67, January 1997.

[55] E. R. Dougherty and R. P. Loce. Precision of Morphological-Representation Estimators for Translation-invariant Binary Filters: Increasing and Nonincreasing. *Signal Processing*, 40:129–154, 1994.

[56] J. Barrera, E. R. Dougherty, and N. S. T. Hirata. Design of Optimal Morphological Operators from Prior Filters. *Acta Steriologica*, 16(3):193–200, 1997. Special issue on Mathematical Morphology.

[57] E. R. Dougherty and J. Barrera. Bayesian Design of Optimal Morphological Operators Based on Prior Distributions for Conditional Probabilities. *Acta Stereologica*, 16(3):167–174, 1997.

[58] E. R. Dougherty and R. P. Loce. Optimal Binary Differencing Filters: Design, Logic Complexity, Precision Analysis, and Application to Digital Document Processing. *Electronic Imaging*, 5(1):66–86, January 1996.

[59] O. V. Sarca, E. R. Dougherty, and J. Astola. Secondarily Constrained Boolean Filters. *Signal Processing*, 71(3):247–263, December 1998.

[60] O. V. Sarca, E.R. Dougherty, and J. Astola. Two-stage Binary Filters. *Electronic Imaging*, 8(3):219–232, July 1999.

[61] E. R. Dougherty. Optimal Mean-Square N-Observation Digital Morphological Filters II. Optimal Gray-Scale Filters. *CVGIP: Image Understanding*, 55(1):55–72, January 1992.

[62] J. Barrera and E. R. Dougherty. Representation of Gray-Scale Windowed Operators. In Henk J.A.M. Heijmans and Jos B.T.M. Roerdink, editors, *Mathematical Morphology and its Applications to Image and Signal Processing*, volume 12 of *Computational Imaging and Vision*, pages 19–26. Kluwer Academic Publishers, Dordrecht, May 1998.

[63] P. Salembier. Structuring element adaptation for morphological filters. *Visual Communication and Image Representation*, 3(2):115–136, 1992.

[64] C. Cuicurean-Zapan, E. R. Dougherty, and Y. Chen. Behavior of adaptive digital erosions. In *Proc. SPIE Statistical and Stochastic Methods for Image Processing*, July 1996.

[65] J. Barrera and G. P. Salas. Set Operations on Closed Intervals and Their Applications to the Automatic Programming of Morphological Machines. *Electronic Imaging*, 5(3):335–352, July 1996.

[66] E. R. Dougherty and D. Sinha. Computational Gray-Scale Mathematical Morphology on Lattices (A Comparator-based Image Algebra) Part I: Architecture. *Real-Time Imaging*, 1:69–85, 1995.

[67] E. R. Dougherty and D. Sinha. Computational Gray-Scale Mathematical Morphology on Lattices (A Comparator-based Image Algebra) Part II: Image Operators. *Real-Time Imaging*, 1:283–295, 1995.

[68] J. Barrera, R. Terada, R. Hirata Jr, and N. S. T. Hirata. Automatic Programming of Morphological Machines by PAC Learning. *Fundamenta Informaticae*, 41(1-2):229–258, January 2000.

[69] P. D. Wendt, E. J. Coyle, and N. C. Gallagher Jr. Stack Filters. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-34(4):898–911, August 1986.

[70] E. J. Coyle and J.-H. Lin. Stack Filters and the Mean Absolute Error Criterion. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36(8):1244–1254, August 1988.

[71] L. Yin, J. T. Astola, and Y. A. Neuvo. Optimal weighted order statistic filters under the mean absolute error criterion. In *Proc. IEEE Conf. Acoustics, Speech, and Signal Processing*, volume 4, pages 2529–2532, 1991.

[72] L. Yin, J. T. Astola, and Y. A. Neuvo. Filtering with applications to image processing. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 41:162–184, 1993.

[73] Nina S. T. Hirata. *Projeto Automático de Operadores – Explorando Conhecimentos a Priori.* PhD thesis, Instituto de Matemática e Estatística da USP, São Paulo, Brazil, October 2000.

[74] N. S. T. Hirata, J. Barrera, and E. R. Dougherty. Design of Statistically Optimal Stack Filters. In J. Stolfi and C. L. Tozzi, editors, *Proc. of Sibgrapi'99*, pages 265–274, Campinas, SP, Brazil, 1999.

[75] E. R. Dougherty. Unification of Nonlinear Filtering in the Context of Binary Logical Calculus, Part II: Gray-Scale Filters. *J. Math. Imaging Vision*, 2(2):185–192, 1992.

[76] I. Pitas and A. N. Venetsanopoulos. *Nonlinear Digital Filters – Principles and Applications.* Kluwer Academic Publishers, 1990.

[77] R. Hirata Jr., Nina S. T. Hirata, Junior Barrera, and Edward R. Dougherty. Two Extensions to the ISI Algorithm. *In preparation*, 2002.

[78] R. Hirata Jr., E. R. Dougherty, and J. Barrera. Aperture Filters. *Signal Processing*, 80(4):697–721, April 2000.

[79] R. Hirata Jr., E. R. Dougherty, and J. Barrera. Optimal Range-Domain Window Filters. In E. R. Dougherty and J. T. Astola, editors, *Nonlinear Image Processing X*, volume 3646 of *Proc. of SPIE*, pages 38–45, San Jose, CA, January 1999.

[80] O. V. Sarca, E.R. Dougherty, and J. Astola. Optimal Binary Filters with Linearly Separable Preprocessing. In *Nonlinear Image Processing*, Proc. of SPIE, January 1998.

[81] R. Hirata Jr., Marcel Brun, Junior Barrera, and Edward R. Dougherty. Image restoration by multiresolution aperture filters. In *Nonlinear Image Processing and Pattern Analysis XII*, volume 4304 of *Proceedings of SPIE*, pages 256–264, January 2001.

[82] Marcel Brun, R. Hirata Jr., Junior Barrera, and Edward R. Dougherty. Nonlinear Filter Design Using Envelopes. *submitted*, 2001.

[83] R. Hirata Jr. Segmentação de Imagens por Morfologia Matemática. Master's thesis, Instituto de Matemática e Estatística - USP, março 1997.

[84] R. Hirata Jr., J. Barrera, F. C. Flores, and R. A. Lotufo. Automatic Design of Morphological Operators for Motion Segmentation. In J. Stolfi and C. L. Tozzi, editors, *Proc. of Sibgrapi'99*, pages 283–292, Campinas, SP, Brazil, 1999.

[85] R. Hirata Jr., Edward R. Dougherty, and Junior Barrera. Some applications of aperture filters. In Luc Vincent, editor, *Mathematical Morphology and its Applications to Image Processing*. Kluwer Academic, Dordrecht. The document in this repository is the manuscript submitted to ISMM'2000.

[86] F. C. Flores, R. Hirata Jr., J. Barrera, R. A. Lotufo, and F. Meyer. Morphological Operators for Segmentation of Color Sequences. In *Proceedings of SIBGRAPI'2000*, pages 300–307, Gramado, Brazil, October 2000.

[87] R. Hirata Jr., F. C. Flores, J. Barrera, R. A. Lotufo, and F. Meyer. Color Image Gradients for Morphological Segmentation. In *Proceedings of SIBGRAPI'2000*, pages 316–322, Gramado, Brazil, October 2000.

[88] K. Konstantinides and J. Rasure. The KHOROS Software Development Environment for Image and Signal Processing. *IEEE Transactions on Image Processing*, 3(3):243–252, 1994.

[89] A. Rosenfeld. *Picture Processing by Computer*. Academic Press, 1969.

[90] Ernest L. Hall. *Computer Image Processing and Recognition*. Academic Press, 1979.

[91] G. A. Baxes. *Digital Image Processing - Principles and Applications*. John Wiley and Sons, Inc., 1994.

[92] Charles W. Therrien. *Discrete random signals and statistical signal processing*. Englewood Cliffs, NJ : Prentice Hall, 1992.

[93] M. H. DeGroot. *Probability and Statistics*. Addison-Wesley, 2nd edition, 1989.

[94] H. Stark and J. W. Woods. *Probability, Random Processes, and Estimation Theory for Engineers*. Prentice Hall, Englewood Cliffs, 1986.

[95] I. I. Gikhman and A. V. Skorokhod. *Introduction to the theory of random processes.* Dover, 1996.

[96] B. D. Ripley. *Spatial Statistics.* Wiley Series in Probability and Mathematical Statistics. John-Wiley, 1981.

[97] B. D. Ripley. *Statistical Inference for Spatial Processes.* Cambridge University Press, 1988.

[98] G. L. Gaile and C. J. Willmott, editors. *Spatial Statistics and Models*, volume 40. D. Reidel Publishing Company, 1984.

[99] Kenneth Hoffman and Ray A. Kunze. *Álgebra Linear.* Livros Técnicos e Científicos, 1979., 1979.

[100] Larry Smith. *Linear algebra.* New York : Springer-Verlag, 1984.

[101] Norbert Wiener. *Extrapolation, interpolation, and smoothing of stationary time series, with engineering applications.* New York, Wiley, 1949.

[102] Norbert Wiener. *Nonlinear problems in random theory.* Technology Press research monographs, 1963.

[103] V. S. Pugachev. *Theory of Random Functions and Its Application to Control Problems.* Pergamon Press, 1965. Revised translation by O. M. Blunn.

[104] I. I. Gikhman and A. V. Skorokhod. *Introduction to the Theory of Random Processes.* Dover, 1996.

[105] G. Birkhoff. *Lattice Theory.* American Mathematical Society, Providence, Rhode Island, 3rd edition, 1967.

[106] J. O. Berger. *Statistical Decision Theory and Bayesian Analysis.* Springer Series in Statistics. Springer-Verlag, 2nd edition, 1985.

[107] E. R. Dougherty and J. Barrera. Computatinal Gray-Scale Operators. In E. R. Dougherty and J. T. Astola, editors, *Nonlinear Filters for Image Processing*, pages 61–98. SPIE and IEEE Press, Bellingham, 1999.

[108] Gerald Jean Francis Banon. Formal introduction to digital image processing. Deposited in the URLib collection., 2000. Second edition. This material is used as class notes for an INPE posgraduate course. This work has been supported by CNPq under contract 300966/90-3.

[109] E. R. Dougherty and D. Sinha. Computational Mathematical Morphology. *Signal Processing*, 38:21–29, 1994.

[110] D. Heath. *A Geometric Framework for Machine Learning.* PhD thesis, The Johns Hopkins University, Baltimore, Maryland, 1992.

[111] S. C. Port. *Theoretical Probability for Applications.* Wiley Series in Probability and Mathematical Statistics. John-Wiley, 1994.

[112] S. Haykin. *Neural Networks – A Comprehensive Foundation.* Mcmillan, 1994.

[113] C. B. Herwig and R. J. Schalkoff. Morphological Image Processing Using Artificial Neural Networks. *Control and Dynamic Systems*, 67, 1994.

[114] M. Di Martino, S. Fanelli, and M. Protasi. Exploring and Comparing the Best Direct Methods for the Efficient Training of MLP-Networks. *IEE Trans. on Neural Networks*, 7(6):1497–1502, November 1996.

[115] M. T. Musavi, K.H. Chan, D. M. Hummels, and K. Kalantri. On the Generalization Ability of Neural Network Classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):659–663, June 1994.

[116] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. The Wadsworth Statistics/Probability Series. Wadsworth International Group, 1984.

[117] S. K. Murthy, S. Kasif, and S. Salzberg. A System for Induction of Oblique Decision Trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994.

[118] N. S. Tomita. Programação Automática de Máquinas Morfológicas Binárias baseada em Aprendizado PAC. Master's thesis, Instituto de Matemática e Estatística - Universidade de São Paulo, São Paulo, SP - Brasil, março 1996.

[119] V. N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998.

[120] Bernhard Scholkopf, Christopher J. C. Burges, and Alex J. Smola. *Advances in Kernel Methods*. MIT Press, 1998.

[121] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[122] A. Gammerman, editor. *Computational Learning Theory and Probabilistic Reasoning*. John Wiley and Sons, 1996.

[123] J. R. Quinlan. Simplifying Decision Trees. *Int. J. Man-Machine Studies*, 27:221–234, 1987.

[124] D. Heath, S. Kasif, and S. Salzberg. Induction of Oblique Decision Trees. Technical report, Johns Hopkins Univeristy, April 1995. 32 pages.

[125] S. Kirkpatrick, C. D. Gellat Jr., and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, May 1983.

[126] N. S. T. Hirata, E. R. Dougherty, and J. Barrera. Iterative Design of Morphological Binary Image Operators. *Optical Engineering*, 39(12):3106–3123, December 2000.

[127] M. Schmitt and F. Preteux. *Image Analysis and Mathematical Morphology*, volume 2, chapter Boolean texture analysis and synthesis. Ed. J. Serra, Academic Press, New York, 1988.

[128] L. Atlas, R. Cole, Y. Muthusamy, A. Lippman, J. Connor, D. Park, M. El-Sharkawi, and R. J. Marks II. A Performance Comparison of Trained Multilayer Perceptrons and Trained Classification Trees. In *Proceedings of the IEEE*, volume 78, pages 1614–1619, October 1990.

[129] N. S. T. Hirata, J. Barrera, and R. Terada. Text Segmentation by Automatically Designed Morphological Operators. In *Proc. of SIBGRAPI'2000*, pages 284–291, Gramado, Brazil, October 2000.

[130] J. Barrera, N. S. Tomita, F. S. C. Silva, and R. Terada. Automatic Programming of Binary Morphological Machines by PAC Learning. In *Neural and Stochastic Methods in Image and Signal Processing*, volume 2568 of *Proc. of SPIE*, pages 233–244, San Diego, 1995.

[131] N. S. T. Hirata, J. Barrera, and E. R. Dougherty. Boolean Function Minimization by Incremental Splitting of Intervals. *submitted*, 2001.

[132] F. J. Hill and G. R. Peterson. *Introduction to Switching Theory and Logical Design.* John Wiley, 3rd edition, 1981.

[133] H. R. Andersen. An Introduction to Binary Decision Diagrams. Technical report, Department of Information Technology, Technical University of Denmark, December 1994.

[134] K. S. Brace, R. L. Rudell, and R. E. Bryant. Efficient Implementation of a BDD Package. In *Proceedings of 27th ACM/IEEE Design Automation conference*, pages 40–45, 1990.

[135] Y. Hong, P. A. Beerel, J. R. Burch, and K. L. McMillan. Safe BDD Minimization Using Don't Cares. In *Proc. of 34th DAC*, pages 208–213, Anaheim, CA, 1997.

[136] J. Jain, J. Bitner, M. Abadir, J. A. Abraham, and D. S. Fussell. Indexed BDDs: Algorithmic Advances in Techniques to Represent and Verify Boolean Functions. 1994.

[137] C. Meinel and Anna Slobodová. A Unifying Theoretical Background for Some BDD-based Data Structures. Technical Report 94-17, Informatik, Universtät Trier, Trier, 1994.

[138] J. V. Sanghavi, R. K. Ranjan, R. K. Brayton, and A. S. Vincentelli. High Performance BDD Package by Exploiting Memory Hierarchy. In *Proc. of 33rd DAC*, pages 635–640, Las Vegas, NE, June 1996.

[139] H. M. F. Madeira, J. Barrera, R. Hirata Jr, and N. S. T. Hirata. A New Paradigm for the Architecture of Morphological Machines : Binary Decision Diagrams. In J. Stolfi and C. L. Tozzi, editors, *Proc. SIBGRAPI'99 - XII Brazilian Symposium in Computer Graphics and Image Processing*, pages 283–292, Campinas, SP, Brazil, November 1999.

[140] H. M. F. Madeira and J. Barrera. Incremental Evaluation of BDD-Represented Set Operators. In *Proceedings of the XIII Brazilian Symposium in Computer Graphics and Image Processing*, Gramado, October 2000. IEEE.

[141] N. S. T. Hirata, E. R. Dougherty, and J. Barrera. A Switching Algorithm for Design of Optimal Increasing Binary Filters Over Large Windows. *Pattern Recognition*, 33(6):1059–1081, June 2000.

[142] E. R. Dougherty, J. Barrera, G. Mozelle, S. Kim, and M. Brun. Multiresolution Analysis for Optimal Binary Filters. *Mathematical Imaging and Vision*, (14):53–72, 2001.

[143] R. P. Loce and E. R. Dougherty. Optimal Morphological Restoration: The Morphological Filter Mean-Absolute-Error Theorem. *Visual Communication and Image Representation*, 3(4):412–432, December 1992.

[144] R. P. Loce and E. R. Dougherty. Facilitation of Optimal Binary Morphological Filter Design Via Structuring Element Libraries and Design Constraints. *Optical Engineering*, 31(5):1008–1025, May 1992.

[145] R. P. Loce and E. R. Dougherty. Mean-Absolute-Error representation and Optimization of Computational-Morphological Filters. *Graphical Models and Image Processing*, 57(1):27–37, 1995.

[146] J. T. Astola and P. Kuosmanen. Representation and optimization of stack filters. In E. R. Dougherty and J. T. Astola, editors, *Nonlinear Filters for Image Processing*, pages 237–279. SPIE and IEEE Press, Bellingham, 1999.

[147] P. Kuosmanen and J. T. Astola. Optimal stack filters under rank selection and structural constraints. *Signal Processing*, 41:309–338, 1995.

[148] M. Gabbouj and E. J. Coyle. Minimum Mean Absolute Error Stack Filtering with Structural Constraints and Goals. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(6):955–968, June 1990.

[149] E. J. Coyle, J.-H. Lin, and M. Gabbouj. Optimal Stack Filtering and the Estimation and Structural Approaches to Image Processing. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(12):2037–2066, December 1989.

[150] I. Tăbuş, D. Petrescu, and M. Gabbouj. A training Framework for Stack and Boolean Filtering – Fast Optimal Design Procedures and Robustness Case Study. *IEEE Transactions on Image Processing*, 5(6):809–826, June 1996.

[151] E. R. Dougherty, editor. *Digital Image Processing Methods.* Marcel Dekker, 1994.

[152] D. E. Knuth. Digital Halftones by Dot Diffusion. *ACM Trans. on Graphics*, 6(4):245–273, October 1987.

[153] Jonas Gomes and Luiz Velho. *Computação Gráfica: Imagem.* IMPA - SBM, 1994.

[154] C. Finch. *Special Effects: Creating Movie Magic.* Abbeville Press, 1984.

[155] T. Poggio. Early Vision: from computational structure to algorithms and parallel hardware. *CVGIP*, 31(1):139–155, 1985.

[156] F. C. Flores. Segmentação de Seqüências de Imagens por Morfologia Matemática. Master's thesis, Instituto de Matemática e Estatística - Universidade de São Paulo, October 2000.

[157] P. Salembier and M. Pardàs. Hierarchical Morphological Segmentation for Image Sequence Coding. *IEEE Transactions on Image Processing*, 3(5):639–651, September 1994.

[158] S. Beucher. Watersheds of Functions and Picture Segmentation. In *ICASSP 82, Proc. IEEE Intern. Conf. on Acoustics, Speech and Signal Processing*, pages 1928–1931, Paris, May 1982.

[159] P. Soille and L. Vincent. Determining Watersheds in Digital Pictures via Flooding Simulations. In *Visual Communications and Image Processing*, pages 240–250. SPIE, 1990. volume 1360.

[160] F. Meyer. Skeletons and Perceptual Graphs. *Signal Processing*, 16:335–363, 1989.

[161] R. Hirata Jr., Marcel Brun, Junior Barrera, and Edward R. Dougherty. Multiresolution design of aperture operators. *To appear in Journal of Mathematical Imaging and Vision*, 2001.

[162] R. Hirata Jr., E. R. Dougherty, and J. Barrera. Design of gray-scale nonlinear filters via multiresolution apertures. In *Proceedings of EUSIPCO 2000*, volume IV, pages 1905–1908, 2000.

[163] R. Hirata Jr., E. R. Dougherty, and J. Barrera. Some Applications of Aperture Filters. In J. Goutsias, L. Vincent, and D. S. Bloomberg, editors, *Mathematical Morphology and its Applications to Image and Signal Processing*, volume 18, pages 119–128. Kluwer Academic Publishers, 2000. Fifth ISMM.

[164] R. Hirata Jr., J. Barrera, R. F. Hashimoto, D. O. Dantas, and G. P. Esteves. Segmentation of Microarray Images by Mathematical Morphology. *To appear in Real Time Imaging.*

[165] R. Hirata Jr., J. Barrera, R. F. Hashimoto, and D. O. Dantas. Microarray Gridding by Mathematical Morphology. In D. L. Borges and S.-T. Wu, editors, *Proceedings of Sibgrapi 2001*, pages 112–119, Florianopolis, Brasil, October 2001. IEEE.

[166] M. Brun, R. Hirata Jr., J. Barrera, and E. R. Dougherty. Hybrid Human-machine Non-linear Filter Design Using Envelopes. In D. L. Borges and S.-T. Wu, editor, *Proceedings of Sibgrapi 2001*, pages 106–111, Florianopolis, Brasil, October 2001. IEEE.

[167] Raquel Bagattini, Renata Gorjão, Alexandre Taminato, Nancy Van Driessche, Guokai Chen, Thiago Teixeira Santos, Gustavo Tadao Okida, Roberto Hirata Jr., João Eduardo Ferreira, Eduardo Jordão Neves, Junior Barrera, Adam Kuspa, Gad Shaulsky, and Glaucia Mendes Souza. A role for yaka, camp and pka in the nitrosoative/oxidative stress response of dictyostelium discoideum cells. In *International Dictyostelium Conference*, La Jolla, California, USA, July 2001.

[168] J Barrera, R. M. César Jr., J. E. Ferreira, M.D. Gubitoso, N. S. T. Hirata, R. Hirata Jr., and E. J. Neves. An Integrated Environment for Storage and Analysis of Genetic Data. Technical Report RT-BIOINFO-2000-01-R, Núcleo de Pesquisa em Bioinformática - USP, Setembro 2000.