
**Universidade Católica Dom Bosco**
Centro de Ciências Exatas e Tecnológicas
Curso de Engenharia de Computação

# Detecção de Bordas utilizando Informações sobre Textura e Cor

Daniel de Azevedo Scalabrini

Profa. Orientadora: Priscila Silva Martins, M. Eng.

*Relatório Final submetido como um dos requisitos para a obtenção do grau de Engenheiro de Computação.*

UCDB - Campo Grande - MS - Novembro/2005

# Resumo

O propósito geral de um sistema de processamento de imagens é reconhecer objetos em uma cena. Tipicamente, um dos primeiros passos de um sistema deste tipo é a detecção das bordas. Pesquisadores se concentraram durante décadas desenvolvendo algoritmos para o processamento de imagens em tons de cinza. Com o avanço da tecnologia e o aumento da capacidade de processamento dos computadores, agora é possível utilizar as informações de cor e textura das imagens para obter melhores resultados em tais processamentos. Diversos algoritmos de detecção de bordas que utilizam este novo conjunto de informações foram propostos, e diversos artigos nesta área foram publicados. Este trabalho apresenta um estudo de diversas técnicas de detecção de bordas, com o objetivo de gerar um módulo de detecção de bordas, que utilize informações de cor e textura das imagens, que possa ser incorporado ao sistema DTCouro, que se trata de um sistema automático de extração de parâmetros numéricos, a partir de imagens digitais, para tornar mais eficiente e preciso o processo de classificação do couro bovino.

# Abstract

The basic purpose of a image system is to recognize objects in a scene. Usually, the first steps of a system like that is the edges detection. Researchers focused for centuries developing algorithms to process the imagens in gray scale. With the avance of thecnology and the increasing capacity of computer's processing way, now is possible to use the informations of color and texture of images to obtain better results in such processing methods. Many algorithms of edge detection which use this new set of information were considered, and diverse articles in this area were published. This work represents an study of edges detection, with the purpose of generating an edges' detection module that uses color informaiton and images's texture wich can be be combined to DTCouro system, which is about an automatic system of numeric parameters extraction, from digital images, to make more efficient and precise the classification process of the ox's leather.

# Conteúdo

# Lista de Figuras

# Lista de Tabelas

# Lista de Algoritmos

# Capítulo 1

# Introdução

O setor coureiro do nosso estado tem seu potencial prejudicado pela baixa qualidade do couro bovino disponível no mercado interno. Isto dificulta a possibilidade de que o país possa agregar valor ao produto e de usufruir das riquezas e oportunidades proporcionadas por este setor [47].

Alguns levantamentos realizados indicam que cerca de 60% dos defeitos no couro são provenientes do manejo dos animais na propriedade rural [17] [26] [6]. Ao longo da cadeia de produção ocorrem os outros 40%, decorrentes de pontas de pregos e parafusos e de lascas de madeira presentes na carroceria dos caminhões, do uso de ferrões no manejo do gado, ou ainda da esfola e salga mal conduzidas [26].

O principal fator limitante à melhoria da qualidade do couro é a inexistência de sistemas de remuneração diferencial pela qualidade da matéria-prima produzida [31], possíveis de serem estabelecidos a partir de programas de classificação de couros e peles. Atualmente, no sistema brasileiro, o pecuarista recebe um valor fixo, muito baixo, pelo couro, independente de sua qualidade [18].

Pesquisas realizadas nesta área visam promover o desenvolvimento socioeconômico e tecnológico da cadeia produtiva de couros e derivados, servindo como alavanca para reafirmar o setor coureiro como um grande potencial gerador de divisas e emprego. Alguns esforços têm sido conduzidos no sentido de superar esses obstáculos, a exemplo do Programa Brasileiro de Melhoria do couro crú, com enfoque voltado para a questão da capacitação de pessoal ou ainda o Programa de Classificação da Qualidade do Couro, com o intuito de valorizar o couro produzido com maior qualidade [36].

A demanda por soluções tecnológicas definitivas para a problemática da qualidade do couro produzido reforçou a proposta de se criar um sistema de classificação de couro verde oficial e harmonizado em base de defeitos, em resposta a essa demanda o Ministério da Agricultura, Pecuária e Abastecimento

estabeleceu por meio da Instrução Normativa Nº 12, em 18 de dezembro de 2002 [35], os critérios de classificação de couro bovino conforme os defeitos presentes na pele do animal.

A partir de então, torna-se interessante o desenvolvimento de um sistema de Visão Computacional, voltado à extração de informações que sejam relevantes no contexto da classificação do couro. Este sistema de classificação deverá ser capaz de detectar as diversas inconformidades que possam ser apresentadas no couro, tais como, carrapatos, bernes, placas de berne, perfurações, marcas, cicatrizes, riscos entre outros, e de dar o parecer final sobre a classificação da peça de couro na sua totalidade. A nomenclatura DTCouro foi atribuída ao projeto que tem como objetivo geral o desenvolvimento de um sistema automático de extração de parâmetros numéricos, a partir de imagens digitais, para tornar mais eficiente e preciso o processo de classificação do couro bovino, além de um modulo de classificação automática do couro, alimentado pelos parâmetros extraídos das imagens, mas com regras de classificação definidas pelo usuário.

Tipicamente, um dos primeiros passos de um sistema deste tipo é a detecção das bordas. Os algoritmos de detecção de bordas normalmente detectam as transições de formas em uma imagem. Estas transições são características das bordas dos objetos. Uma vez que as bordas são detectadas, outras etapas do processamento podem ser executadas, tais como, segmentação da imagem, reconhecimento e contagem de padrões ou objetos, entre outros.

Várias técnicas de detecção de bordas em imagens coloridas foram introduzidas e muitos artigos tratam sobre este assunto. Cada uma dessas técnicas se baseia em algum tipo de medida de diferença, que, na maioria das vezes, é escolhida com base no espaço de cor que está sendo utilizado [56]. Existe uma grande possibilidade de se obter melhores resultados na detecção de bordas através da união de duas ou mais técnicas, e este será o foco principal deste trabalho.

## 1.1 Objetivos do Trabalho

O objetivo deste projeto consiste em desenvolver um módulo de detecção de bordas que utilize informações de textura e cor da imagem, para ser incorporado ao sistema de detecção de imperfeições em couro DTCouro.

Os objetivos específicos são:

1. Revisão teórica dos conceitos de textura e cores;
2. Aprofundar o conhecimento de pré-requisitos do ambiente de desenvolvimento;

3. Estudo de técnicas de detecção de bordas utilizando cor;
4. Estudo de técnicas de detecção de bordas utilizando textura;
5. Comparação dos algoritmos de detecção de bordas estudados;
6. Implementação de um algoritmo de detecção de bordas utilizando as informações de cor e textura;
7. Produção de documentação.

## 1.2 Justificativas

O estado de Mato Grosso do Sul é atualmente um dos maiores produtores de gado de corte do país, gerando assim uma grande quantidade de matéria prima para confecção de artefatos em couro. Porém a baixa qualidade do couro bovino produzido no estado vem prejudicando, financeiramente, o crescimento do setor coureiro. Acredita-se que o Brasil deixe de ganhar cerca de US$ 500 milhões/ano em função da baixa qualidade do couro [36]. O principal fator limitante da melhoria da qualidade do couro é a inexistência de sistemas de remuneração diferencial pela qualidade da matéria-prima, possíveis de serem estabelecidos a partir de programas de classificação de couros e peles.

O desenvolvimento e implantação de um sistema de detecção de imperfeições em couro bovino é justificado por incentivar os produtores a elevar a qualidade do couro por eles produzido, viabilizar a implantação de um sistema de remuneração ao produtor pela qualidade do couro, possibilitar que seja agregado valor ao couro bovino dentro do estado, favorecendo portanto, o crescimento econômico do setor coureiro. Além de aumentar o estoque de conhecimento nas áreas de qualidade de couro, processamento industrial de peles, relações mercadológicas e computação/automação aplicada ao agronegócio, buscamos incentivar o estudo e aplicação das técnicas de visão computacional aplicadas a diversas outras áreas, estabelecendo assim, novas vagas no mercado de trabalho, principalmente para profissionais especializados na implantação e operação de sistemas automáticos.

Pesquisadores se concentraram durante décadas desenvolvendo sistemas e algoritmos para processar imagens em tons de cinza, devido ao seu custo computacional relativamente baixo quando comparado ao custo de processamento de imagens coloridas. Com o avanço da tecnologia e o aumento da capacidade de processamento dos computadores, é agora possível utilizar as informações de cor e textura das imagens para tais processamentos. Um

dos benefícios é o aumento da quantidade de informações disponível para o processamento da imagem, proporcionando um melhor resultado para os problemas onde apenas a utilização de imagens monocromáticas não era satisfatória [56].

## 1.3 Organização do Texto

O texto está organizado da seguinte maneira: no Capítulo 2 está a fundamentação teórica, neste capítulo são explorados os conceitos relacionados ao tema de detecção de bordas e várias técnicas sobre o assunto são discutidas. O Capítulo 3 retrata os detalhes de implementação, enquanto o Capítulo 4 detalha os experimentos e a análise de resultados. As considerações finais são apresentadas no Capítulo 5. No Anexo A, estão incluídas as imagens resultantes dos experimentos realizados.

# Capítulo 2

# Fundamentação Teórica

## 2.1 Detecção de Bordas

Uma borda é o contorno entre um objeto e o fundo, indicando o limite entre objetos sobrepostos. Computacionalmente, bordas são definidas como picos da magnitude do gradiente, ou seja, são variações bruscas que ocorrem ao longo de curvas baseadas nos valores do gradiente (valor que quantifica a intensidade da variação entre o pixel e seus vizinhos) da imagem. As bordas são regiões da imagem onde ocorre uma mudança de intensidade em um certo intervalo do espaço, em uma certa direção [54].

A detecção de bordas é uma técnica bastante utilizada pela visão humana no reconhecimento de objetos. É o processo de localização e realce dos pixels de borda, utilizando para isso da informação da variação dos valores de luminosidade dos pixels da imagem [20].

Como as imagens possuem duas dimensões: altura e largura, as mudanças de intensidade ocorrem seguindo essas duas linhas de orientação, onde a orientação é uma característica importantíssima quando falamos em bordas bidimensionais. A Figura 2.1 ilustra a variação do contraste ao longo da linha de orientação, definindo assim uma borda [54].

A detecção de bordas em imagens é interessante por vários motivos. Pode-se identificar diferentes objetos na imagem através da detecção das bordas dos mesmos. Como exemplo, podemos imaginar uma imagem onde estão representados uma zebra e um leopardo. Sabemos que a zebra possui listras em sua pelagem ao mesmo passo que o leopardo possui manchas. Através da detecção das bordas, aplicada a esta imagem, pode-se facilmente diferenciar a zebra do leopardo, visto que as formas dos desenhos de suas pelagens são bastante distintas [13].

Os pontos da imagem onde as mudanças de brilho são relevantes, são cha-

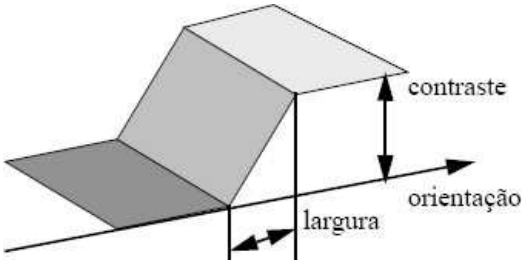


Figura 2.1: Gráfico da intensidade representando uma borda bidimensional. (Fonte: [54])

mados de pontos da borda. Seria interessante que estes pontos se agrupassem nas fronteiras dos objetos contidos na imagem, porém isto não ocorre em todos os casos. Uma superfície rugosa irá apresentar pontos com diferentes intensidades de brilho que não fazem parte da borda do objeto [13].

Tipicamente, separar bordas verdadeiras de bordas espúrias (bordas falsas), em uma imagem, não é uma tarefa fácil e requer uma grande quantidade de informações. Apesar de tudo, estudos na área de visão computacional afirmam que existem várias propriedades interessantes em uma imagem que são de grande importância na classificação e distinção entre bordas verdadeiras e falsas [13].

A maioria das técnicas de detecção de bordas empregam operadores diferenciais de primeira ou segunda ordem. Os operadores diferenciais ressaltam os contornos das bordas mas também amplificam o ruído. Por esse motivo, grande parte dos operadores de borda utilizam algum tipo de suavização de imagem antes da operação diferencial [54].

### 2.1.1 Ruído

Na conversão da imagem analógica para o meio digital (digitalização), surge o chamado ruído. O ruído é causado na fase de captura da imagem, através dos sensores de dispositivos de captura digital, tais como câmeras digitais, webcams e filmadoras, iluminação mal adequada, entre outros. O ruído geralmente aparece como variações discretas em pixels isolados e é de natureza aleatória. Imagens sem ruído na prática não existem e tão pouco podemos mensurá-lo ou prevê-lo. Simplesmente toma-se o cuidado de adquirir imagens com o mínimo de ruído possível. Para tanto, faz-se necessário equipamentos de ótima qualidade com sensores mais modernos, uma boa iluminação da cena e principalmente um profissional que saiba ajustar o dispositivo digital para cada tipo de cenário [38].

A Figura 2.2 ilustra com clareza o efeito provocado pelo ruído em uma

imagem sintética. Observando o gráfico da variação de intensidade dos pixel contidos na linha horizontal que corta a figura, podemos notar como o ruído prejudica a identificação das bordas verdadeiras.

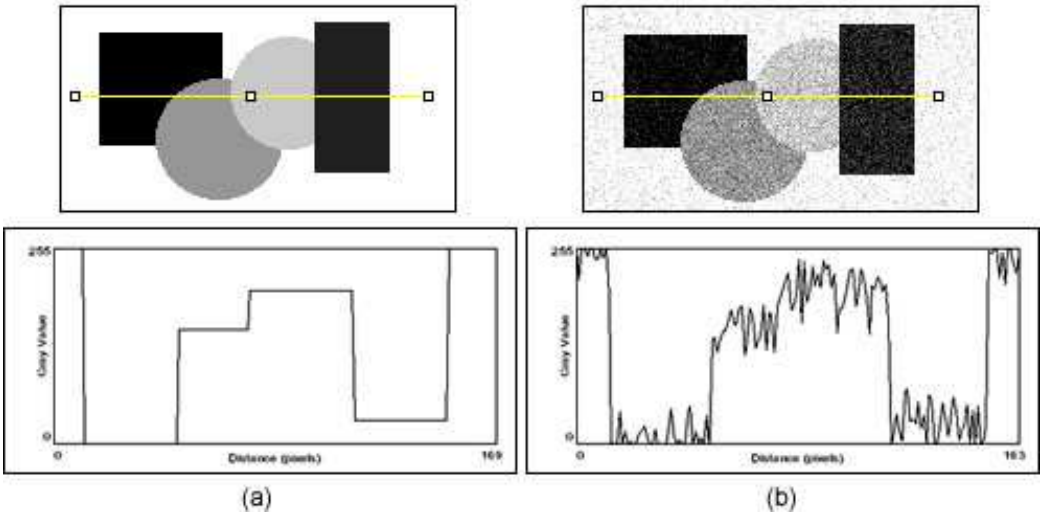


Figura 2.2: Variações de intensidade que ocorrem na imagem sem ruído (a) e com ruído (b).

Para tentar diminuir o efeito do ruído na imagem, aplica-se alguma espécie de filtro à mesma, essa técnica recebe o nome de alisamento ou suavização *(smoothing)* da imagem. Podemos afirmar que o alisamento da imagem suprime alguns tipos de ruído, no entanto, para sermos mais precisos, precisamos de um modelo de ruído. Habitualmente, o termo ruído quer dizer que existe algum tipo de informação indesejada anexada a imagem, com o qual precisamos nos preocupar, o restante é a informação útil [13].

### 2.1.2 Filtros Lineares e Convolução

Vários efeitos importantes ao processamento de imagens podem ser obtidos através de modelos matemáticos extremamente simples. Podemos construir uma matriz com as mesmas dimensões da imagem e então preencher cada posição desta matriz com o somatório sobrecarregado (influenciado por pesos) dos valores de seus respectivos vizinhos na imagem original. Estes pesos são armazenados em uma matriz com dimensões que podem variar de acordo com o efeito desejado, e recebe o nome de núcleo ou *kernel*. Um exemplo de núcleo é ilustrado na Figura 2.3. Existe uma grande variedade de possíveis núcleos, e cada um deles representa um processo diferente. Este tipo de processo recebe o nome de Filtro Linear [41].

$$\begin{bmatrix} \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \\ \frac{2}{16} & \frac{4}{16} & \frac{2}{16} \\ \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \end{bmatrix}$$

Figura 2.3: Exemplo de núcleo.

### Convolução

Aplicar um filtro linear a uma imagem, habitualmente recebe o nome de convolução. A notação matemática do processo da convolução é bastante simples e está representada pela Equação 2.1. Dado o núcleo $H$, sua convolução com a imagem $F$ é uma imagem $R$ de mesma dimensão [13].

$$R_{ij} = \sum_{u,v} H_{(i-u,j-v)} F_{(u,v)} \tag{2.1}$$

Este tipo de operação é comumente utilizado para suavização da imagem e operações derivadas. Existem alguns modelos matemáticos que podem ser utilizados para a suavização de uma imagem, dentre os quais podemos citar a suavização através da média e a suavização Gaussiana, os quais também são definidos por operadores de convolução [13].

### Suavização Através da Média

Os pixels de uma imagem geralmente possuem valores similares ao dos seus vizinhos. Assumimos uma imagem que tenha sido afetada por algum tipo de ruído de natureza aleatória. Uma boa maneira de remover este ruído seria a substituição do valor dos pixels afetados pelo valor da média de seus vizinhos. Este processo muitas vezes é chamado de suavização da imagem [13].

Podemos pensar em um modelo de suavização, que simplesmente substitua cada pixel da imagem pela média uniforme, ou seja, sem uso de pesos dos pixels de uma determinada região. Como exemplo, podemos calcular a média dos pixels contidos na região $2k + 1 \times 2k + 1$ ao redor do pixel de interesse, onde $k$ representa o raio do núcleo. Para uma imagem $F$, teremos como resultado uma imagem $R$. A representação matemática deste processo é descrita pela Equação 2.2 [13].

$$R_{ij} = \frac{1}{(2k+1)^2} \sum_{u=i-k}^{u=i+k} \sum_{v=j-k}^{v=j+k} F_{(u,v)} \tag{2.2}$$

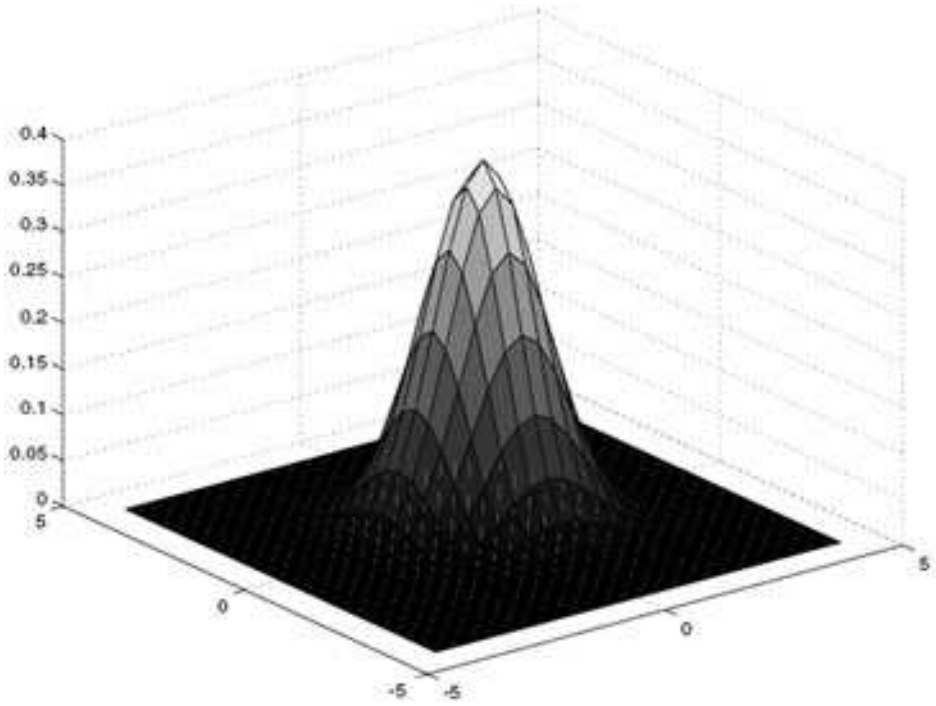


Figura 2.4: Visualização 3D de um núcleo Gaussiano. (Fonte: [13])

### Suavização Gaussiana

Um bom modelo de suavização é o modelo baseado no núcleo Gaussiano, representado pela Figura 2.4. Sua representação matemática é mostrada na Equação 2.3 [13].

$$G_{(x,y)} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right) \tag{2.3}$$

Na Equação 2.3, $\sigma$ *(sigma)* representa o desvio padrão. Este modelo calcula a média dos pixels vizinhos ao pixel de interesse, baseado em pesos. Os pesos mais ao centro do núcleo tem maior influência no cálculo da média, esta

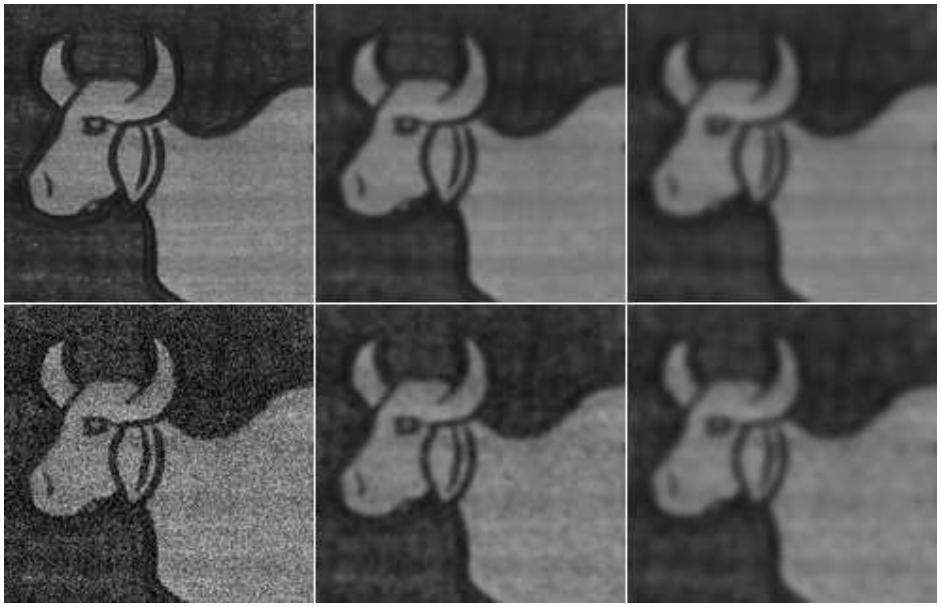

Figura 2.5: Aplicação da suavização Gaussiana na eliminação de ruído.

influência diminui gradativamente a medida que se afasta do centro. Analisando o funcionamento deste modelo podemos tirar as seguintes conclusões [13]:

- Se o valor de $\sigma$ for muito pequeno (menor que um pixel), a suavização terá um efeito muito pequeno, visto que os pesos do centro do núcleo serão muito pequenos;
- Para valores muito grandes de $\sigma$, a suavização causará perda de muitos detalhes da imagem, que serão esmaecidos juntamente com o ruído;
- A escolha do $\sigma$ ideal depende inteiramente do problema e do tipo de ruído que se deseja suavizar.

A Figura 2.5 ilustra os itens citados acima, onde é possível notar a notável redução do ruído de acordo com a variaçao do desvio padrão da função. Nas aplicações, pode-se construir um núcleo Gaussiano $H$, de dimensões $2k+1 \times 2k+1$ através da Equação 2.4 [13].

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{((i-k-1)^2 + (j-k-1)^2)}{2\sigma^2}\right) \tag{2.4}$$

A Figura 2.6 ilustra a comparação entre a suavização através da média local uniforme e a suavização Gaussiana. A suavização através da média

é considerada uma boa técnica de suavização, no entanto, ela gera alguns efeitos na imagem além do esmaecimento. A imagem mais a esquerda representa a imagem de um jardim. A imagem do meio representa a suavização através da técnica da média local uniforme. A imagem mais a direita representa a suavização através da média utilizando pesos Gaussianos. O grau de esmaecimento é o mesmo para as duas figuras, no entanto, a técnica da média uniforme gera um conjunto de barras horizontais e verticais na imagem, tornando-a levemente quadriculada [13].

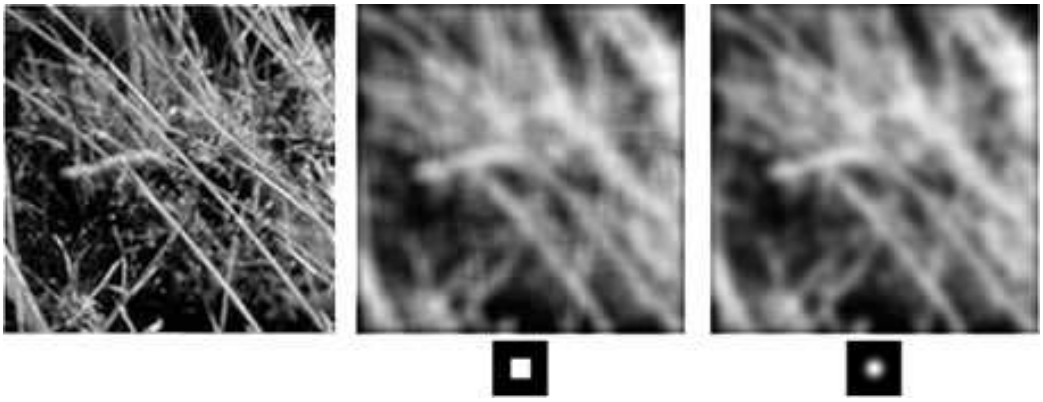

Figura 2.6: Comparação entre a suavização através da média e a suavização Gaussiana. (Fonte: [13])

## 2.2 Detecção de Bordas em Imagens em Tons de Cinza

Vários algoritmos para detecção de bordas em tons de cinza foram desenvolvidos nas décadas de 60 e 70 [21]. Como exemplo, podemos citar os algoritmos de Roberts, Prewitt, Sobel e Canny. São algoritmos bastante simples mas poderosos, que continuam sendo utilizados por muitas aplicações, embora outros algoritmos mais sofisticados tenham sido desenvolvidos [23]. Estes algoritmos se baseiam principalmente na técnica da convolução e muitos deles foram adaptados para serem utilizados em imagens coloridas [56].

O algoritmo de Roberts é uma simples aproximação de operadores que utilizam a diferenciação de imagem. A implementação pode ser feita por meio de duas máscaras de dimensões 2 x 2, representadas em 2.5 [56].

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{2.5}$$

O algoritmo de Pretwitt é uma extensão do algoritmo de Roberts, no entanto as máscaras possuem dimensão 3 x 3, representadas em 2.6. São computadas as diferenças na direção vertical e na direção horizontal.

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.6}$$

O algoritmo de Sobel é muito similar ao algoritmo de Pretwitt [56]. São enfatizadas as linhas horizontais e verticais em torno do pixel central. As máscaras estão representadas em 2.7.

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.7}$$

A Figura 2.7 ilustra a detecção das bordas através do algoritmo de Sobel, aplicado, separadamente, ao eixo $X$ e ao eixo $Y$ da imagem. Na imagem mais a esquerda observa-se o detalhe da imagem de uma tábua entalhada. A imagem do centro mostra o resultado da convolução com a máscara de Sobel aplicada na direção do eixo $Y$, o que faz com que as bordas horizontais sejam fortemente destacadas e as bordas verticais sejam fracamente destacadas. A imagem mais a direita mostra o resultado da convolução com a máscara de Sobel aplicada na direção do eixo $X$, fazendo com que as bordas verticais sejam fortemente destacadas e as bordas verticais sejam fracamente destacadas.

O algoritmo de Canny [5] utiliza a técnica do filtro linear com um núcleo Gaussiano para diminuir o ruído na imagem. O próximo passo durante a execução do algoritmo é o cálculo da magnitude do gradiente das bordas e o cálculo da direção das mesmas. Estas operações são feitas através da diferenciação da imagem nas direções horizontal e vertical, então a magnitude do gradiente das bordas é calculado através da raiz quadrada do somatório dos quadrados das derivadas calculadas. A arco tangente da razão entre as derivadas, é utilizada para computar a direção do gradiente. O próximo passo do algoritmo é a supressão não-máxima, nesta etapa cada um dos pixels da imagem resultante são comparados com os seus vizinhos na direção do gradiente, o pixel será setado para zero caso não seja maior que seus vizinhos. Os pixels remanescentes da supressão não máxima recebem o nome de "candidatos de borda"e ainda são submetidos a um método de limiarização para obter o mapa final de bordas [51].

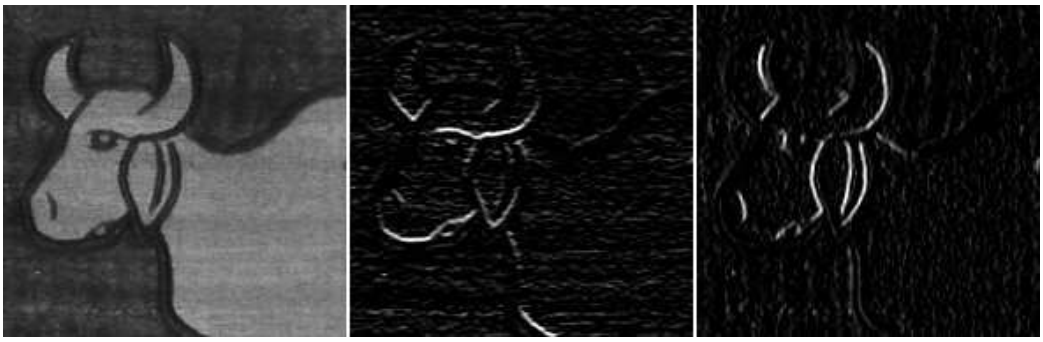

Figura 2.7: Detecção das bordas na direção do eixo $X$ e na direção do eixo $Y$.

## 2.3 Detecção de Bordas em Imagens Coloridas

O avanço da tecnologia e o aumento da capacidade de processamento dos computadores fez com que novos paradigmas no campo da detecção de bordas pudessem ser alcançados. Utilizar as informações de cor das imagens durante o processamento deixou de ser um problema e passou ser visto como um benefício, pois agora a quantidade de informações úteis para a detecção de bordas é maior, possibilitando a obtenção de resultados mais precisos no processamento das imagens [56].

A percepção das cores no sistema visual humano é de fundamental importância, humanos se baseiam nas informações de cor, saturação e intensidade para interpretar o mundo real. Conseqüentemente, estas informações podem ser utilizadas para aumentar a exatidão dos algoritmos de detecção de bordas em tons de cinza existentes. Muitos pesquisadores já utilizaram imagens coloridas em aplicações complexas, tais como: localização de textos [43], inspeção automatizada de granito [48] e segmentação de mapas coloridos [30]. No processamento de imagens coloridas, várias tarefas são executadas da mesma forma que eram executadas anteriormente nas imagens em tons de cinza. A principal diferença é a disponibilidade dos valores cromáticos da imagem [56].

Todo sistema de aquisição de imagens coloridas se baseia em algum modelo de representação de cores, geralmente é um sistema aditivo tri-cromático, que envolve as cores primárias. Existem vários espaços de cor, tais como o RGB, CIE XYZ, HSI, entre outros, que serão discutidos mais adiante. O sistema RGB é o modelo que mais se assemelha aos sensores vermelho, verde e azul da maioria dos sensores CCD, enquanto a percepção das cores no sis-

tema visual humano é representada mais fielmente pelo sistema de cores HSI [56].

### 2.3.1 Cores

A cor dos objetos é uma característica que está intimamente relacionada com a luz. Existem duas linhas de raciocínio que podem ser tomadas: a Cor-Luz, onde a cor é a própria luz e a Cor-Pigmento, onde a cor é a luz que é refletida pelo objeto, fazendo com que o olho humano perceba esse estímulo como cor. Os dois extremos da classificação das cores são: o branco, ausência total de cor, ou seja, luz pura, e o preto, ausência total de luz, o que faz com que nenhuma cor seja refletida [13].

As cores podem se formar por meio de um processo aditivo, subtrativo ou formação por pigmentação. No processo aditivo, ocorre uma combinação de dois ou mais raios luminosos de freqüências diferentes, nesta combinação a formação da cor ocorre pela soma da energia dos fótons. No processo de formação por subtração, a luz é transmitida através de um filtro que absorve a radiação luminosa de um determinado comprimento de onda. A luz também pode ser transmitida através de um corante constituído por partículas que agem como filtros absorvendo radiação luminosa de um determinado comprimento de onda. Na formação por pigmentação, os pigmentos podem absorver, refletir ou transmitir a radiação luminosa [13].

A luminância e crominância são dois conceitos importantes para a compreensão do conceito de percepção de cor. A luminância é uma grandeza que indica a razão entre a intensidade luminosa emitida por uma superfície, numa dada direção, e a área da superfície emissora projetada sobre um plano perpendicular a esta direção. O cérebro humano compreende esta informação como a quantidade de cinza presente na cor, ou seja, o brilho da imagem. A Figura 2.8 ilustra a influência da luminância em uma imagem. A crominância é uma característica da imagem que é definida por dois valores: coloração e saturação, ilustrado na Figura 2.9. A coloração é a parte da luz refletida por um objeto. Este absorve luz e reflete apenas uma parte do espectro visível. A saturação define a proporção de branco que uma cor contém. Estas duas informações combinadas, em diferentes proporções, permitem ao cérebro humano perceber todo o espectro de cores visível numa cena [13].

As cores são divididas em três grupos: cores primárias, cores secundárias e cores terciárias. Cores primárias são as cores puras, ou seja, que não se fragmentam. As cores primárias da Cor-Luz são o vermelho, o verde e o azul (RGB), já para a Cor-Pigmento as cores primárias são o vermelho, o amarelo e o azul. As combinações de cores que surgem através da combinação de duas cores primárias recebem o nome de cores secundárias e são representadas pelo

Figura 2.8: Influência da luminância na imagem.

Figura 2.9: Variação da saturação para a cor vermelha.

laranja, combinação entre o amarelo e o vermelho, o verde, combinação do azul com o amarelo, e o violeta, combinação do vermelho com o azul. As cores terciárias são obtidas através da mistura de uma cor primária com uma ou mais cores secundárias. A Figura 2.10 representa a distribuição das cores primárias, secundárias e terciárias [32].

### Representação das Cores

Existem várias maneiras diferentes de se representar uma cor. A maneira mais comum, utilizada na representação de imagens em um computador, é o uso de um conjunto de três valores, representando as intensidades das cores primárias. Cada combinação diferente entre esses valores representa uma cor distinta. O espaço tridimensional que descreve a distribuição espacial das cores é chamado de espaço de cor [56].

São muitos os espaços de cor existentes, entre eles podemos citar o RGB, CMY, YIQ, YUV, HSI, HSV, CIELAB, CIELUV, rgb, $c_1c_2c_3$, $l_1l_2l_3$, $YC_bC_r$, entre outros [56].

### Espaços de Cor

**RGB:** O espaço de cor RGB *(Red, Green, Blue)* é freqüentemente utilizado para representar as cores em telas de computador e em aplicações de processamento de imagens [56]. Neste modelo as cores são formadas pela combinação de três intensidades dos componetes cromáticos básicos (vermelho, verde e azul). A partir da aplicação de diferentes intensidades dessas três cores primárias pode-se obter as demais cores. Por exemplo, a cor amarela é obtida pela combinação de vermelho e verde. No modelo RGB a intensidade de cada cor primária varia no espaço de um byte (0..255), ou seja, são possíveis 256 valores diferentes para cada uma das componentes básicas, logo, temos 256 x 256 x 256, obtem-se o valor de 16.777.216 variações possíveis

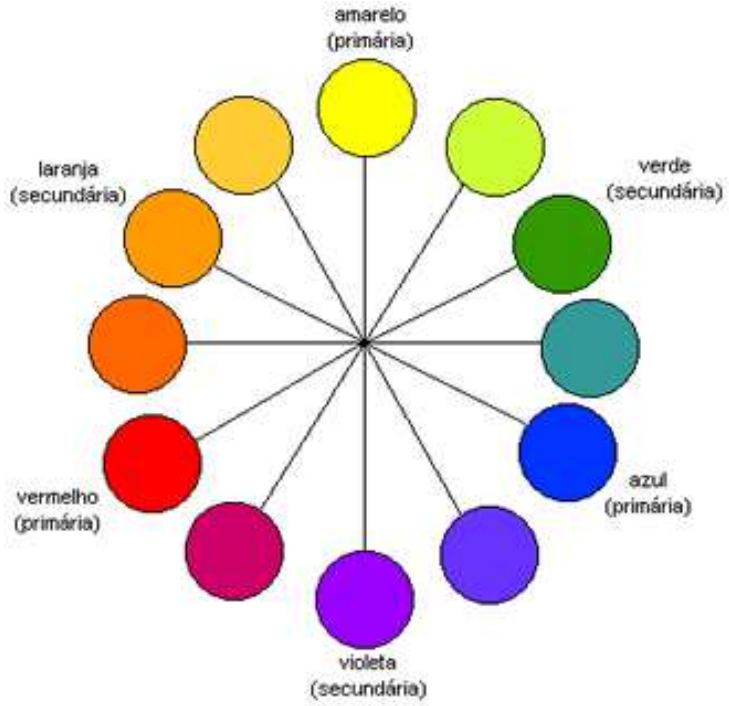


Figura 2.10: Representação das cores primárias, secundárias e terciárias. (Fonte: [32])

de cores. Usualmente, a representação gráfica do modelo RGB é um cubo, onde as arestas adjacentes à origem representam as componentes cromáticas básicas de acordo com a Figura 2.11.

**CIE XYZ:** O espaço de cor CIE XYZ foi desenvolvido como uma alternativa ao modelo RGB [32], é um modelo de representação de cores bastante conhecido. Para descrever uma cor, utilizando este modelo, basta especificar as coordenadas de X, Y e Z [13]. Partindo do conceito de que é impossível escolher três cores primárias, as quais misturadas, possam originar todas as demais cores, três cores primárias imaginárias foram definidas quando este modelo foi criado em 1931 [32].

O espaço de cor CIE XYZ permite uma grande variedade de representações gráficas em duas ou três dimensões, no entanto, é muito mais difícil desenhar em três dimensões do que em duas, por este motivo é comum utilizar a representação gráfica do modelo XYZ através do plano $X+Y+Z=1$, como mostrado na Figura 2.12, o sistema de coordenadas nesta representação se dá através da Equação 2.8. A visualização deste modelo em duas dimensões recebe o nome de CIE xy e pode ser observado na Figura 2.13 [13].

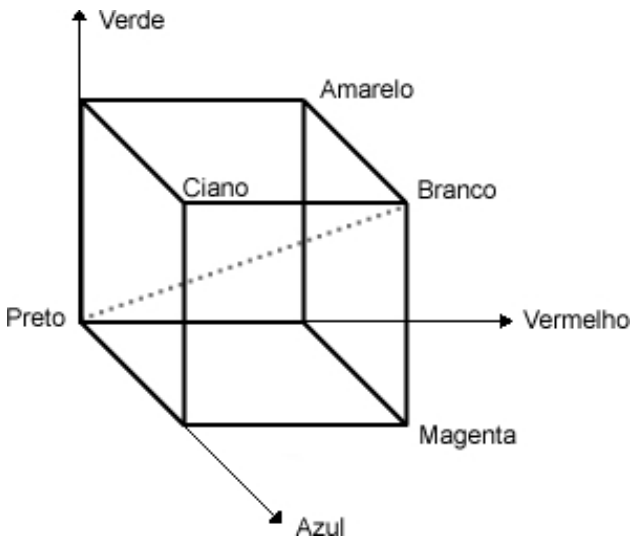


Figura 2.11: Representação gráfica do modelo RGB.

$$(x, y) = \left( \frac{X}{X + Y + Z}, \frac{Y}{X + Y + Z} \right) \tag{2.8}$$

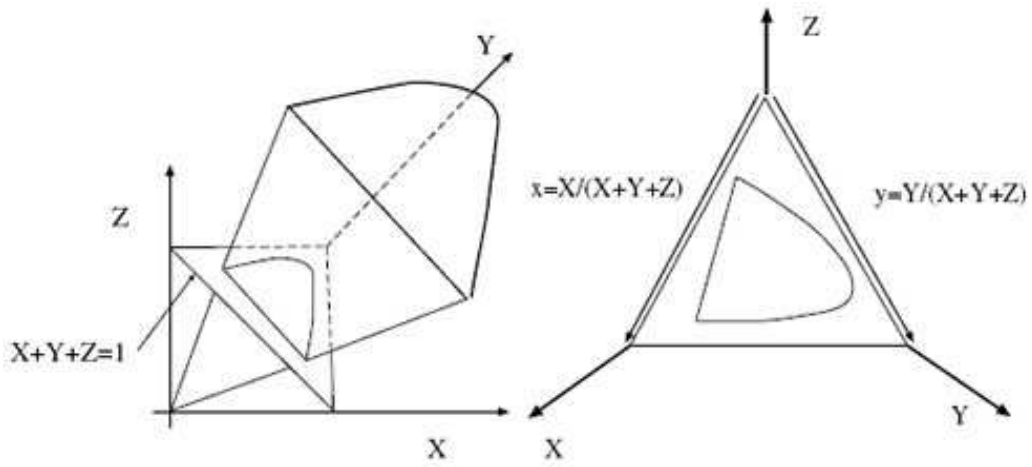


Figura 2.12: Representação gráfica do modelo CIE XYZ. (Fonte: [13])

O espaço de cor CIE XYZ pode ser obtido através da transformação do espaço de cor RGB através da transformação ilustrada pela Equação 2.9 [56].

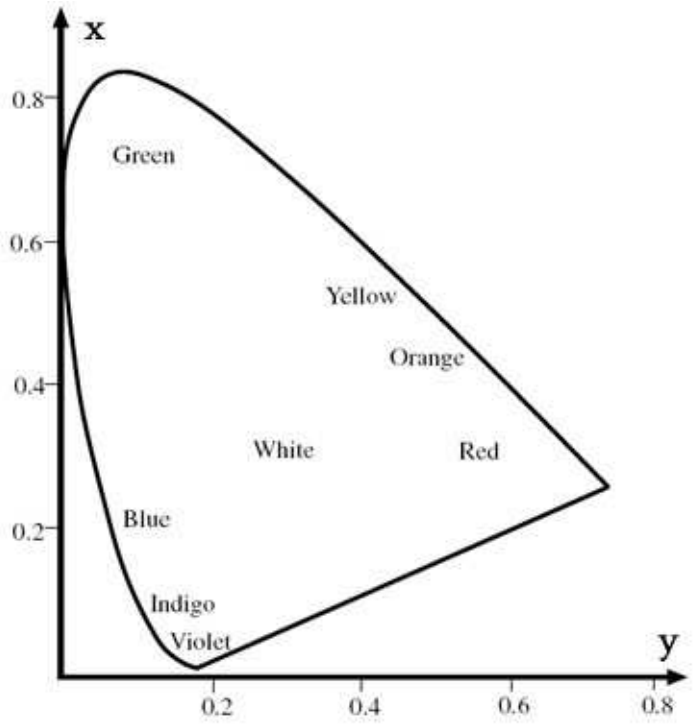


Figura 2.13: Representação gráfica do modelo CIE XYZ em duas dimensões (CIE xy). (Fonte: [13])

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.607 & 0.174 & 0.200 \\ 0.299 & 0.587 & 0.114 \\ 0.000 & 0.066 & 1.116 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.9}$$

**HSI:** O espaço de cor HSI *(Hue, Saturation, and Intensity)* é o que representa com maior fidelidade o mecanismo de percepção das cores do sistema visual humano. $H$ representa a cor fundamental. $S$ representa a saturação (considerando a cor vermelha, o rosa possui um baixo valor de saturação, a cor vermelha é completamente saturada). $I$ representa o brilho total ou a quantidade de luz, é um valor que independe da cor [56].

A definição do modelo de cor HSI é dada através das equações abaixo [20]:

$$cos\ \theta = \frac{(R-G)+(R-B)}{2*[(R-G)^2+(R-B)*(G-B)]^{1/2}} \tag{2.10}$$

$$H = \{\theta \; se \; (B < G) \; OU \; (2\pi - \theta) \; em \; caso \; contrario\}$$

$$S = 1 - \frac{3 * min(R, G, B)}{R + G + B}$$

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

### 2.3.2 Redução de Cores

A redução de cores consiste em transformar uma imagem que utiliza $M$ *bits* de cor para $N$ *bits*, sendo que $M > N$. Ou seja, se temos uma imagem que possui 256 tons de cinza podemos transformar o espaço de cor desta imagem para 64 tons de cinza. Formalmente, a redução de cores é um processo de "discretização", denominado de quantização [19].

Utiliza-se a quantização de cores principalmente para reduzir a quantidade de memória necessária para armazenar/representar uma imagem, ou então para igualar os valores de pixels semelhantes. Quando aplicamos uma quantização, estamos dividindo um espaço de cor numa série de conjuntos de cores. Cada um destes conjuntos é denominado de célula de quantização e a cada célula está associado um valor constante denominado nível de quantização. Por exemplo, para quantizar uma imagem de 256 tons de cinza *(8 bits)* para 16 tons de cinza *(4 bits)* iremos necessitar de quatro células de quantização. Todos os pixels com valor entre 0 e 64 na imagem original receberiam o valor 32 na imagem quantizada, todos os pixels com valor entre 65 e 128 na imagem original receberiam o valor 96 na imagem quantizada e assim por diante. Este tipo de quantização recebe o nome de quantização uniforme, e o nível de quantização é dado pela Equação 2.11 [19].

$$q_i = \frac{c_i + c_{i-1}}{2}, \; 1 \leq i \leq L \tag{2.11}$$

Onde $q_i$ é o nível de quantização da célula de quantização $i$, $c_i$ é o valor da céula de quantização e $L$ é a quantidade de cores que se deseja reduzir.

A quantização uniforme pode ser obtida de maneira bem simples, no entanto, ela pode gerar resultados não fiéis a imagem original. Para melhorar o resultado visual da quantização existem métodos de quantização adaptativos, dentre os quais podemos citar a quantização por seleção direta, quantização por subdivisão recursiva e algoritmo do corte mediano [19].

O algoritmo do corte mediano foi desenvolvido por Heckbert [24] e sua idéia básica é a subdivisão repetitiva do cubo de cor (no sistema RGB) em

retângulos menores. Seja $K$ o nível de quantização desejado, o algoritmo do corte mediano para reduzir uma imagem para $K$ cores é o seguinte:

1. Encontrar o menor e maior valor de vermelho, verde e azul na imagem;
2. Estes limites determinam um paralelepípedo de volume mínimo que contém todas as cores presentes na imagem;
3. Encontrar a componente de cor que possui o maior intervalo entre seus limites superior e inferior (que vem a ser a maior aresta deste paralelepípedo);
4. Ordenar as triplas RGB que compõem a imagem pela componente de cor que foi encontrada na etapa 3;
5. Calcular a mediana das triplas de cores ordenadas na etapa 4;
6. Obtém-se assim duas sub-regiões do paralelepípedo, que são cada uma delas uma célula de quantização. Aplicar recursivamente para cada sub-região o algoritmo a partir da etapa 3 até que: (A) as sub-regiões não contenham mais cores presentes na imagem ou (B) a quantidade de $K$ células foi alcançada;
7. Calcular o nível de quantização de cada célula a partir da média das cores que compõem cada célula de quantização.

O algoritmo do corte mediano é um algoritmo bastante utilizado para a redução de cores devido a sua facilidade de implementação, baixo custo computacional e bons resultados de nível visual [19].

### 2.3.3 Medidas de Similaridade

As medidas de similaridade são utilizadas na extração de características de uma imagem afim de se obter algum tipo de informação que possa ser utilizada para realizar algum processamento sobre a imagem [45]. As características mais utilizadas são: cor, textura e forma, bem como qualquer tipo de combinação entre essas características [2] [40]. Estas características podem ser classificadas em: características gerais e características de domínio específico. Nas características gerais enquadram-se cor, textura e forma, e nas características de domínio específico podemos citar características referentes a impressões digitais e faces humanas [45].

Existem diferentes técnicas para medir a similaridade entre as cores. A escolha da técnica a ser utilizada depende do espaço de cor que estiver em

uso, por exemplo, a distância Euclidiana é utilizada para os espaços de cor RGB [56], CIELUV [44] e CIELAB [58], variações da distância Euclidiana vem sendo aplicadas ao espaço de cor HCI [8]. As medidas de similaridade podem estar no espaço Euclidiano (espaço métrico) ou serem não métricas. Muitos métodos assumem que o vetor de características a ser comparado está no espaço euclidiano [2], embora a percepção visual humana de similaridade nem sempre se encontra neste espaço [27]. As principais métricas utilizadas são a distância Euclidiana e o ângulo entre vetores [56]. É desejável que uma função de medida de similaridade $d(i, j)$ siga as seguintes propriedades, que garantem a boa discriminação das diferenças entre as cores [1]:

1. $d(i, j) \geq 0$;
2. $d(i, j) = 0$ se e somente se $i = j$ (positividade);
3. $d(i, j) = d(i, j)$ simetria;
4. $d(i, j) \leq d(i, j) + d(i, k)$ (desigualdade triangular).

### Distância Euclidiana

A distância Euclidiana ($D_E$) é geralmente utilizada para calcular a distância em um espaço de $n$ dimensões e é definida pela Equação 2.12 [56].

$$D_E(\vec{v_1}, \vec{v_2}) = \left\| \vec{v_1} - \vec{v_2} \right\| \tag{2.12}$$

Para um plano tridimensional a distância Euclidiana é definida por (Equação 2.13):

$$D_E(\vec{v_1}, \vec{v_2}) = \sqrt{(v_{1,1} - v_{2,1})^2 + (v_{1,2} - v_{2,2})^2 + (v_{1,3} - v_{2,3})^2} \tag{2.13}$$

Onde $\vec{v_1} = [v_{1,1}\ v_{1,2}\ v_{1,3}]^T$ representa as três componentes básicas de um espaço de cores, por exemplo o RGB.

**Comportamento no espaço de cor RGB:** No espaço de cor RGB, a distância Euclidiana representa, ao mesmo tempo, as diferenças de intensidade, cor e saturação entre duas cores. Não é nítida a proporção em que cada uma dessas características é representada. Experimentos provam que a intensidade é a característica que tem maior influência durante o cálculo desta diferença [56].

Dados dois pixels com as mesmas características de saturação e cor, mas com valores de intensidades diferentes, a distância Euclidiana será um valor diferente de zero, devido a variação na intensidade.

$$D_E(\vec{v_1}, \vec{v_2}) \neq 0 \tag{2.14}$$

Conclui-se que a distância entre duas cores depende em grande parte do valor da intensidade, o que é indesejável para o cálculo da distância entre duas cores diferentes mas com um valor similar de intensidade. Por outro lado, esta é uma característica desejável para o cálculo entre duas cores que possuem a mesma informação de cor [56].

Observando a imagem sintética 2.14, podemos notar que as amostras de cor $A$ e $B$ possuem as mesmas características de coloração, no entanto, a região $A$ possui uma incidência de iluminação 20% maior que a região $B$. Para a segmentação da imagem, seria interessante que houvesse diferença no calculo da distancia entre as cores $A$ e $B$ ou entre as cores $C$ e $D$, o que não é obtido com o uso da distância Euclidiana [10].

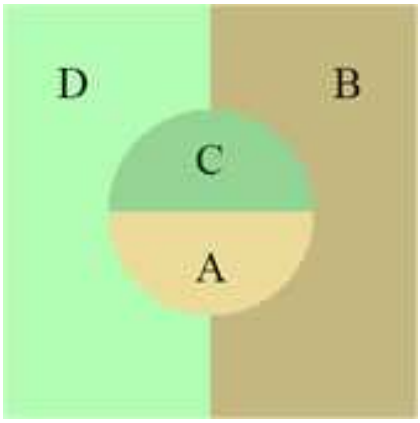


Figura 2.14: Amostras de cores. (Fonte: [56])

**Comportamento em outros espaços de cor:** A distância Euclidiana caracteriza a diferença entre as cores de maneira diferente para cada um dos espaços de cor. A Tabela 3.1 exibe o resultado do cálculo da distância Euclidiana para diversos espaços de cor, utilizando como referência, a Figura 2.14 [56].

| **Espaço de cor** | **Distância Euclidiana** | | | **Discriminação** |
|---|---|---|---|---|
| | $\mathbf{D}_E(A,B)$ | $\mathbf{D}_E(C,D)$ | $\mathbf{D}_E(B,C)$ | |
| RGB | 59.72 | 59.69 | 60.34 | Fraca |
| XYZ | 60 | 60 | 33 | Fraca |
| CIELAB | 6.8 | 6.8 | 23.4 | Boa |
| CIELUV | 7 | 7 | 37 | Boa |
| rgb | 0.0014 | 0.0024 | 0.12 | Boa |
| $l_1l_2l_3$ | 0.0054 | 0 | 0.77 | Boa |
| $h_1h_2h_3$ | 17.1 | 0 | 117.3 | Boa |

Tabela 2.1: Comparação do cálculo da distância Euclidiana para diversos espaços de cor [56].

### Ângulo entre Vetores

O ângulo entre vetores ($D_{VA}$) é uma técnica de medida de similaridade definida pela Equação 2.15:

$$\cos\theta = \frac{\vec{V_1}^T \vec{V_2}}{\left\|\vec{V_1}\right\| . \left\|\vec{V_2}\right\|} \tag{2.15}$$

Ao contrário da distância Euclidiana, o ângulo entre vetores é mais sensível às diferenças nos valores de cor e saturação do que no valor de intensidade. Existe um empecilho quando se utiliza o ângulo $\theta$ como medida de distância entre duas cores [55], pois a magnitude do valor do $\cos\theta$ ou $1-\cos\theta$ para ângulos pequenos é muito pequena se comparada a magnitude do valor do $\sin\theta$ [10]. Por este motivo, a utilização do $\sin\theta$ foi proposta [10] como a atual medida da distância entre ângulos, definida pela Equação 2.16.

$$D_{VA} = \sin\theta = \left[1 - \left(\frac{\vec{V_1}^T \vec{V_2}}{\left\|\vec{V_1}\right\| . \left\|\vec{V_2}\right\|}\right)^2\right]^{1/2} \tag{2.16}$$

**Comportamento no espaço de cor RGB:** Considerando o valor de dois pixels $\vec{V_1}$ e $\vec{V_2}$ oriundos de áreas distintas da imagem com as mesmas características de cor e saturação, mas com valores de intensidade diferentes, o cálculo da distância entre vetores, no espaço de cor RGB, para estes pixels resultará em zero [56].

Isto mostra uma característica interessante desta técnica, a medida da distância entre duas cores no espaço de cor RGB é insensível a variações de intensidade, mas sensível às diferenças de cor e saturação [56].

**Comportamento em outros espaços de cor:** O ângulo entre vetores caracteriza a diferença entre as cores de maneira diferente para cada um dos espaços de cor. A Tabela 3.2 exibe o resultado do cálculo do ângulo entre vetores para diversos espaços de cor, utilizando como referência, a Figura 2.14 [56].

| **Espaço de cor** | **Ângulo entre Vetores** | | | | | | **Discriminação** |
|---|---|---|---|---|---|---|---|
| | $\mathbf{D}_{VA}(A,B)$ | | $\mathbf{D}_{VA}(C,D)$ | | $\mathbf{D}_{VA}(B,C)$ | | |
| | $\theta$ | $\sin\theta$ | $\theta$ | $\sin\theta$ | $\theta$ | $\sin\theta$ | |
| RGB | $0.11^o$ | 0.002 | $0.11^o$ | 0.002 | $11.6^o$ | 0.20 | Boa |
| XYZ | $0.20^o$ | 0.035 | $0.05^o$ | 0.000 | $6.3^o$ | 0.11 | Boa |
| CIELAB | $0.64^o$ | 0.011 | $0.12^o$ | 0.002 | $14.8^o$ | 0.26 | Boa |
| CIELUV | $0.95^o$ | 0.017 | $0.15^o$ | 0.003 | $22.9^o$ | 0.39 | Boa |
| rgb | $0.11^o$ | 0.002 | $0.11^o$ | 0.002 | $11.6^o$ | 0.20 | Boa |
| $l_1l_2l_3$ | $0.44^o$ | 0.008 | $0^o$ | 0.000 | $66^o$ | 0.91 | Boa/Ruído |
| $h_1h_2h_3$ | $0.46^o$ | 0.008 | $0^o$ | 0.000 | $71.1^o$ | 0.95 | Boa/Ruído |

Tabela 2.2: Comparação do cálculo do ângulo entre vetores para diversos espaços de cor [56].

A Figura 2.14 foi submetida a detecção de bordas através de um detector baseado na distância Euclidiana (Figura 2.15) e a detecção de bordas através de um detector baseado no vetor entre ângulos (Figura 2.16). Ambos os métodos diferenciam cada uma das regiões, entretanto, quando é aplicado o vetor de ângulos, o resultado é uma imagem onde as bordas entre as regiões A e B e entre as regiões C e D não existem. Isto se deve ao fato de que o ângulo entre os vetores destes pares de cores são muito próximos de zero [56].

### 2.3.4 Combinações de Medidas de Similaridade

Uma área emergente no ramo do processamento de imagens é a combinação das técnicas de similaridade baseadas nas informações de cor e intensidade. Existem duas maneiras principais de se combinar estas técnicas: combinação baseada na intensidade [56] ou combinação baseada na saturação [1]. Ambas possuem suas vantagens e suas desvantagens.

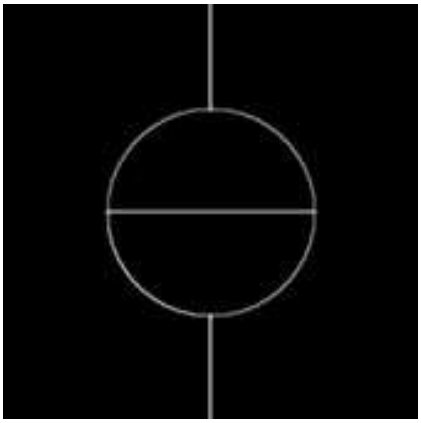

Figura 2.15: Detecção de bordas através da distância Euclidiana. (Fonte: [56])

### Combinação Baseada na Intensidade

Uma maneira simples de se combinar as técnicas de medida de similaridade envolve o valor da intensidade dos pixels. Uma maneira de calcular a intensidade é através da média dos compontentes RGB do pixel. O uso da intensidade como parâmetro de comparação é uma escolha que pode ser justificada devido ao fato de que a distância entre ângulos não é satisfatória para pixels com valores baixos de intensidade. Então, a distância entre vetores pode ser utilizada quando ambos os pixels possuirem valores de intensidade elevados, e a distância Euclidiana, quando um dos pixels possuir o valor de intensidade baixo [56].

### Combinação Baseada na Saturação

A combinação de medidas de similaridade baseada na informação de saturação para detecção de bordas foi proposta por Carron e Lambert [7], o espaço de cor RGB foi convertido para o espaço de cor HSI utilizando a transformação do espaço de cor $YC_1C_2$ representada pela Equação 2.17.

$$\begin{bmatrix} X \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1 & -1/2 & -1/2 \\ 0 & -\sqrt{3}/2 & \sqrt{3}/2 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.17}$$

Resultando na definição da saturação representada pela Equação 2.18.

$$S = \sqrt{C_1^2 + C_2^2} \tag{2.18}$$

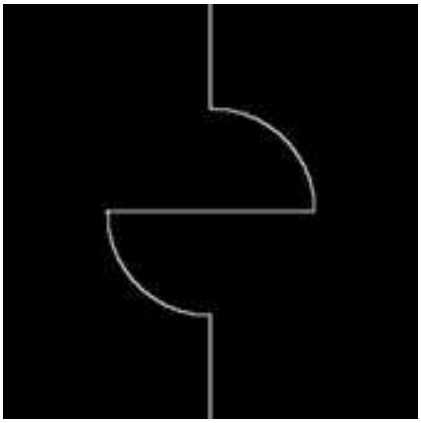

Figura 2.16: Detecção de bordas através do ângulo entre vetores. (Fonte: [56])

Eles argumentam que esta forma da saturação é menos sensível a efeitos lineares do que que a forma clássica, que é dada pela Equação 2.19.

$$S = 1 - \frac{3 * min(R, G, B)}{R + G + B} \tag{2.19}$$

O ângulo entre vetores é uma boa unidade de medida de similaridade para cores que possuem diferentes valores de cor e saturação, ao mesmo passo que a distância Euclidiana é uma boa unidade de medida para cores que possuam diferentes valores de intensidade, no espaço de cor RGB. Em resumo, quando dois pixels são demasiadamente saturados, é utilizado o ângulo entre vetores como unidade de medida, e quando um dos pixels possui o valor de saturação baixo, a distância Euclidiana é utilizada [56].

### 2.3.5 Algoritmos de Detecção de Bordas Baseados em Cor

Vários algoritmos de detecção de bordas foram adaptados para trabalhar com imagens coloridas utilizando as medidas de similaridade descritas na seção anterior: distância Euclidiana e ângulo entre vetores [56]. Dentre eles podemos citar o algoritmo modificado de Roberts, o algoritmo modificado de Sobel, o algoritmo modificado de Canny [14], o algoritmo do vetor de gradientes [48] e o algoritmo do vetor de desigualdade [59].

### Algoritmo Modificado de Roberts

O algoritmo de Roberts é um algoritmo de detecção de bordas muito simples. A versão modificada, baseada na distância Euclidiana, deste algoritmo [10] calcula o máximo valor absoluto da diferença entre os pixels adjacentes diagonalmente em uma máscara de dimensões 2 x 2, ao invés de computar a magnitude do gradiente assim como foi mostrado na Seção 3.2. Esta modificação pode ser generalizada através da Equação 2.20.

$$E_R = max\left(D_E(\vec{v}(x,y), \vec{v}(x+1,y+1)), (D_E(\vec{v}(x+1,y), \vec{v}(x,y+1))\right) \quad (2.20)$$

Onde $\vec{v}(x,y)$ é o vetor que contém os valores das componentes de cor do pixel na coordenada $(x,y)$. É possível ainda propor que a técnica do ângulo entre vetores seja utilizada como medida de similaridade. Esta modificação esta representada pela Equação 2.21.

$$S_R = max\left(D_{VA}(\vec{v}(x,y), \vec{v}(x+1,y+1)), (D_{VA}(\vec{v}(x+1,y), \vec{v}(x,y+1))\right) \quad (2.21)$$

Ainda é possível a utilização das técnicas da distância euclidiana e do ângulo entre vetores em conjunto, assim como foi citado na Seção 3.3.5.

### Algoritmo Modificado de Canny

Adaptar o algoritmo de Canny para imagens coloridas não é uma tarefa trivial. Uma alternativa seria aplicar o algoritmo de Canny a cada um dos componentes do espaço de cor da imagem. No entanto, o resultado não seria o mesmo obtido ao se utilizar os componentes do espaço de cor ao mesmo tempo. Uma técnica interessante foi proposta por Gauch [14], nesta técnica ele sugere a utilização da técnica *cubic splines* para calcular as derivadas.

### Algoritmo do Vetor de Gradientes

O algoritmo do vetor de gradientes calcula o valor máximo da distância entre o pixel central e seus oito vizinhos adjacentes, utilizando a técnica de similaridade desejada [56].

A versão deste algoritmo baseada na distância Euclidiana está representada pela Equação 2.22.

$$E_{VG} = \max_{i=1..8} \{\|\vec{v}_i(x,y) - \vec{v}_0(x,y)\|\} \quad (2.22)$$

Onde $i$ é um contador que representa cada um dos pixels que formam a vizinhança do pixel. A Matriz 2.23 ilustra o modelo de vizinhança em relação ao pixel central.

$$\begin{bmatrix} 1 & 2 & 3 \\ 8 & X & 4 \\ 7 & 6 & 5 \end{bmatrix} \tag{2.23}$$

A versão deste algoritmo baseada no ângulo entre vetores está representada pela Equação 2.24.

$$S_{VG} = \max_{i=1..8} \left[ \sqrt{1 - \left( \frac{\vec{v}_i^T(x,y).\vec{v}_0(x,y)}{\|\vec{v}_i(x,y)\| \ \|\vec{v}_0(x,y)\|} \right)^2} \right] \tag{2.24}$$

Ainda é possível unificar as duas técnicas através da Equação 2.25.

$$C_{VG} = \rho(X_1, X_2)S_{VG} + (1 - \rho(X_1, X_2))E_{VG} \tag{2.25}$$

**Algoritmo do Vetor de Desigualdade**

O algoritmo do vetor de desigualdade é muito similar ao algoritmo do vetor de gradientes. Ele é caracterizado por uma máscara de dimensão 3 x 3 que percorre a imagem, calculando o valor máximo do gradiente nas direções transversais ao pixel central [56].

A versão deste algoritmo baseada na distância Euclidiana está representada pela Equação 2.26.

$$E_{DV} = \max_{i=1..4} \{\|\vec{v}_i(x,y) - \vec{v}_{4+i}(x,y)\|\} \tag{2.26}$$

Onde $i$ representa um dos quatro vizinhos nos sentidos transversais do pixel central. A versão deste algoritmo baseada no ângulo entre vetores está representada pela Equação 2.27.

$$S_{DV} = \max_{i=1..4} \left[ \sqrt{1 - \left( \frac{\vec{v}_i^T(x,y).\vec{v}_{4+i}(x,y)}{\|\vec{v}_i(x,y)\| \ \|\vec{v}_{4+i}(x,y)\|} \right)^2} \right] \tag{2.27}$$

Ainda é possível unificar as duas técnicas através da Equação 2.28.

$$C_{DV} = \rho(X_1, X_2)S_{DV} + (1 - \rho(X_1, X_2))E_{DV} \tag{2.28}$$

## 2.4 Detecção de Bordas em Textura

A identificação e segmentação de regiões com diferentes texturas é uma fase crítica durante o processamento de imagens. Obter uma análise de textura precisa e confiável é uma tarefa muito difícil que ainda não possui uma solução completamente consolidada [34].

### 2.4.1 Textura

Textura é um fenômeno que se alastra sobre alguma superfície, fácil de ser reconhecido e difícil de ser definido. A visualização de um grande número de pequenos objetos pode ser definida como sendo uma textura [13]. A Figura 2.17 ilustra alguns exemplos de textura.

Figura 2.17: Exemplos de textura.

Os problemas que envolvem o uso de texturas podem ser separados em quatro classes básicas [53]:

- Segmentação baseada em texturas: é a tarefa de "quebrar"uma imagem em componentes que possuem textura constante. É uma tarefa árdua, pois inicialmente não conhecemos os tipos de textura existentes na imagem. É necessária uma maneira de diferenciar dois tipos distintos de textura. Técnicas de segmentação baseada em texturas foram utilizada por Tuceryan e Jain [33] e por Voorhess e Peggio [52] em imagens naturais e Du Buf e Kardam [11] estudou e comparou a performance de várias técnicas de segmentação baseada em texturas;

- Classificação de texturas: consiste basicamente em definir a qual categoria pertence cada tipo de textura de uma imagem observada. A princípio devemos definir cada uma das classes e extrair algumas informações da textura que servirão de parâmetro para a classificação. Em seguida um classificador de padrões irá atribuir a cada uma das texturas contidas na imagem uma das classes definidas. A classificação de texturas foi utilizada por Haralick e Shanmugam [22] na classificação de regiões em imagens de satélite e por Farrokhnia [12] em sua tese sobre inspeção automática de pintura;

- Síntese de texturas: é a tentativa de construir amplas regiões texturizadas utilizando pequenas imagens de amostra como exemplo. É um problema bastante popular na computação gráfica, onde a renderização de objetos texturizados é necessária praticamente em todas as aplicações;

- Forma da textura: consiste na recuperação das informações de orientação e forma das superfícies a partir da textura das mesmas. Stevens [50] observou que algumas propriedades da textura eram bastante significantes para a extração da geometria da superfície. Existem dois efeitos causados diretamente pela forma das superfícies, nas texturas: redução da escala dos elementos da textura *(textons)* e variação de sua intensidade. Bajcsy e Lieberman [3] usaram o gradiente dos elementos da textura para determinar a forma da superfície e Witkin [57] usou a informação de orientação das bordas da imagem para estimar a orientação da superfície.

### 2.4.2 Representação de Texturas

Uma textura geralmente consiste em um conjunto de pequenos objetos regulares que se repetem ao longo de uma superfície. Estes pequenos objetos podem ser chamados de *textons* que são a unidade básica da textura [49]. Uma maneira simples de se definir uma textura seria procurar pelos *textons* e descrever a maneira como eles estão organizados [13].

A maior dificuldade em relação a esta definição é a inexistência de um conjunto definido de *textons* a serem procurados na imagem. Uma maneira de se contornar este problema é procurar por elementos simples na textura, tais como linhas ou pontos, considerando sua localização e posicionamento espacial, o que pode ser feito através da aplicação de filtros específicos na imagem [13].

### 2.4.3 Extração de Parâmetros e Detecção de Imperfeições em Texturas

Seres humanos possuem a espetacular capacidade de encontrar, facilmente, imperfeições em superfícies texturizadas. Tal mecanismo funciona quando é conhecido o padrão ideal da textura e quais são os defeitos passíveis de ocorrer. Através da simples observação de uma superfície contendo imperfeições é possível dizer o que há de anormal com ela. Esta habilidade do sistema visual humano inspira inúmeros modelos e aplicações de visão computacional para solucionar este tipo de problema [9].

A definição dos critérios para a análise de texturas envolve muitos fatores relacionados aos *textons*, incluindo, um certo grau de tolerância para as variações de tamanho, de orientação, e formato [9].

Muitos métodos foram propostos para solucionar problemas de inspeção de textura em determinadas superfícies. Odemir *et al* [37] estudaram diversos métodos de inspeção de defeitos em produtos têxteis. Kim e Koivo [29] utilizaram detecção de imperfeições em textura para localizar e classificar os defeitos em tábuas de madeira. Serafim [46] conduziu esforços na utilização de segmentação baseada em textura no reconhecimento de imperfeições em couro.

### Detecção de Defeitos Estruturais

Chetverikov [9] especifica uma técnica para detecção de defeitos estruturais. De acordo com o autor, o processo inicia-se com a definição das regras de regularidade, que define algumas regras de como os *textons* estão organizados. Em um segundo momento é feita a detecção de exceções à estas regras, onde são encontrados os pontos da superfície texturizada que fogem do padrão da textura.

A seguir iremos detalhar as duas fases do processo de detecção de defeitos estruturais definidas por Chetverikov.

**Definição das Regras de Regularidade:** Esta fase quantifica a regularidade dos *textons* através da avaliação, em coordenadas polares, da periodicidade da função de autocorrelação, que segundo Huang [25], é correlação entre variáveis aleatórias em dois pontos distintos no espaço ou no tempo. Considerando uma imagem $I(m, n)$ de dimensões $m \times n$ e o espaço vetorial $(d_x, d_y)$, então a autocorrelação normalizada da imagem $I(m, n)$ é denotada por $\rho_{xy}(d_x, d_y)$. A normalização de $\rho_{xy}$ é realizada via FFT (*Fast Fourier Transform*), utilizando a relação conhecida entre a função de autocorrelação e a transformada de Fourier.

A representação polar $\rho_{pol}(\alpha, d)$, onde $\alpha$ representa o ângulo formado entre o vetor e o eixo $x$ e $d$ representa a magnitude do vetor, é então computada na grade de coordenadas polares $(\alpha_i, d_j)$ através da interpolação de $\rho_{xy}(d_x, d_y)$, nos valores não inteiros. A matriz resultante é denotada por $\rho_{pol}(i, j)$. Então calculamos outra matriz, chamada de mapa de interação polar, através de: $M_{pol}(i, j) = 1 - \rho_{pol}(i, j)$.

Cada linha de $M_{pol}(i, j)$ é chamada de função de contraste. Uma função de contraste $F_i(j)$ representa a variação do contraste no espaço $d_j$ na direção $\alpha_i$. Uma textura periódica, ou seja, regular, possui funções de contraste com valores e periodicidade mínima.

Para um ângulo $i$, a regularidade direcional é definida por $R(i) = [R_{int}(i).R_{pos}(i)]^2$, onde $R_{int}(i)$ e $R_{pos}(i)$ são a regularidade de intensidade e a regularidade de posição, respectivamente. $R_{pos}(i)$ reflete a regularidade (periodicidade) da maneira como os *textons* preenchem a textura, enquanto $R_{int}(i)$ indica quão regular é a intensidade dos *textons*. Os procedimentos para computar $R_{int}(i)$ e $R_{pos}(i)$ são os seguintes:

1. Aplicar um filtro para remover o ruído em $F_i(j)$;
2. Determinar a amplitude de $F_i(j)$ através da atribuição de um mínimo para cada máximo. Então, a maior amplitude $F_{max} - F_{min}$ é selecionada. E, a regularidade de intensidade é definida por $R_{int} = 1 - F_{min}/F_{max}$;
3. Encontrar as posições $j_1$ e $j_2$ em $F_i$ onde os dois valores serão mínimos e $j_1 < j_2$. Então a regularidade de posição é definida por: $R_{pos} = 1 - |1 - 2j_1/j_2|$.

Chetverikov [9] definiu as regras de regularidade da seguinte maneira: $T_k$ é a seqüência com valores de máximos locais de $R(i)$. Para selecionar o valor máximo relevante, foi aplicado uma limearização $T_{thr} = 0.15$. Dois valores foram calculados na sequência limiarizada: o valor máximo de $M_R$ e a média $\mu_R$. $0 \leq \mu_R \leq M_R \leq 1$, onde quanto mais próximo de 1 indica uma textura altamente regular e valores próximo de 0 significam o contrário, ou seja, irregular. O autor ainda acrescenta que outros parâmetros podem ser extraídos para outros tipos de análise.

**Detecção de Exceções:** As duas medidas de regularidade foram computadas para várias janelas da imagem, onde uma janela é uma pequena porção da imagem. A maioria das janelas contém o básico, padrões livre de defeito, enquanto algumas janelas podem conter defeitos. As janelas livres de defeitos produziram um aglomerado ao redor do pixel central $p_c$. Janelas com defeito resultaram em exceções as quais distam de $p_c$ excedendo o raio do aglomerado. A existência de exceções indica uma grande probabilidade de que a textura contenha defeitos.

O princípio do algoritmo está representado na Figura 2.18. A idéia surgiu da robusta técnica de regressão e detecção de exceções especificada por Rousseeuw e Leroy (1987) apud [9]. Os vetores de características das janelas são chamados de $p_i$. Os vetores são representados por pontos no espaço, onde a maior concentração desses pontos definem o aglomerado. Então é possível achar o $p_c$ do aglomerado que é o ponto o qual a distância média de

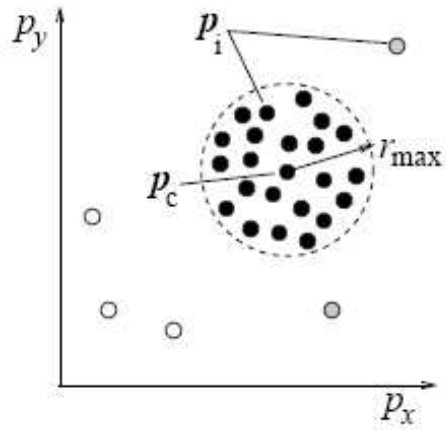


Figura 2.18: Princípio da detecção de exceções. (Fonte: [9])

todos os outros pontos é mínima: $d_{med}(i) > d_{med}(c)$ para todo $i \neq c$, onde $d_{med}(i) = media_{j \neq i} \|p_i - p_j\|$.

Um ponto $p_k$ é tido como exceção quando o raio $r_k$ ultrapassa o limiar $r_{max}$, que representa o raio do aglomerado. O autor ainda acrescenta que o cálculo de $r_{max}$ pode ser realizado de duas formas: através da simples média das distâncias de cada um dos pontos do aglomerado até o ponto central, ou então, pode ser definida pelo usuário, aumentando assim, a tolerância da detecção de defeitos em textura.

### 2.4.4 Algoritmos de Detecção de Bordas Baseados em Textura

Todas as superfícies são texturizadas em alguma escala. A maioria delas está compreendida em uma escala pequena, geralmente de 1 a 10 pixels, o que facilita a utilização de operadores aplicáveis a pequenas áreas para extrair informações das mesmas. No entanto, quando a escala da textura ultrapassa o tamanho do operador, esta abordagem fica comprometida. Então, devemos considerar operadores designados para comparar a distribuição de cores ou características das texturas que sejam aplicáveis a superfícies texturizadas de modo escalar [34].

Ruzon e Tomassi [42] propuseram o operador de compasso, originalmente designado para comparar a distribuição de cores e detectar as bordas em imagens RGB. A grande característica deste operador é a capacidade de processar a informação de distribuição das cores em grandes áreas da imagem e o uso de funções sofisticadas para compará-las. O ponto mais fraco deste

operador fica por conta do custo computacional, que, segundo Maxwell e Brubaker [34] pode chegar à ordem de 30 minutos para uma imagem em resolução 640x480 em um processador Athlon de 1.8GHz.

O poder do operador de compasso desperta o interesse em utilizá-lo para a detecção de bordas através da análise de texturas. A seguir é apresentada a idéia geral do operador de compasso.

### Operador de Compasso

O operador de compasso segue a mesma linha de raciocínio da maioria dos detectores de borda existentes, ele divide a janela que está sendo processada em duas metades e as compara para verificar se existe alguma diferença entre elas. Mas ao contrário da maioria dos detectores de borda, o operador de compasso permite que sejam utilizados vários pixels, abrangidos pelo suposto "raio do compasso", no cálculo da média dos valores dos vizinhos. Se o raio do compasso for definido para 1, então o operador de compasso se comporta de maneira idêntica aos outros operadores de detecção de bordas [42].

A medida de similaridade a ser utilizada, pode ser escolhida livremente, pode-se utilizar a distância Euclidiana ou o ângulo entre vetores por exemplo. Ruzon e Tomassi enfatizam a utilização da métrica *Earth Mover's Distance* (EDM), que, segundo eles, facilita a tarefa de trabalhar com o ponto de massa da janela que está sendo processada. Maxwell e Brubaker [34] propuseram a utilização da métrica *dynamic time-warping* (DTW) ao contrário da métrica EDM proposta por Ruzon e Tomassi [42], por questões de desempenho, que segundo eles chegava à ordem de 33 minutos para uma imagem de resolução 768 x 512 pixels.

A idéia do operador de compasso é inspirada, como o próprio nome diz, na ferramenta compasso utilizada para traçar circunferências. O centro do compasso é o pixel central, ou seja, o pixel de interesse [42].

Maxwell e Brubaker acrescentam ainda, que é possível passar a saída do detector de bordas pelo operador de compasso pelas etapas de supressão não-máxima e histerese, para obter melhores resultados.

### DWT - *Dynamic Time Warping*

A técnica de *Dynamic Time Warping* (DTW) é uma técnica de programação dinâmica, originalmente desenvolvida para calcular, de forma eficiente, o valor de correspondência entre duas sequências numéricas espalhadas no tempo[28]. Estas sequências, também conhecidas como sinais, são uma forma muito comum de representação de dados em muitas disciplinas científicas. Uma operação comumente realizada é a comparação de sinais em um certo

intervalo de tempo. Para a maioria dos domínios, uma simples medida de dissimilaridade tal como a Distância Euclidiana é suficiente. De qualquer forma, os sinais devem ter aproximadamente o mesmo formato, mas este formato não está totalmente alinhado ao longo da linha do tempo, como ilustrado pela Figura 2.19. Com o objetivo de determinar o valor de similaridade entre as sequências, devemos "entortar"a linha do tempo de uma ou de ambas, até encontrar a melhor correspondência entre elas.

Técnicas de DTW foram utilizadas por Gavrila e Davis [15] no reconhecimento de gestos, por Rabiner e Juang [39] no processamento de voz, por Gollmer e Posten [16] na avaliação de produção manufaturada e por Caiani *et. al* [4] em aplicações voltadas para a área da medicina.

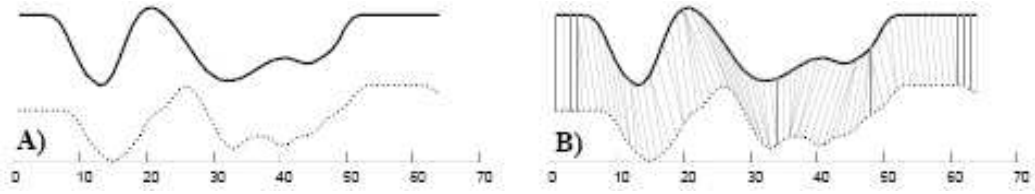


Figura 2.19: Exemplo da utilização da técnica de *Dynamic Time Warping*

**Algoritmo de *Dynamic Time Warping*:** Supondo que temos dois sinais no tempo $Q$ e $C$, de comprimentos $n$ e $m$ respectivamente, onde:

$$Q = q_1, q_2, ..., q_i, q_n \tag{2.29}$$

$$C = c_1, c_2, ..., c_j, c_n \tag{2.30}$$

Para alinhar as duas sequências usando a técnica de DTW, iremos construir uma matriz de dimensões $n \times m$, onde cada elemento na posição $(x, y)$ da matriz contém a distância $d(q_i, c_j)$ entre os pontos $q_i$ e $c_j$. Tipicamente é utilizada a Distância Euclidiana entre dois pontos, então teremos: $d(q_i, c_j) = (q_i - c_j)^2$. Cada elemento $(i, j)$ da matriz corresponde ao alinhamento entre os pontos $q_i$ e $c_j$ das sequências, assim como ilustrado na Figura 2.20. O vetor de distorção *(Warping Path)* $W$, é um conjunto contínuo de valores da matriz que define o mapeamento entre $Q$ e $C$. Cada elemento de $W$ é definido por $w_k = (i, j)_k$, assim teremos:

$$W = w_1, w_2, ..., w_k, w_n \qquad max(m, n) \leq K < m + n - 1 \tag{2.31}$$

O vetor de distorção geralmente segue algumas regras [28]:

- **Condições de fronteira:** $w_1 = (1, 1)$ e $w_k = (m, n)$, determina que o vetor de distorção começa e termina sobre a diagonal da matriz.
- **Continuidade:** dado $w_k = (a, b)$, então $w_{k-1} = (a', b')$, onde $a - a' \leq 1$ e $b - b' \leq 1$. Isto restringe o vetor de distorção para as células adjacentes (incluído células adjacentes diagonalmente).
- **Monotonicidade:** dado $w_k = (a, b)$, então $w_{k-1} = (a', b')$, onde $a - a' \geq 0$ e $b - b' \geq 0$. Isto força que os pontos em $W$ sejam monotônicamente espalhados no tempo.

Existem inúmeros vetores de distorção que satisfazem as condições acima, no entanto, estamos interessados apenas no vetor de distorção que minimiza o custo do cálculo da distorção entre os sinais, visando uma redução no tempo de processamento. Segundo Keogh e Pazzani, a melhor função que minimiza esse custo está representada pela Equação 2.32.

$$DTW(Q, C) = \min \left\{ \sqrt{\sum_{k=1}^{K} w_k} \ / \ K \right\} \tag{2.32}$$

A variável $K$ no denominador é utilizada para compensar o fato de que as sequências comparadas podem ter comprimentos diferentes.

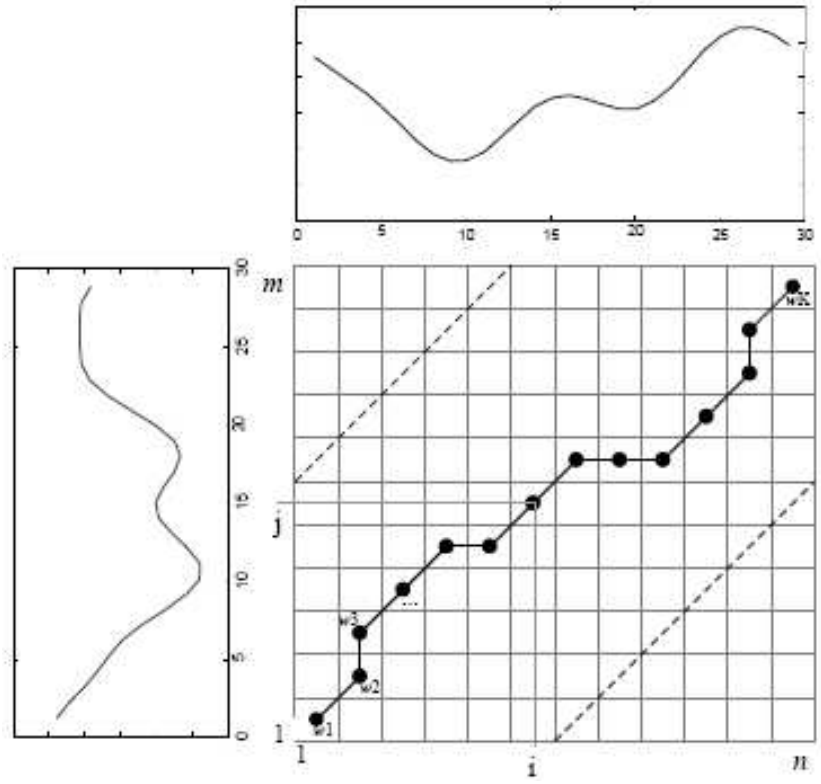


Figura 2.20: Comparação de sinais através da técnica DTW

# Capítulo 3

# Implementação

A implementação dos algoritmos utilizados neste projeto foi baseada na plataforma ImageJ *(Image Processing and Analysis in Java)*, que é um robusto manipulador de imagens desenvolvido por Wayne Rasband[1] do NIH *(National Institutes of Health)*, em linguagem Java. Seu código fonte é aberto e é disponibilizado gratuitamente na internet. Além de possuir um painel com várias ferramentas para o processamento de imagens, o ImageJ possibilita a instalação de novos módulos *(plugins)* também desenvolvidos em linguagem Java. O acesso à imagem é feito através de uma camada intermediária implementada na classe *ImageAccess*, desenvolvida Dr. Daniel Sage[2] do *Biomedical Imaging Group*, que abstrai o acesso aos dados, criando um acesso de alto nível aos pixels e demais informações da imagem.

Foram implementados algoritmos de detecção de bordas aplicáveis a imagens em tons de cinza e a imagens coloridas, utilizando as informações de cor e de textura extraídas das imagens. Também foram implementados alguns algoritmos já consolidados, como o algoritmo de detecção de Bordas de Canny e o algoritmo de Roberts e alguns métodos auxiliares, com finalidade de aprimorar os conhecimentos na área de detecção de bordas e otimizar os resultados obtidos.

## 3.1 Algoritmo de Detecção de Bordas Utilizando Informação de Cor

A cor de uma imagem é uma informação pontual, ou seja, é uma característica particular de cada um dos pixels da imagem. É possível, através

[1]email: Wayne@helix.nih.gov

[2]http://bigwww.epfl.ch/sage/index.php

da utilização de uma simples medida de dissimilaridade, implementar um algoritmo que calcula o valor máximo da diferença entre dois pixels adjacentes diagonalmente em uma máscara de dimensões $2 \times 2$. Uma síntese do algoritmo de detecção de bordas utilizando informações de cor implementado está representado pelo Algoritmo 3.1.

**Algoritmo 3.1** Detecção de Bordas em Cor

1: $Imagem \leftarrow GaussianBlur(Imagem)$
2: **para** $i = 0$ **até** Altura da Imagem **faça**
3: **para** $j = 0$ **até** Largura da Imagem **faça**
4: $pixel_1 \leftarrow InformacaoDeCor(Imagem[i][j])$
5: $pixel_2 \leftarrow InformacaoDeCor(Imagem[i+1][j+1])$
6: $pixel_3 \leftarrow InformacaoDeCor(Imagem[i+1][j])$
7: $pixel_4 \leftarrow InformacaoDeCor(Imagem[i][j+1])$
8: $Bordas[i][j] \leftarrow \max(MedidaDissimilaridade(pixel_1, pixel_2), \ldots$ $\ldots\ MedidaDissimilaridade(pixel_3, pixel_4))$
9: **fim para**
10: **fim para**

A medida de dissimilaridade a ser utilizada pode ser escolhida livremente dentre inúmeras possibilidades existentes. Durante o desenvolvimento deste projeto foram implementadas duas medidas de dissimilaridade diferentes: a distância Euclidiana entre vetores, representada pelo Procedimento 3.2, e o ângulo entre vetores, representado pelo Procedimento 3.3.

**Procedimento 3.2** Cálculo da distância Euclidiana entre vetores

**Recebe** dois vetores $\vec{v1}$ e $\vec{v2}$
1: $DistanciaEuclidiana \leftarrow \left\| \vec{v1} - \vec{v2} \right\|$

**Procedimento 3.3** Cálculo do ângulo entre vetores

**Recebe** dois vetores $\vec{v1}$ e $\vec{v2}$
1: $AnguloEntreVetores \leftarrow \vec{V_1}\vec{V_2}\ /\ \left\|\vec{V_1}\right\| . \left\|\vec{V_2}\right\|$

Devido ao fato de a cor ser uma informação pontual, o ruído na imagem, por se manifestar aleatoriamente em pixels isolados da imagem, pode afetar diretamente a comparação entre os pixels adjacentes. Foi incluído ao algoritmo uma fase de esmaecimento do ruído, através de um filtro de suavisação Gaussiano, ilustrado pelo Procedimento 3.4.

**Procedimento 3.4** Suavisação Gaussiana

1: $NucleoGaussiano \leftarrow CriaNucleoGaussiano(Raio, DesvioPadrao)$
2: $Imagem \leftarrow Convolucao(Imagem, NucleoGaussiano)$

Transformações para diversos espaços de cor também foram implementadas, afim de tornar possível a comparação entre os resultados obtidos pelos algoritmos de detecção de bordas aplicados a diferentes espaços de cor. Foi implementada uma classe em java que recebe a representação do pixel no formato de cor RGB e a transforma para um dos seguintes formatos: YUV, HSB, HSV, CIELAB, HSL, xyY, XYZ e CIELUV.

## 3.2 Algoritmo de Detecção de Bordas Utilizando Textura

A textura de uma imagem, ao contrário da cor, não é uma informação pontual, ou seja, não depende apenas da informação presente em um único ponto da imagem, mas sim envolve o conceito de vizinhança, utilizando o valor dos pixels vizinhos ao pixel de interesse durante a fase de extração da informação.

Ruzon e Tomasi [42] destacam como uma das vantagens do operador de compasso, a possibilidade da utilização de vários pixels, abrangidos pelo suposto "raio do compasso", no cálculo da informação, tordando-o viável para aplicações de extração de parâmetros de textura.

Ainda há a necessidade de uma medida de dissimilaridade capaz de calcular a diferença entre as duas metades do compasso, Maxwell e Brubaker [34] propuseram a utilização da métrica *dynamic time-warping* (DTW) por questões de desempenho.

A Figura 3.1 representa a técnica implementada, utilizando o operador de compasso com o corte realizado em $45^o$, juntamente com a técnica de DTW, para calcular o valor de dissimilaridade entre as duas metades do compasso. A Figura 3.1 **(A)** é a representação do posicionamento do operador de compasso sobre uma imagem, onde o pixel destacado é a posição onde se localiza a "agulha do compasso" e a linha vermelha é a circunferência de raio igual ao raio do compasso formada através da revolução do mesmo. A Figura 3.1 **(B)** representa a ampliação da área abrangida pelo compasso com o corte realizado na direção de $45^o$, formando as regiões $Q$ e $C$ representadas na Figura 3.1 **(C)**. O próximo passo é representar as regiões $Q$ e $C$ através de seus respectivos histogramas, como ilustrado na Figura 3.1 **(D)**. Por fim, a distância entre $Q$ e $C$ é calculada através da técnica DTW, conforme indicado na Figura 3.1 **(E)**.

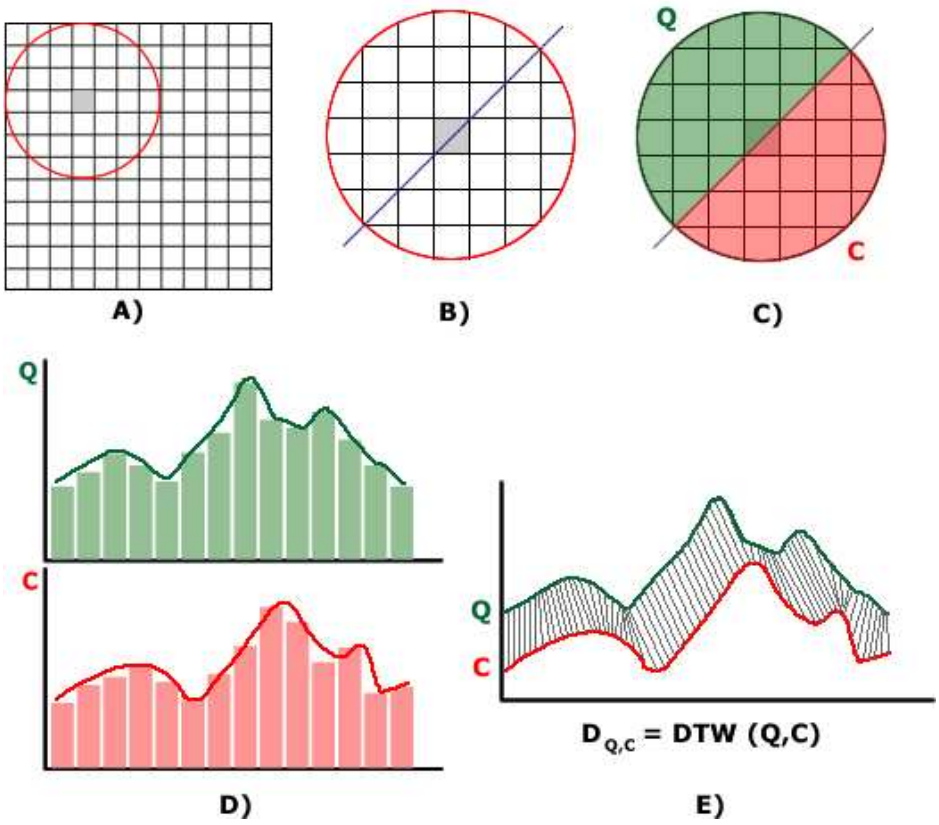


Figura 3.1: Exemplo da aplicação da técnica do operador de compasso em conjunto com a técnica de DTW.

O Algoritmo 3.5 representa a detecção de bordas utilizando a técnica do operador de compasso em conjunto com a técnica DTW.

## 3.3 Métodos Auxiliares

Com finalidade de otimizar os resultados obtidos, alguns métodos auxiliares foram implementados. O algoritmo de detecção de bordas de Canny [5] possui duas funcionalidades interessantes que foram implementadas e adaptadas para serem utilizadas por outros algoritmos de detecção de bordas, denominadas supressão de não-máximos e limiarização.

**Algoritmo 3.5** Detecção de Bordas em Texturas (Operador de Compasso em conjunto com DTW)

```
1: raio ← 3 {Define o raio do compasso para 3}
2: para i = 0 até Altura da Imagem faça
3:   para j = 0 até Largura da Imagem faça
4:     Bordas[i][j] ← 0
5:     θ ← 0^o
6:     enquanto θ ≤ 135^o faça
7:       mag = DTW(AplicaCompasso(Imagem[i][j], θ, raio))
8:       se Bordas[i][j] < mag então
9:         Bordas[i][j] ← mag
10:      fim se
11:      θ ← θ + 45^o
12:    fim enquanto
13:  fim para
14: fim para
```

### 3.3.1 Supressão de Não-Máximos de Canny

A supressão de não-máximos, proposta por Canny, consiste na eliminação dos pixels cujos valores não são máximos locais na direção perpendicular à borda, ou seja, busca-se, na direção do gradiente da imagem, por valores de pixels que são máximos locais. Este processo produz um afinamento das bordas [51].

A Figura 3.2 **(A)** ilustra o caso onde o pixel central ($L$,$C$) é examinado. O valor desse pixel é um máximo local e a direção do seu gradiente é de 45°. Para exemplificar o processo de supressão não máxima, supõe-se que uma máscara de tamanho 3x3 percorre $M_{[i,j]}$ e compara a magnitude do gradiente do pixel central com a magnitude de seus vizinhos no sentido do gradiente ($L-1$,$C+1$) e ($L+1$,$C-1$), de acordo com a Figura 3.2 **(B)**. Como o pixel central é maior que ambos os pixels envolvidos na comparação, ele será mantido, caso contrário ele seria igualado a zero [51];

O Procedimento 3.6 implementa a supressão de não-máximos utilizada pelo algoritmo de detecção de bordas de Canny.

### 3.3.2 Limiarização de Canny *(Histerese)*

A limiarização é uma etapa de processamento que tem por finalidade eliminar uma possível fragmentação das bordas. O processo é aplicado ao mapa de bordas extraído da imagem e utiliza dois limiares: limiar baixo ($L_b$) e

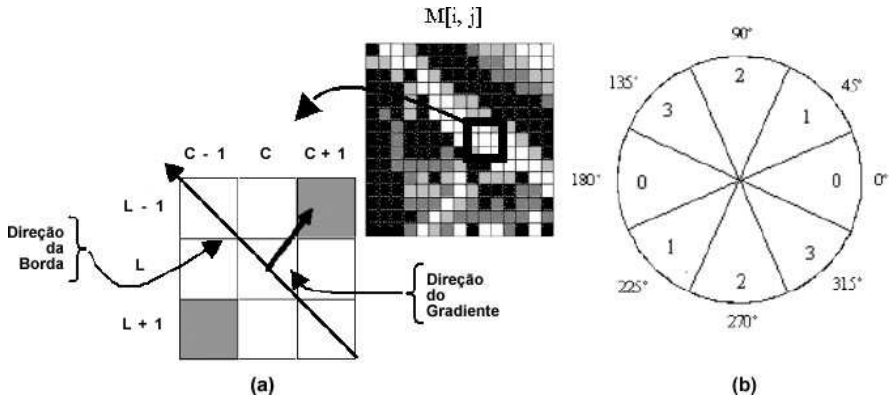


Figura 3.2: Supressão de não-máximos (Fonte: [51])

**Procedimento 3.6** Supressão de não-máximos

```
1: para i = 0 até Altura da Imagem faça
2:   para j = 0 até Largura da Imagem faça
3:     se Bordas[i][j] < LimiarBaixo então
4:       SupressaoNaoMaximos[i][j] ← 0
5:     senão
6:       se Bordas[i][j] > Vizinhos(DirecaoDoGradiente) então
7:         SupressaoNaoMaximos[i][j] ← Bordas[i][j]
8:       senão
9:         SupressaoNaoMaximos[i][j] ← 0
10:      fim se
11:    fim se
12:  fim para
13: fim para
```

limiar alto ($L_a$). A limiarização é iniciada a partir do limiar alto, que geralmente corresponde de 80% a 90% do valor máximo que um pixel pode assumir. Então todos os pixels que estiverem acima desse limiar serão classificados como pontos de borda, formando um conjunto $C_1$. O limiar baixo é aplicado para eliminar todos os pixels que estiverem a baixo de $L_b$, como ilustra a Figura 3.3. Geralmente o valor de $L_b$ varia entre a metade de $L_a$ ou um terço de $L_a$. Ao aplicar a limiarização as bordas poderão ainda ficar fragmentadas, em virtude da não uniformidade das bordas. Para resolver o problema utizamos os pixel que ficaram entre os limiares $L_a$ e $L_b$, que formam o conjunto $C_2$. O algoritmo consiste em buscar no conjunto $C_1$, a ocorrência de extremidades de contornos e, no segundo conjunto $C_2$ escolher os pontos que completam esse contorno. O algoritmo realiza tal busca até que não haja

mais fragmentos de contorno isolados em $C_1$ ou até quando não exista mais pixels aproveitáveis no conjunto $C_2$ [5].

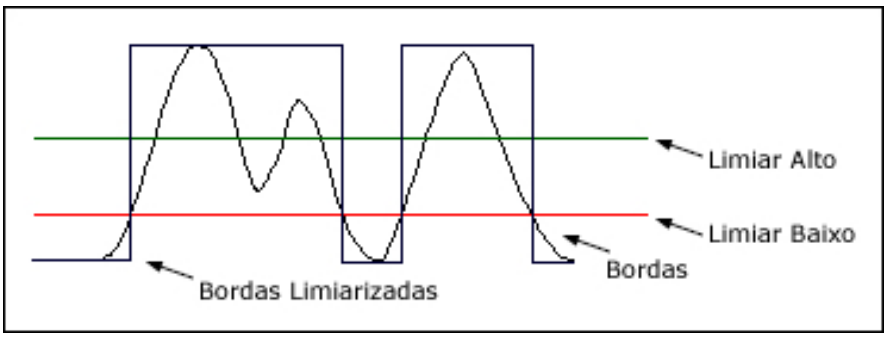


Figura 3.3: Ação da limiarização aplicada a uma borda

O Procedimento 3.7 implementa o processo de limiarização utilizado por Canny.

**Procedimento 3.7** Limiarização

```
1: para i = 0 até Altura da Imagem faça
2:   para j = 0 até Largura da Imagem faça
3:     se Bordas[i][j] > LimiarAlto então
4:       se Bordas[i][j] = ExtremidadeDeContorno então
5:         BuscaPontosDeBorda()
6:       fim se
7:     fim se
8:   fim para
9: fim para
```

# Capítulo 4

# Experimentos e Análise de Resultados

Para testar e avaliar o desempenho dos algoritmos implementados foi utilizado um computador com processador Athlon XP 1.9 GHz com 512 MB de memória principal. Foram fornecidas pela pesquisadora M. Sc. Mariana de Aragão Pereira[1], da EMBRAPA Gado de Corte, algumas fotos de imperfeições em couro bovino.

As imagens foram capturadas por meio de uma câmera digital Sony Cybershot DSC-P73 em resolução de 4.1 mega-pixels. Posteriormente foram recortadas com o programa Macromedia Fireworks e agrupadas em dois conjuntos de imagens de teste: o conjunto de amostras de imperfeições em couro crú, representado pela Figura A.1 (conjunto **A**) e o conjunto de amostras de imperfeições em couro na fase *wetblue*, representado pela Figura A.2 (conjunto **B**). As imagens não passaram por nenhuma etapa de pré-processamento, sendo utilizadas com suas configurações originais de brilho e contraste.

Por motivos de organização, todas as imagens pertinentes a este capítulo estão listadas no Anexo A.

## 4.1 Experimentos com Cor

A seguir estão representados os resultados obtidos com o algoritmo de detecção de bordas baseado em informações de cor implementado. Os testes foram realizados para três espaços de cor distintos: RGB, CIELAB e HSV. Foram utilizadas como métricas de dissimilaridade a distância Euclidiana e o ângulo entre vetores.

[1] http://www.cnpgc.embrapa.br/ mariana/

### 4.1.1 Utilizando Distância Euclidiana

Nesta seção aplicamos o algoritmo de detecção de bordas baseado em informações de cor, utilizando como medida de dissimilaridade a distância Euclidiana. Foram utilizadas para os testes a Imagem B do conjunto de amostras A e a Imagem B do conjunto de amostras B.

Para a fase de suavização do ruído por meio de um núcleo Gaussiano, o raio do núcleo ($r$) foi definido para $r = 5$, e o desvio padrão de Gauss ($\sigma$) foi definido como $\sigma = 1$. Os parâmetros da fase de supressão de não-máximos e limiarização foram determinados segundo o parecer de Canny, que sujere que o valor de $L_a$ seja aproximadamente 80% do valor máximo que um pixel pode assumir, e o valor de $L_b$ varia entre a metade de $L_a$ ou um terço de $L_a$.

Os experimentos foram realizados de duas maneiras: com as etapas de supressão de não-máximos e limiarização, e sem as etapas de supressão de não-máximos e limiarização. Os resultados dos experimentos omitindo-se as etapas de supressão de não-máximos e limiarização, realizados com as imagens selecionadas dos conjuntos de amostras A e B, podem ser observados nas Figuras A.3 e A.5, respectivamente. Para os experimentos envolvendo as etapas de supressão de não-máximos e limiarização, os resultados estão ilustrados nas Figuras A.4 e A.6, respectivamente.

### 4.1.2 Utilizando Ângulo entre Vetores

Nesta seção aplicamos o algoritmo de detecção de bordas baseado em informações de cor, utilizando como medida de dissimilaridade o ângulo entre vetores. Foram utilizadas para os testes a Imagem B do conjunto de amostras A e a Imagem B do conjunto de amostras B.

Utilizando esta medida de dissimilaridade os resultados obtidos não foram satisfatórios, portanto não foram aplicadas as etapas de supressão de não-máximos e limiarização. Os parâmetros para a suavização de ruído foram mantidos os mesmos.

Os resultados dos experimentos realizados com as imagens selecionadas dos conjuntos de amostras A e B podem ser observados nas Figuras A.7 e A.8, respectivamente.

## 4.2 Experimentos com Textura

A seguir estão representados os resultados obtidos com o algoritmo de detecção de bordas baseado em textura implementado com a união das técnicas do operador de compasso e DTW.

Os experimentos foram realizados de duas maneiras: com as etapas de supressão de não-máximos e limiarização, e sem as etapas de supressão de não-máximos e limiarização. Os resultados dos experimentos omitindo-se as etapas de supressão de não-máximos e limiarização, realizados com os conjuntos de amostras A e B, podem ser observados nas Figuras A.9 e A.10, respectivamente. Para os experimentos envolvendo as etapas de supressão de não-máximos e limiarização, os resultados estão ilustrados nas Figuras A.11 e A.12, respectivamente.

## 4.3 Análise dos Resultados

A análise visual das imagens obtidas através da aplicação dos algoritmos implementados, aos conjuntos de amostras A e B, revela alguns detalhes sobre o desempenho dos algoritmos e sobre os fatores de afetam diretamente a detecção de bordas.

Quando utilizamos as informações de cor das imagens para detectar as bordas, aplicamos duas técnicas distintas de medida de dissimilaridade: a distância Euclidiana e o ângulo entre vetores, utilizando três modelos de representação de cores: RGB, CIELAB e HSV. Os resultados obtidos com estas técnicas estão ilustrados nas Figuras A.3, A.4, A.5, A.6, A.7 e A.8.

Como foi discutido no Capítulo 2, a distância Euclidiana é mais sensível a variações das intensidades dos pixels da imagem do que o ângulo entre vetores. Podemos notar claramente este efeito através da comparação das Imagens A.6 e A.8, onde a variação de intensidade da imagem não é detectada pela técnica do ângulo entre vetores.

Os resultados obtidos nos diversos espaços de cor foram bastante parecidos, no entanto, nota-se uma leve superioridade nos resultados obtidos para o espaço de cor CIELAB com a distância Euclidiana. Isso se deve ao fato de que no espaço de cor CIELAB, a variação de intensidade é melhor discriminada pela distância Euclidiana em relação aos demais espaços de cor utilizados.

Podemos notar no conjunto de amostras A, representado pela Figura A.1, a existência de brilho natural nas imagens do couro. Este fator afeta diretamente a detecção de bordas, e poderíamos obter resultados superiores com a utilização de alguma técnica para remover o brilho das imagens originais.

Traçando um comparativo entre as Imagens A.3 (não limiarizada) e A.4 (limiarizada) concluímos que as etapas de supressão de não-máximos e limiarização contribuem significamente para a precisão da detecção de bordas. Nestas etapas são utilizados dois limiares que, segundo Canny, assumem valores que dependem da intensidade máxima assumida por um pixel. Através

de ajustes nos valores dos limiares é possível obter melhores resultados na detecção de bordas.

Quando utilizamos as informações de textura das imagens para detectar as bordas, aplicamos as técnicas do operador de compasso em conjunto com a técnica DTW. Os resultados obtidos com a aplicação destas técnicas estão ilustrados nas Figuras A.9, A.10, A.11 e A.12.

Podemos notar através da comparação das Imagens A.9 (não limiarizada), A.10 (limiarizada) que a supressão de não-máximos e a limiarização são fundamentais para a qualidade da detecção de bordas obtida. Nestas etapas também foram utilizados os valores de limiares sugeridos por Canny. Através de ajustes nos valores dos limiares é possível obter melhores resultados na detecção de bordas.

# Capítulo 5

# Considerações Finais

O presente projeto apresentou um estudo sobre as técnicas de detecção de bordas, utilizando informações de textura e cor da imagem. Como fruto deste levantamento bibliográfico foram desenvolvidos, um módulo de detecção de bordas utilizando textura e outro utilizando informações de cor, os quais serão incorporados ao projeto DTCOURO, o qual visa à construção de um detector automático de imperfeições no couro bovino, tanto no couro cru, como na forma *wetblue*.

Depois de este estudo ter sido concluído, foram realizadas implementações de algumas técnicas de detecção, e a combinação de algumas dentre todas as já apresentadas nos capítulos anteriores. Após ter convergido a um conjunto de técnicas, este foi utilizado na realização dos experimentos utilizando imagens de algumas imperfeições no couro cru e wetblue, as quais foram obtidas através do convênio com a EMBRAPA.

Tanto os resultados obtidos como os códigos estão disponíveis, junto ao projeto DTCOURO, o qual ainda encontra-se em fase de desenvolvimento.

Uma das dificuldades encontradas durante o desenvolvimento deste projeto, foi em relação ao levantamento de dados sobre a detecção utilizando-se de dados sobre textura. Pois, este tema depende, ainda, muito do poderio tecnológico ao qual se disponibiliza para a realização do experimento.

Como não foi pesquisada nenhuma métrica que pudesse mensurar precisamente os resultados obtidos, propõem-se como um trabalho futuro o estudo de técnicas que possam realizar esta métrica com precisão.

Como outras propostas para trabalhos futuros pode-se citar:

- Desenvolvimento de um método para estimar automaticamente os valores ótimos para os valores dos limiares utilizados na fase de limiarização;
- Desenvolvimento de uma técnica de pré-processamento das imagens do couro, visando a eliminação de ruídos como o brilho natural das

imagens;

- Estudo de técnicas capazes de estimar o desempenho dos algoritmos implementados.

# Anexo A

# Imagens dos Experimentos


Figura A.1: Conjunto de amostras **A** - Amostras em couro crú

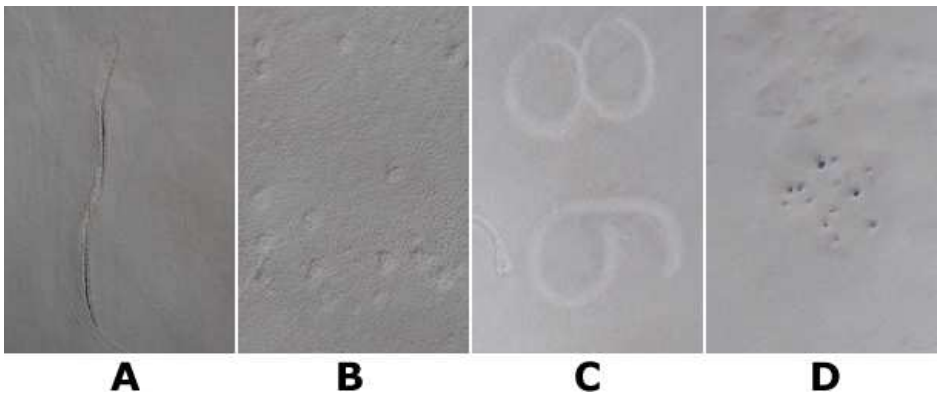


Figura A.2: Conjunto de amostras **B** - Amostras em couro na fase *wetblue*

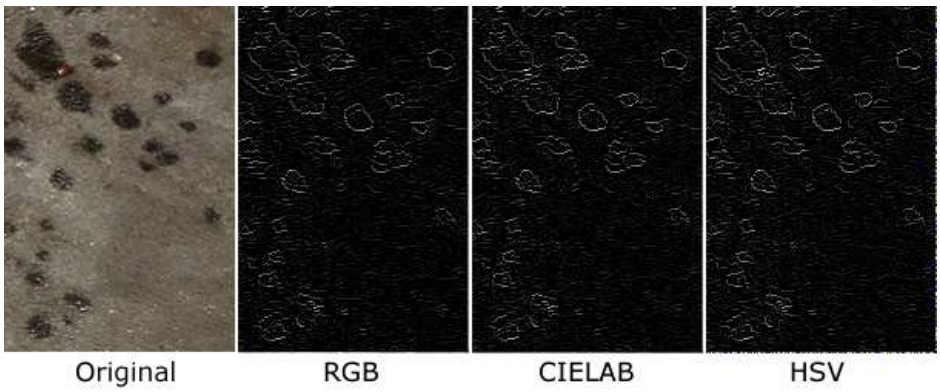


Figura A.3: Resultados para a imagem B do conjunto amostras A, utilizando cor e distância Euclidiana sem limiarização

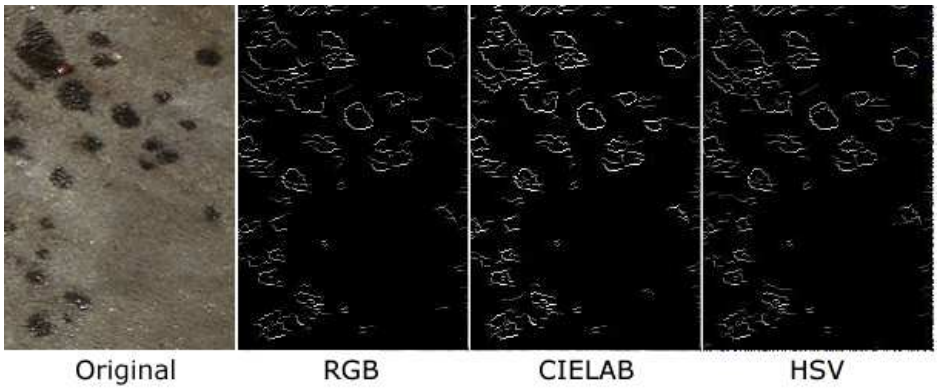


Figura A.4: Resultados para a imagem B do conjunto amostras A, utilizando cor e distância Euclidiana com limiarização

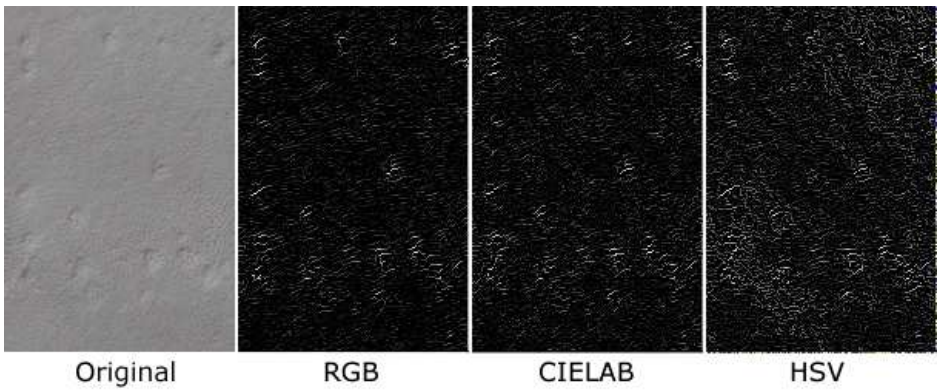


Figura A.5: Resultados para a imagem B do conjunto amostras B, utilizando cor e distância Euclidiana sem limiarização

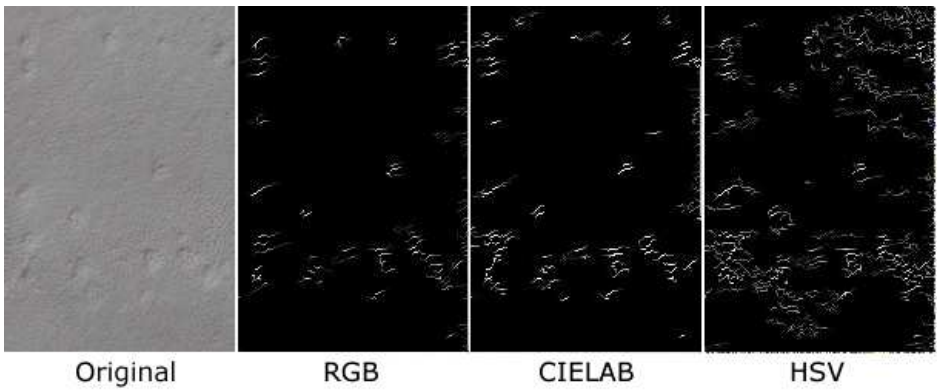


Figura A.6: Resultados para a imagem B do conjunto amostras B, utilizando cor e distância Euclidiana com limiarização

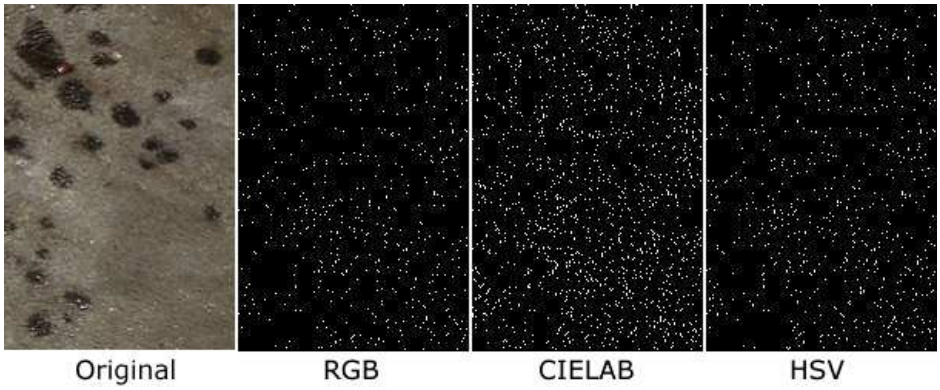


Figura A.7: Resultados para a imagem B do conjunto amostras A, utilizando cor e ângulo entre vetores sem limiarização

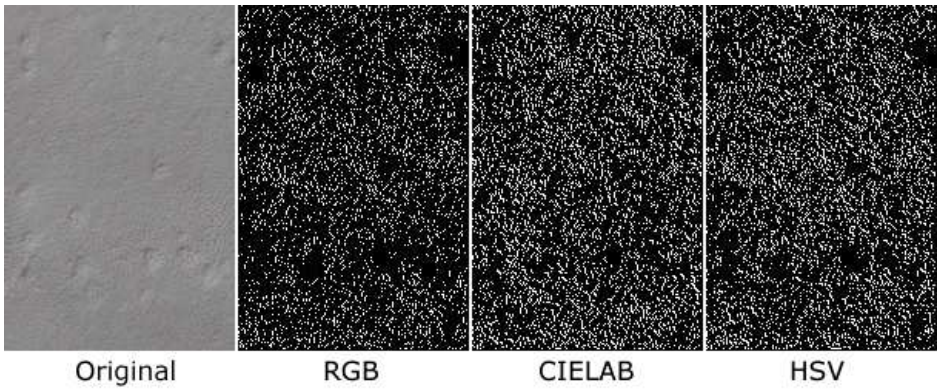


Figura A.8: Resultados para a imagem B do conjunto amostras B, utilizando cor e ângulo entre vetores sem limiarização

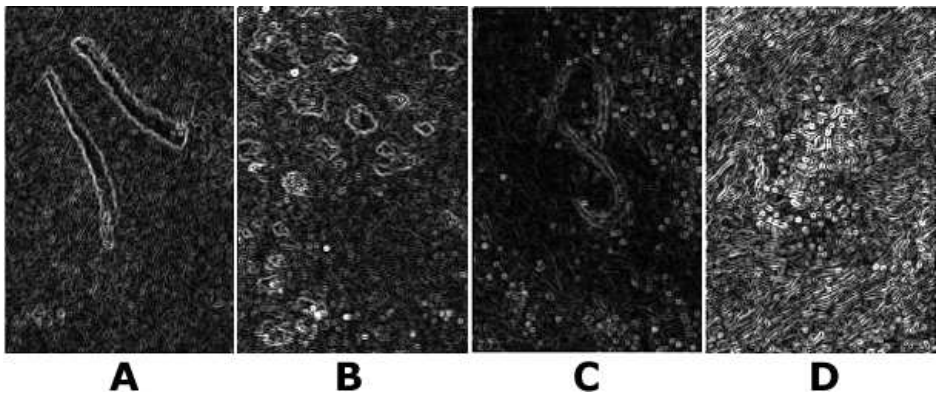


Figura A.9: Resultados para o conjunto de amostras A, utilizando textura sem limiarização

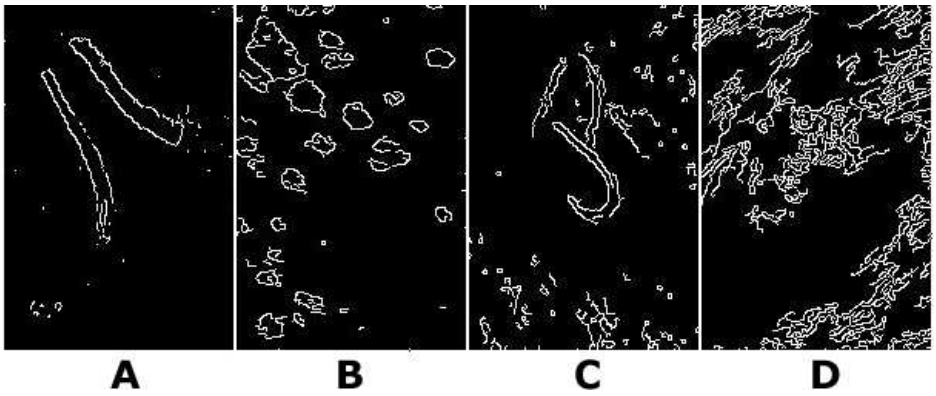


Figura A.10: Resultados para o conjunto de amostras A, utilizando textura com limiarização

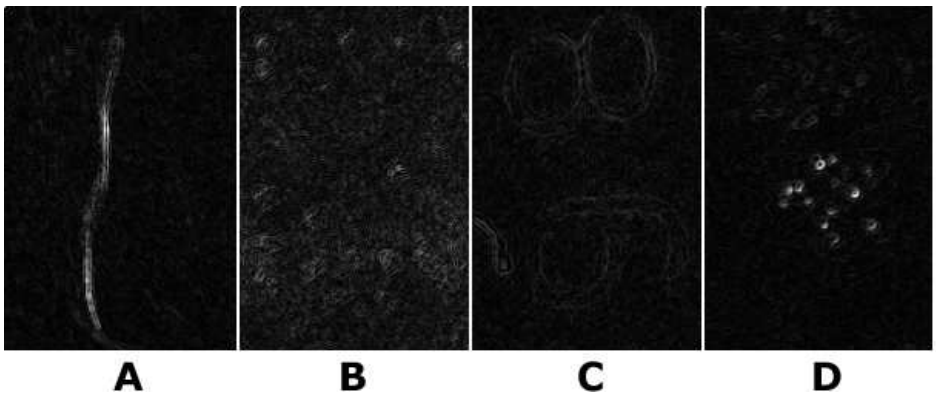


Figura A.11: Resultados para o conjunto de amostras B, utilizando textura sem limiarização

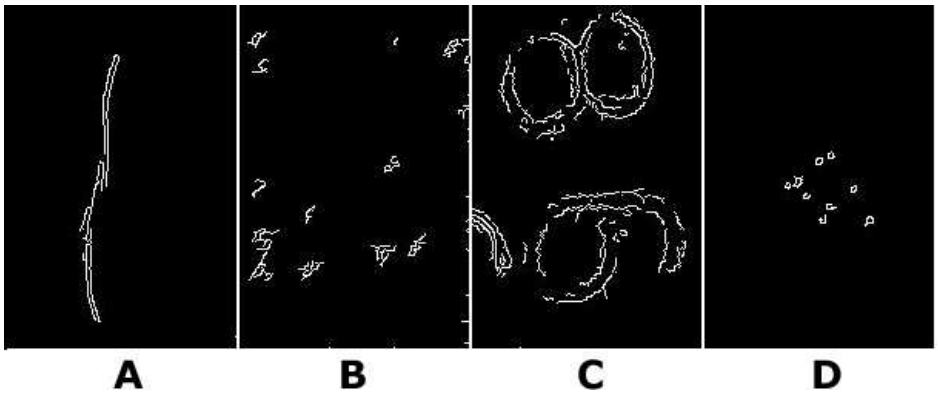


Figura A.12: Resultados para o conjunto de amostras B, utilizando textura com limiarização

# Referências Bibliográficas


[1] D. Androutsos, K. N. Plataniotis, e A. N. Venetsanopoulos. Distance measures for color image retrival. *IEEE Conference on Image Processing*, outubro 1998.

[2] S. Antani. A survey on the use of pattern recognition methods for abstraction, indexing and retrieval of images and video. *Pattern Recognition Letters*, (35):945–965, 2002.

[3] R. Bajcsy e L. Lieberman. Texture gradient as a depth cue. *Computer Graphics and Image Processing*, 5:52–67, 1976.

[4] E. G. Caiani, A. Porta, G. Baselli, M. Turiel, S. Muzzupappa, F. Pieruzzi, C. Crema, A. Malliani, e S. Cerutti. Warped-average template technique to track on a cycle-by-cycle basis the cardiac filling phases on left ventricular volume. 25, 1998.

[5] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Machine Intel*, 8(06):679–698, 1986.

[6] E. Cardoso, A. Gomes, e V. S. Lírio. Análise da cadeia produtiva de peles e couros no brasil. *EMBRAPA - Comunicado técnico*, (68):1–4, novembro 2001.

[7] T. Carron e P. Lambert. Color edge detector using jointly hue, saturation and intensity. *IEEE International Conference on Image Processing*, páginas 977–981, outubro 1994.

[8] T. Carron e P. Lambert. Symbolic fusion of hue-chroma-intensity features for region segmentation. *IEEE International Conference on Image Processing*, páginas 971–974, outubro 1996.

[9] D. Chetverikov. Structural defects: General approach and application to textile inspection. *Proceedings of the International Conference on Pattern Recognition*, 2000.

[10] R. D. Donny e S. Wesolkowski. Edge detection on color images using rgb vector angle. *Proceedings of CCECE '99*, 1999.

[11] H. Du Buf, M. Spann, e M. Kardan. Texture feature performance for image segmentation. *Pattern Recognition*, 23:291–309, 1990.

[12] F. Farrokhnia. *Multi-channel filtering techniques for texture segmentation and surface quality inspection*. Tese de Doutoramento, Computer Science Departmente of Michigan State University, 1990.

[13] D. A. Forsyth. *Computer Vision A Modern Aproach*. Prentice Hall, 2003.

[14] J. Gauch. Source code in c for the color canny operator. `http://www.iv.optica.csic.es/projects/kuim/html/edge/canny.html`, julho 1997.

[15] D. M. Gavrila e L. S. Davis. Towards 3-d model-based tracking and recognition of human movement: a multi-view approach. *International Workshop on Automatic Face and Gesture Recognition*, 1995.

[16] K. Gollmer e C. Posten. Detection of distorted pattern using dynamic time warping algorithm and application for supervision of bioprocesses. on-line fault detection and supervision in the chemical process industries. 1995.

[17] A. Gomes. Como melhorar a qualidade do couro. *Gado de corte informa*, 10(3):3, setembro 1997.

[18] A. Gomes. Aspectos da cadeia produtiva do couro bovino no brasil e em mato grosso do sul. *EMBRAPA Gado de Corte*, páginas 61–72, 2002.

[19] J. Gomes e L. Velho. Computação gráfica: Imagem. *IMPA/SBM*, página 424, 1994.

[20] R. C. Gonzalez e P. Wintz. *Digital Image Processing*. Addison-Wesley Publishing Company, 1987.

[21] R. C. Gonzalez e R. E. Woods. *Digital Image Processing*. Addison Wesley, 1993.

[22] R. M. Haralick, K. Shanmugam, e Dinstein I. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, páginas 610–621, 1973.

[23] M. Heath, S. Sarkar, T. Sanocki, e K. Bowyer. Comparison of edge detectors: Metodology and initial study. *Computer Vision and Image Understanding*, 69(1):38–54, janeiro 1998.

[24] P. S. Heckbert. Color image quantization for frame display. *ACM SIGGRAPH '82 Proceedings*, 16(3):297–307, 1982.

[25] K. Huang. *Statistical Mechanics*. Wiley, 2nd ed. edição, 1987.

[26] IEL Instituto Euvaldo Lodi, CNA Confederação Nacional da Agricultura, e SEBRAE/NACIONAL Serviço Brasileiro de Apoio às Micro e Pequenas Empresas. Estudo sobre a eficiência econômica e competitividade da cadeia da pecuária de corte no brasil. página 398, 2000.

[27] D. W. Jacobs. Classification with nonmetrics distances: Image retrieval and class representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6), junho 2000.

[28] J.K. Keogh e M. J. Pazzani. Derivative dynamic time warping. 2001.

[29] C. K. Kim e A. J. Koivo. Hierarchical classification of surface defects on dusty wood boards. *Pattern Recognition '90 Proceedings 10th International Conference*, 1:775–779, 1990.

[30] B. Lauterbach e A. Anheier. Segmentation of scanned maps in uniform color spaces. *MVA '94 IAPR Workshop on Machine Vision Applications*, páginas 322–325, 1994.

[31] A Lavoura. 60% dos defeitos no couro do boi ocorrem na fazenda. 105(640):42, março 2002.

[32] H. Levkowitz. Color theory and modeling for computer graphics, visualization and multimedia applications. *Kluwer Academic Publishers*, 1997.

[33] Tuceryanm M. e A. K. Jain. Texture segmentation using voronoi polygons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):211–216, 1990.

[34] B. A. Maxwell e Brubaker S. J. Texture edge detection using the compass operator. 2003.

[35] MAPA Ministério da Agricultura, Pecuária e Abastecimento. Instrução normativa nº 12, de 18 de dezembro de 2002. estabelece critérios de classificação para qualificação do couro bovino visando sua valorização

comercial e dá outras providências. *Diário Oficial (da República Federativa do Brasil*, dezembro 2002.

[36] MDIC Ministério do Desenvolvimento, Indústria e Comércio Exterior. Couro do brasil, a busca da qualidade. página 35, 2002.

[37] S. Odemir, A. Baykut, R. Meylani, A. Ercil, e A. Ertuzun. Comparative evaluation of texture analysis algorithms for defect inspection of textile products. *Pattern Recognition '98 Proceedings 14th International Conference*, 2:1738–1740, 1998.

[38] J. R. Parker. *Algorithms for image processing and computer vision*. John Wiley & Sons, Inc., 1997.

[39] L. Rabiner e B. Juang. Fundamentals of speech recognition. 1993.

[40] Y. Rui. Image retrieval: Past, present and future. *International Symposium on Multimedia Information Processing*, dezembro 1997.

[41] J. C. Russ. *The Image Processing Handbook*. CRC Press, 4th ed. edição, 2002.

[42] M. A. Ruzon e C. Tomasi. Color edge detection with the compass operator. *IEEE Conference on Computer Vision and Pattern Recognition '99*, 2:160–166, junho 1999.

[43] S. S. Saloum. Arabic hand-written text recognition. *IEEE Transactions on Image Processing*, página 106, 2001.

[44] R. Schettini. A segmentation algorithm for color images. *Pattern Recognition Letters*, 14:499–506, junho 1993.

[45] N. Sebe e M. S. Lew. Color-based retrieval. *Pattern Recognition Letters*, (22):223–230, 2001.

[46] A. F. L. Serafim. Segmentation of natural images based on multiresolution pyramids linking of the parameters of an autoregressive rotation invariant model: Application to leather defects detection. *Pattern Recognition '92 Proceedings 11th IAPR International Conference*, 3:41–44, 1992.

[47] SEBRAE/MS Serviço Brasileiro de Apoio às Micro e Pequenas Empresas. Cadeia produtiva de carne bovina e o mato grosso do sul (documento final). página 54, 2001.

[48] L. Shafarenko, M. Petrou, e J. Kittler. Automatic watershed segmentation of randomly textured color images. *IEEE Transactions on Image Processing*, 6:1530–1544, novembro 1997.

[49] H. Y. Shum. In search of textons. *IEEE Proceedings of the Shape Modeling International*, 2003.

[50] K. A. Stevens. Surface perception from local analysis of texture and contour. *MIT Technical Report*, 1990.

[51] G. M. Vale e A. P. D. Poz. Processo de detecção de bordas de canny. *Bol. Ciênc. Geod.*, 8(2):67–78, 2002.

[52] H. Voorhees e T. Poggio. Detecting textons and texture boundaries in natural images. *In Proceedings of the First International Conference on Computer Vision*, páginas 250–258, 1987.

[53] PSP Wang, editor. *The Handbook of Pattern Recognition and Computer Vision*. World Scientific Publishing Co., 2nd ed. edição, 1998.

[54] A. V. Wangenheim. Detecção de bordas - visão computacional - aldo von wangenheim's homepage. `http://www.inf.ufsc.br/~visao/bordas.pdf`, março 2005.

[55] G. S. Watson. Statistics on spheres. *Wiley Interscience*, 1983.

[56] S. Wesolkowski e M. E. Jernigan. Color edge detection in rgb using jointly euclidean distance and vector angle. *Vision Interface '99*, páginas 9–16, maio 1999.

[57] A. P. Witkin. Recovering surface shape and orientation from texture. *Artificial Intelligence*, 17:17–45, 1981.

[58] W. Woelker. Image segmentation based on an adaptive 3d analysis of the cielab color space. *Visual Communications and Image Processing '96*, 2727:1197–1203, 1996.

[59] Y. Yang. Color edge detection and segmentation using vector analysis. *Master's thesis Electrical and Computer Engineering*, 1995.