UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE TECNOLOGIA

FACULDADE DE ENGENHARIA DA COMPUTAÇÃO E TELECOMUNICAÇÕES

Estudo da redução do tempo de execução de uma técnica semiautomática de segmentação de imagens.

Autor: Danilo Henrique Costa Souza

Orientador: Prof. Dr. Ronaldo de Freitas Zampolo

Belém/PA, 11 de setembro de 2015.

UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE TECNOLOGIA

FACULDADE DE ENGENHARIA DA COMPUTAÇÃO E TELECOMUNICAÇÕES

Estudo da redução do tempo de execução de uma técnica semiautomática de segmentação de imagens.

Autor: Danilo Henrique Costa Souza

Orientador: Prof. Dr. Ronaldo de Freitas Zampolo

Disciplina: Trabalho de Conclusão de Curso

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do Grau de Bacharel em Engenharia da Computação pela Universidade Federal do Pará.

Belém/PA, 11 de setembro de 2015.

Estudo da redução do tempo de execução de uma técnica semiautomática de segmentação de imagens.

Autor: Danilo Henrique Costa Souza

Banca examinadora:

Prof. Dr. Ronaldo de Freitas Zampolo
(Orientador – Engenharia da Computação)

Prof. Dr. A DEFINIR
(Membro – Ciência da Computação)

Prof. Dr. A DEFINIR
(Membro – Engenharia da Computação)

Agradecimentos

Resumo

PALAVRAS–CHAVE:

Abstract

KEYWORDS:

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Capítulo 2

Processamento de imagem

2.1 Introdução

A III revolução industrial, na década de 1970, proporcionou diversos avanços tecnológicos nas mais diversas áreas da ciência, inclusive abrindo espaço novos ramos de estudo, como por exemplo a Inteligência Artificial que teve seus princípios imaginado por Alan Turing mas que só pode ser propriamente desenvolvida depois dos avanços alcançados por esta revolução. Outra área que se beneficiou desta revolução foi a tecnologia de sensores (e.g, sensores ópticos, de luz, de temperatura, de pressão, de resistência entre outros) devido ao grande avanço no processo de fabricação de *chips* e componentes tornando possível a digitalização desses dispositivos, aumento assim sua precisão e reduzindo seu tamanho a níveis microscópicos nos dias de hoje.

O avanço da tecnologia de sensores e armazenamento digital abriu possibilidades para diversas aplicações e dispositivos, inicialmente à nível militar e de pesquisa, como por exemplo a câmera digital que se tornou possível devido a criação de sensores de luminosidade mais precisos e principalmente ao armazenamento digital de informações em dispositivos de memória menores e com maior densidade, haja visto que o conceito de imagem digital reside no fato de que a imagem deve ser armazenada digitalmente.

Ao final de década e 1990 e ínio dos anos 2000 começou a popularização das câmeras digitais e seu uso e suas vantagens começaram a ser mais difundidos. Neste ponto as memórias digitais já haviam avançado o suficiente para que as pessoas pudessem tirar algumas dezenas/centenas de fotos com suas câmeras. Na década de 2010 as câmeras digitais entraram definitivamente para a vida das pessoas com a popularização dos *smartphones* e paralelo a isso houve também um aumento desses dispositivos principalmente em aplicações relacionadas à

segurança como monitoramento de tráfego nas cidades, controle de velocidade, segurança de propriedades privadas, entre outros. A popularização das câmeras digitais em diversas áreas acaba por gera uma grande quantidade de conteúdo que muitas vezes não está organizado da melhor forma possível para interpretação do usuário.

2.1.1 Por que utilizar processamento de imagem?

Esse conteúdo gera a necessidade de técnicas de processamento e análise de imagem cada vez mais eficientes e robustas para que se possa aproveitar esse mundo de imagens geradas da melhor forma possível, portanto com o avanço da tecnologia de digitalização e armazenamento de imagens surge também a necessidade de pesquisa e melhorias nas técnicas para melhor aproveitar o material produzido, um exemplo prático é o reconhecimento automático de placas de automóveis em radares de detecção de velocidade para aplicação de multas. Em sistemas modernos o radar detecta que um veículo está acima do limite permitido e imediatamente tira uma foto da traseira do veículo, esta imagem é então processada por um algoritmo de segmentação de imagem que vai detectar a placa do veículo e posteriormente segmentar cada elemento da placa (números e letras) em imagens diferentes e enviar estas imagens para um algoritmo classificador que irá identificar a placa propriamente dita e enviar essa informação ao sistema principal para que a infração seja gerada para o dono do veículo.

O exemplo apresentado mostra uma das principais vantagens de utilizar processamento de imagem no dia-a-dia, a automatização de sistemas, proporcionando eficiência, fluidez e praticidade em atividades repetitivas que antes precisavam da intervenção humana para analisar visualmente e então tomar decisões, já não necessitam de intervenção com o advento da visão computacional, [1], entretanto para que as decisões sejam tomadas de forma correta e os efeitos desse tipo de automação sejam positivos é necessário que as técnicas empregadas atendam à demanda existente, partindo dessa visão pode-se considerar a área de processamento de imagem como um meio para se chegar a visão computacional, conde máquinas analisam elementos visualmente para tomada de decisão.

O uso de máquinas para tomada de decisões baseadas em imagens é importante não apenas para automatizar e tonar determinadas tarefas mais eficientes mas também para analisar imagens que não são visíveis ao olho humano (i.e, infra-vermelho e ultra-violeta), a Figura 2.1 mostra o espectro electromagnético, onde é possível perceber que a luz visível ao olho humano é apenas uma pequena parte de todo o espectro e por isso a importância de sistemas

computacionais eficientes para interpretar e analisar imagens do espectro não visível, como as de raio gamma e raios-s, muito comuns em estudos astronômicos.

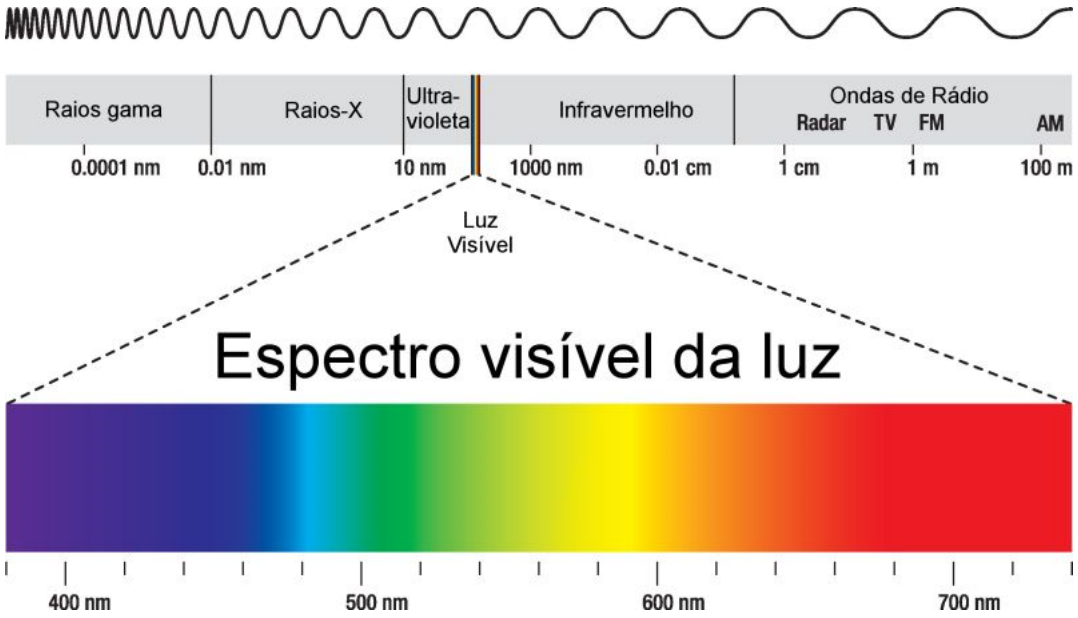

FIGURA 2.1: Espectro electromagnético. Ilustração: Peter Hermes Furian / Shutterstock.com

As seções seguintes irão detalhar os principais conceitos relacionados à processamento digital de imagens bem como as principais técnicas existentes e suas aplicações, sendo reservado uma seção para segmentação de imagens, que é o foco principal deste trabalho.

2.1.2 Definição e principais conceitos

Definição

(Gonzáles et. al) definem em [2] que processamento digital de imagens têm dois focos principais: (1) Melhorar a qualidade para percepção humana e (2) Processar a imagem para armazenamento, transmissão e representação para percepção de máquinas autônomas.

Ainda segundo [2], não existem fronteiras bem definidas no campo de estudos entre processamento de imagens e visão computacional, entretanto pode-se considerar três tipos de processos informatizados nesse caminho: (1) Baixo, (2) Médio e (3) Alto nível que serão detalhados a seguir:

1. **Processos de Baixo nível:** Envolve operações primitivas com imagens, tais como redução de ruído, aprimoramento de contraste e aguçamento de

imagem. Este tipo de processo tem como característica o fato de que ambas entrada e saída são imagens.

2. **Processos de Médio nível:** Este processo envolve operações mais complexas como segmentação (particionamento da imagem em regiões ou objetos) e descrição desses objetos em um formato entendível para o processamento por computadores, além de classificação (reconhecimento) desses objetos. Normalmente as entradas deste processo são imagens e as saídas são atributos extraídos das imagens (e.g, bordas, contornos, e a identidade de objetos).

3. **Processos de Alto nível:** Este processo envolve a interpretação de objetos identificados e/ou reconhecidos pelas etapas anteriores, ou seja, dar significado a partir de funções cognitivas associadas à visão.

Conceitos básicos

O conceito de processamento digital de imagem pode ser definido com o conjunto formado pelos três processos acima citados, criando assim uma ferramenta robusta capaz de extrair informações do mundo real, adequá-las a um formato apropriado (Baixo nível), extrair informações e parâmetros (Médio nível), analisar e tomar decisões com base nas informações coletadas (Alto nível). Tomando o exemplo dos radares de controle de velocidade mencionados na seção anterior para ilustrar o uso desses níveis na hora de estruturar uma ferramenta é possível fazer as seguintes associações:

- **Processo de Baixo nível:** Aplicação de filtros para melhorar o contraste da imagem a fim de facilitar a identificação de onde está a placa do automóvel.

- **Processo de Médio nível:** Segmentar as letras/números da placa do veículo, transformar cada caractere da placa em um arquivo, por exemplo, por fim classificá-los (i.e, identificar quais são as letras/números presentes naquela placa em análise).

- **Processo de Alto nível:** De posse dos caracteres, o sistema desenvolvido irá identificar (reconhecer) que aquele array de caracteres naquela ordem significa a placa de um automóvel e enviará a informação da placa (no formato correto) para o sistema que irá gerar o auto de infração.

Para aprofundar os conhecimentos em processamento digital de imagens é necessário primeiramente o entendimento de alguns conceitos básicos que serão utilizados posteriormente para o entendimento de técnicas mais complexas, os principais conceitos são:

- **Imagem:** Segundo [2] uma imagem é definida como uma função bi-dimensional, $f(x, y)$, onde x e y são coordenadas espaciais e a amplitude de f em qualquer par de coordenadas (x, y) é chamada de intensidade ou nível de cinza da imagem naquele ponto.

- ***Pixel:*** É a menor unidade de uma imagem digital, ou seja, corresponde a um par de coordenadas (x, y), representando um ponto espacial dentro da imagem.

- **Texel:** É o elemento fundamental de uma textura, formado por um conjunto de *pixels* que agrupados segundo uma ordem lógica formam a textura de uma imagem.

- **Textura:** De acordo com [1] a textura de uma imagem é um conjunto de métricas calculadas no processamento da imagem designadas para quantificar a textura perceptível de uma imagem. Essa textura guarda informações referentes ao arranjo espacial de cores ou intensidades em uma imagem ou em uma região selecionada da imagem.

- **Níveis de cinza:** Níveis para medir a intensidade (valor) de um *pixel* em uma imagem digital, normalmente são valores no intervalo $[0, 255]$

- **Filtragem espacial:** Operação de convolução ou correlação (no domínio espacial) entre uma imagem e uma máscara (filtro), onde esta pode ter diversos formatos utilizados para suavização ou aguçamento de imagens, essa operação será detalhada logo abaixo

Operações básicas

Para realizar qualquer operação em uma imagem é necessário primeiro entender o conceito de janela ou elemento estruturante, este elemento consistem em uma matriz, que representa uma máscara para filtragem ou um operador morfológico. Existem dois tipos principais de operações fundamentais que podem ser efetuadas sobre uma imagem e são elas: (1) correlação e convolução, onde a diferença entre a primeira e a segunda é que na convolução a máscara é rotacionada em 180°, por definição da própria operação, [3] e (2) Operações

morfológicas. O primeiro tipo de operação envolve cálculos matemáticos enquanto que o segundo está associado com operações lógicas.

A forma mais comum do uso das janelas são utilizar elementos quadrado ($N_{linhas} = N_{colunas}$) para que exista o elemento central, isto ocorre devido ao seu funcionamento que se dá da seguinte maneira: A janela é posicionada sobre parte da imagem de tal forma que seu *pixel* central fique sobre o primeiro *pixel* (e.g, ponto x_1, y_1) da imagem de entrada e partir disso os elementos da janela são multiplicados por seus correspondentes na imagem, os resultados são então somados e o resultado final é armazenado no ponto x_1, y_1 da imagem de saída, este procedimento será explicado com mais detalhes.

Apesar de serem muito utilizadas, operações com janela geram um problema no que diz respeito às bordas da imagem que ocorre quando o pixel central do operador (janela) está sobre os *pixels* mais extremos da imagem fazendo com que parte da janela fique para fora da imagem, o que acarreta em uma operação sem valor para o resultado final pois a janela estaria atuando em cima de valores que não fazem parte da função $f(x, y)$ (imagem). Para resolver este problema existem 2 abordagens clássicas:

1. Expandir a imagem de tamanho, em $a = \frac{(m-1)}{2}$ linhas em cima e embaixo e $b = \frac{(n-1)}{2}$ colunas nas laterais, onde NxM é o tamanho da janela.

2. Reduzir a imagem de saída em $a = \frac{(m-1)}{2}$ linhas em cima e embaixo e $b = \frac{(n-1)}{2}$ colunas nas laterais da imagem utilizando o como *pixel* inicial o ponto $(x + a, y + b)$

Nas operações de correlação e convolução a janela representa o filtro ao qual a imagem será submetido, este procedimento é muito utilizado para realizar suavizações (filtros do tipo passa-baixa) ou aguçamentos na imagem (filtros do tipo passa-alta) devido à sua facilidade de implementação computacional. Os filtros mais comumente utilizados são matrizes quadradas e de tamanho ímpar para que haja a presença de um elemento central e caso o filtro $h(m\ X\ n)$, onde m e n são o tamanho do filtro, seja simétrico as operações de convolução e correlação são equivalentes. Estas operações são muito utilizadas na etapa de pré-processamento de imagens, para eliminação de ruído, realce de bordas, suavização de imagens, existem diversos tipos de filtros com objetivos específicos (e.g, detecção de texturas) que serão explicados mais detalhadamente adiante neste capítulo.

A Figura 2.2 mostra um exemplo prático da convolução da Imagem I pelo filtro h, resultando na imagem de saída Y, neste exemplo foi utilizada a

abordagem (2), cita acima, para contornar o problema de borda. Generalizando este procedimento é possível chegar à Equação 2.1 para definir de forma genérica a imagem Y em função da imagem de entrada I e do tamanho do filtro.

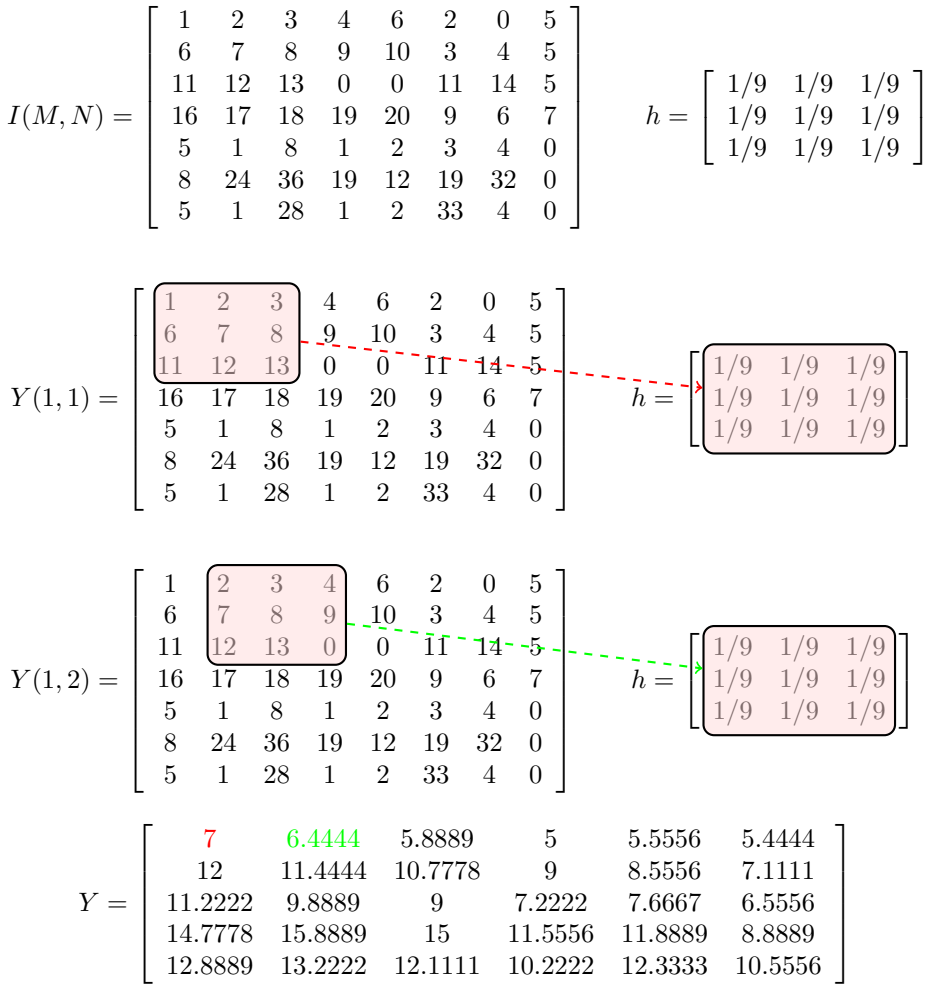

FIGURA 2.2: Exemplo de convolução/correlação 2-D

$$Y(x,y) = \sum_{s=1}^{m}\sum_{t=1}^{n} I([(x+a) - \frac{m+1}{2} + s], [(y+b) - \frac{n+1}{2} + t])h(s,t) \qquad (2.1)$$

Dentro do conjunto de operações básicas existem também as operações de erosão ($\ominus$) e dilatação ($\oplus$) da imagem, utilizadas, respectivamente, para remover objetos menores que o elemento estruturante e preencher espaços vazios, entretanto essas operações resultam, respectivamente, em encolhimento e aumento indesejável dos objetos restantes nas imagens e para resolver este problema utiliza-se as duas técnicas: (1) Abertura e (2) Fechamento. Considere a matriz A e o elemento estruturante B, ambos binários, erosão e dilatação de A

por B são representadas, respectivamente, por $A \ominus B$ e a dilatação por $A \oplus B$ e podem ser definidas como:

- **Erosão:** O elemento estruturante "anda" pela imagem tendo como origem seu elemento central, como se fosse um filtro, e para cada posição (x, y) de B dentro de A, a saída $g(x, y)$ será igual a 1 se os elementos diferentes de zero de B e de A para aquela janela (que possui tamanho igual ao de B) coincidirem, ou seja, se B encaixar em A'.

  A Figura 2.4 ilustra $A \ominus B$, na Figura 2.4a é possível visualizar que os elementos em verde de B coincidem com os elementos em verde de A (considerando a origem de B o ponto $B(2, 2)$, entretanto há elementos em B (marcados em vermelho) que estão fora de A ou não coincidem com elementos de A diferentes de zero, nesse caso a erosão não ocorre e portanto $g(1, 1) = 0$.

- **Dilatação:** O algoritmo é parecido com o da erosão à exceção de que para que a saída seja '1' o elemento estruturante B não precisa encaixar na janela atual de A mas apenas tocá-la. A Figura 2.5 ilustra a dilatação de A por B. É possível visualizar na Figura 2.5a que nenhum elemento não nulo de A coincide com os elementos não nulos de B, logo $g(1, 6) = 0$, enquanto que na Figura 2.5b o elemento $B(3, 2)$ coincide com um dos elementos não nulos de A, portanto $g(2, 5) = 1$.

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

FIGURA 2.3: Matriz A e elemento estruturante B

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

(a) Caso onde não ocorre a erosão

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

(b) Caso onde ocorre a erosão

$$g(x,y) = A \ominus B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(c) Resultado final de s$A \ominus B$

FIGURA 2.4: Erosão de A por B. A Figura 2.4a mostra o caso em que a erosão não ocorre, a Figura 2.5b mostra o caso em que a erosão ocorre e por fim a Figura 2.4c mostra o resultado final de $A \ominus B$.

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

(a) Caso onde não ocorre a dilatação

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

(b) Caso onde ocorre a dilatação

$$g(x,y) = A \oplus B = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

(c) Resultado final de s$A \ominus B$

FIGURA 2.5: Dilatação de A por B. A Figura 2.5a mostra o caso em que a dilatação não ocorre, a Figura 2.5b mostra o caso em que a dilatação ocorre e por fim a Figura 2.4c mostra o resultado final de $A \oplus B$.

2.1.3 Aplicações de processamento de imagem e principais técnicas

2.2 Segmentação de imagens

2.2.1 Tipos de segmentação

2.2.2 Principais aplicações

2.2.3 Principais técnicas

Capítulo 3

A técnica estudada e Metodologia

3.1 A escolha da técnica

Diversas técnicas de segmentação de imagens foram apresentadas neste trabalho, entretanto, uma técnica em especial chama a atenção pois permite a interação do usuário de forma mais ativa onde o objeto de interesse é marcado, usando marcação simples do tipo pincel com uma cor e o fundo ou outros objetos na imagem são marcados de outra cor, a técnica então se encarrega de separar regiões de interesse com marcações em comum (i.e, de mesma cor) de tal forma que a imagem resultante é a subtração da imagem original por todas as regiões exceto a região de interesse, ou seja, cada *pixel* da imagem é classificado como pertencente a uma determinada região.

A classificação é feita em função da probabilidade calculada com base na Função Densidade de Probabilidade (FDP) e da distância de um ponto para uma região de interesse qualquer.

Conforme descrito em [4] uma grande vantagem deste técnica é que as marcações não precisam ser minuciosas (i.e, *pixel* a *pixel*), elas precisam apenas representar as características de cor e/ou textura das regiões de interesse (e.g, se o fundo de uma imagem não é uniforme então as regiões não-uniformes devem ser marcadas separadamente mas pertencendo à mesma região). A intervenção do usuário facilita o processo de segmentação tornando-o mais simples e eficiente. Essas marcações podem ser consideradas como uma heurística para o algoritmo que é simplificado baseado nessas premissas.

Para o caso a imagem seja complexa (i.e, o fundo e os objetos de interesse possuem cores e/ou texturas parecidas) se faz necessária uma marcação mais abrangente e que marque de forma mais clara a posição dos objetos, conforme mostrado na Figura 3.1 , para que a distância exerça uma influência maior

(a) Fundo da Imagem

(b) Objeto 1

(c) Objeto 2

FIGURA 3.1: Exemplo de marcação de uma figura complexa. A Figura 3.1a mostra a marcação do fundo da imagem, a Figura 3.1b mostra um dos objetos de interesse e a Figura 3.1c mostra o outro objeto de interesse.

na classificação. É possível perceber na Figura 3.1 que os objetos em si são não-uniformes e não apenas o fundo, resultado em uma marcação bastante extensa (i.e, com muitos *pixels*) o que resultará em uma aumento no tempo de execução, uma vez que para se calcular a menor distância de um ponto à uma região é necessário calcular a distância deste ponto para todos os *pixels* da região em questão.

Conforme mostrado acima a técnica apresentada em [4] é bastante robusta e pode ser usada tanto em imagens simples (i.e, uniformes e com poucas cores) quanto em imagens complexas (i.e, nao-uniformes e com cores semelhantes no objeto e no fundo) e isso ocorre basicamente porquê a técnica se baseia em dois pilares fundamentais, a probabilidade e a distância de um *pixel* para uma região específica.

3.2 Descrição da técnica estudada

3.2.1 Segmentação de regiões uniformes

A técnica implementada neste trabalho, introduzida em [4], pode ser classificada como semi-automática pois necessita da intervenção do usuário para marcar as regiões de interesse da imagem. Estas regiões podem ser objeto ou fundo, havendo a possibilidade de se marcar mais de um objeto para segmentação ou marcar objetos não-uniformes, nesse caso a imagem final seria a soma das imagens de cada região não-uniforme separada.

O algoritmo consiste em encontrar a menor distância entre cada *pixel* da imagem de entrada e as regiões marcadas, isso é feito calculando a distância geodésica (que nesse caso é euclidiana, ou seja, uma reta entre os dois pontos de interesse) de cada *pixel* para os pontos da região marcada ponderada por um peso Ω, chamado de peso geodésico, calculado a partir dos valores dos *pixels* das regiões marcadas. Para que um ponto seja considerado de uma determinada região tanto a sua distância para a região quando a sua intensidade são levados em consideração.

Partindo da premissa de que as regiões de interesse a serem definidas são bem distintas em termos de cor e textura e utilizando o conjunto de *pixels* marcados Δ_l, $\forall\ l = 1, 2, 3, \ \ldots\ , N_l$, sendo N_l o número de regiões distintas, é calculada a FDP (Função Densidade de Probabilidade), neste caso foi utilizada a função gaussiana, mostrando a probabilidade de um ponto *p(x,y)* pertencer a uma determinada região l. Com base nessas distribuições são calculados pesos (ω_i) para cada canal da imagem que serão explicados mais detalhadamente. A Figura 3.2 mostra um exemplo da FPD de uma imagem com duas regiões.

Em [4] o autor utilizou 19 canais para segmentação, sendo 3 destes canais a Luminância (Y) e Crominância (Cr e Cb) e os outros 16 são o resultado da filtragem do canal de Y por 16 filtros de Gabor, [5] Procurar mais artigos sobre gabor. O 4 direções ($\theta = 0, \pi/4, \pi/2$ e $3\pi/4$) e 4 frequências centrais ($\omega = 1/2, 1/4, 1/8$ e $1/16$) foram utilizadas para definir os filtros. A escolha de apenas 4 direções se dá em função da simetria, uma vez que o sentido não importa, ou seja, $0 = \pi$, $\pi/4 = 5\pi/4$, $\pi/2 = 3\pi/2$ e $3\pi/4 = 7\pi/4$, sendo assim possível descrever um conjunto maior e mais rico de texturas usando o mínimo de filtros. O filtro de Gabor pode ser substituído por outro tipo de filtro 2D, entretanto foi escolhido pelo autor devido à sua avançada capacidade em distinguir texturas ([6], [7]). (ENCONTRAR REFERENCIAS SOBRE ISSO)

A utilização de vários canais torna a técnica adaptativa uma vez que os

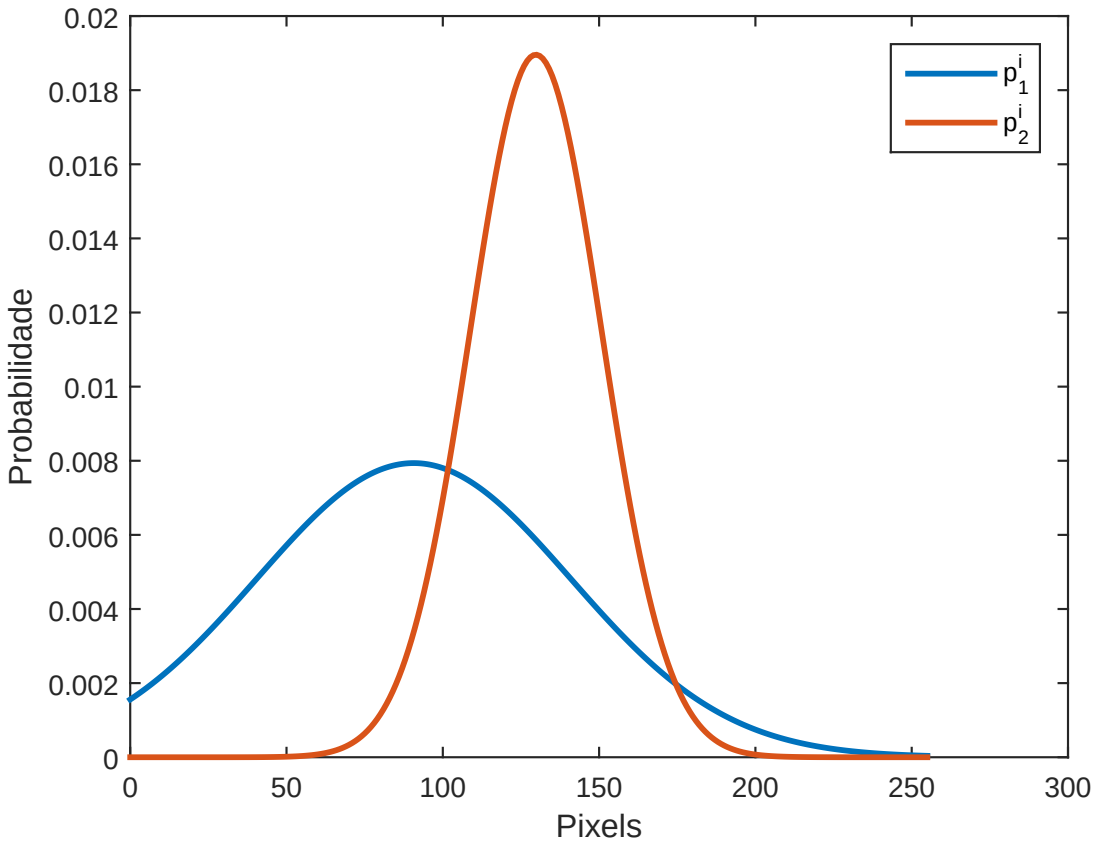

FIGURA 3.2: FDP de uma imagem com duas regiões

pesos (importância) de cada canal varia de acordo com a imagem e por isso a necessidade de usar um conjunto de filtros capaz de descrever um rico conjunto de texturas. A ideia é que cada um dos filtros realce uma parte diferente da imagem e com isso o canal resultante que ressaltar melhor a(s) parte(s) de interesse da imagem ganha um maior peso.

Os pesos mencionados anteriormente são calculados usando a equação 3.1 com base na probabilidade de um *pixel* x ser erroneamente assinalado à uma região (equação 3.2). Dessa forma tem-se um vetor com N_c (número total de canais) posições que representa o peso que cada canal terá na hora de calcular a probabilidade de um pixel pertencer a uma determinada região, equações 3.3 e 3.4, privilegiando o canal com maior peso, ou seja, os valores da FDP dos *pixels* deste canal é que irão de fato definir a qual região pertence o *pixel* em questão.

$$\forall\ i = 1, 2, 3, \ \ldots\ , N_c\ :\ \omega_i = \frac{(P_i^{-1})}{\sum_{k=1}^{N_c}(P_k^{-1})} \tag{3.1}$$

$$\forall\ k = 1, 2, \ \ldots\ , l\ : P_i = \frac{1}{l}\int_{-\infty}^{\infty} min(p_1^i(x), p_2^i(x), \ \ldots\ , p_k^i(x))\ dx \tag{3.2}$$

$$P^i_{1|2}(x) = \frac{p^i_1(F_i(x))}{p^i_1(F_i(x)) \ + \ p^i_2(F_i(x))} \tag{3.3}$$

$$P_{1|2}(x) \ := \ P_r(x \in l_1) = \sum_{i=1}^{N_c} \omega^i P^i_{1|2}(x) \tag{3.4}$$

Expandindo as equações 3.3 e 3.4 para l regiões ao invés de apenas duas têm-se a equação 3.5. O peso geodésico de um pixel da região a competindo somente com a região b é dado pela equação 3.6a, generalizando para mais de duas regiões uniformes obtêm-se a equação 3.6b. Este peso é utilizado para calcular a menor distância de um *pixel* x para uma marcação $l_k, k \in [1, N_l]$, dada pela equação 3.7 que utiliza a menor distância entre o *pixel* x e todos os *pixels* da marcação l_k calculada de acordo com a equação 3.8.

$$P_{a|b}(x) \ := \ P_r(x \in l_a) = \sum_{i=1}^{N_c} \omega^i \frac{p^i_a(F_i(x))}{p^i_a(F_i(x)) \ + \ p^i_b(F_i(x))} \tag{3.5}$$

$$\Omega_a = \Omega_{a|b} = 1 - P_{a|b}(x) \tag{3.6a}$$

$$\Omega_a = \sum_{b=1, \ a \neq b}^{l} \Omega_{a|b} \tag{3.6b}$$

$$d_k(x) = min_{s \in \Delta_c : label(s) = l_k} d(s,t) \tag{3.7}$$

$$d(x,t) := min_{C_{x,t}}(\Omega \dot{C}_{x,t}) \tag{3.8}$$

3.2.2 Segmentação de regiões não-uniformes

O algoritmo apresentado até este ponto trabalha recebe como entrada imagens com regiões distintas e uniformes, porém pode também ser expandido para trabalhar com imagens que possuam regiões não-uniformes. A mudança acontece apenas no cálculo dos pesos geodésicos Ω. As regiões da imagem são divididas em sub-regiões de tal forma que cada sub-região compete apenas com sub-regiões de regiões distintas. Definindo l^s_k como a componente (sub-região) s da região k pode-se definir o peso Ω^s_k pela equação. Um exemplo deste tipo de imagem pode ser visto na Figura 3.1b.

$$\Omega^s_k = \sum_{k \neq r} \sum_{l} \Omega_{l^s_k | l^l_r} \tag{3.9}$$

3.3 Metodologia

A técnica escolhida é bastante robusta sendo capaz de segmentar objetos em imagens complexas e não-uniformes, entretanto o tempo de execução do algoritmo é alto uma vez que é necessário calcular a distância de cada *pixel* da imagem para cada um dos *pixels* marcados afim de descobrir a menor distância e de acordo com o algoritmo descrito fazer a classificação dos pontos da imagem.

Após análise detalhada dos passos da técnica implementada percebeu-se que o tempo gasto para calcular todas as distâncias para todos os *pixels* representava boa parte do tempo total de execução do algoritmo e partir disto foi decidido que o foco do trabalho seria em melhorar esse tempo de cálculo das distâncias.

A solução encontrada foi reduzir o número de *pixels* marcados, ou seja, fazer uma re-amostragem desses *pixels* seguindo o princípio da técnica de utilizar a heurística provida pelo usuário nas marcações, em outras palavras, a re-amostragem tem que ser uniforme para que não haja as informações de cor e textura inicialmente selecionadas sejam preservadas. As seções seguintes irã detalhar as modificações realizadas e como a avaliação será feita.

3.3.1 Modificações realizadas

A implementação da técnica será mostrada com mais detalhes no Capítulo 4, porém para melhor compreensão da modificação alguns passos serão descritos nesta seção. O mais importante neste caso é que os *pixels* marcados são armazenados em uma matriz, chamada de matriz posição, que guarda tanto o valor quanto a posição dos *pixels* escolhidos.

A função *reamostragem.m* é responsável por esse passo, recebendo como entrada uma matriz contendo a sub-região a ser re-amostrada e a taxa de re-amostragem, e retorna a matriz re-amostrada de acordo com a taxa escolhida, mais detalhes sobre essa função serão apresentados no Capítulo 4. Essa nova matriz é então utilizada pelo algoritmo para calcular as probabilidades e posteriormente fazer a classificação dos *pixels*.

O parâmetro modificado para análise do tempo é a taxa de re-amostragem, o valores escolhidos foram: 100 %, 50 %, 10 % e 1 % dos *pixels* originalmente marcados. Os valores foram escolhidos para que seja possível analisar se a redução dos conjunto de *pixels* marcados está diretamente ligada ao tempo de execução do algoritmo, por exemplo, se o conjunto Ω for reduzido pela metade acarretaria na redução pela metade no tempo total. Os número 10

e 1 foram escolhidos por representarem extremos, ou seja, verificar o impacto da eliminação de quase todos os *pixes* de Ω no resultado final e determinar um *trade-off* sobre o ganho de tempo em relação à taxa de re-amostragem desses pontos.

3.3.2 Parâmetros avaliados

Para avaliar o resultado das diferentes taxas de re-amostragem foram utilizados dois parâmetros principais: tempo e o erro na classificação dos *pixels*, sendo que para avaliar o último foi considerado a imagem resultado sem re-amostragem como ponto de referência e a comparação é feita subtraindo as outras imagens resultado dessa imagem referência. Para análise do tempo foram utilizadas duas funções do *MatLab*, *tic/toc* e *etime* para garantir que a conformidade do tmepo medido. Esta avaliação de tempo ocorre de duas formas diferentes, primeiramente a função *tic/toc* é utilizada para medir o tempo de execução de cada passo do algoritmo, que foi divido em oito etapas listadas abaixo. Enquanto que a função *etime* foi usada para medir o tempo de execução total da função *segmenta.m*, que será detalhada no Capítulo 4.

Etapas do algoritmos

1. Posição dos pixels marcados
2. Cálculo dos Canais
3. Pesos dos canais
4. Pesos da distância geodésica
5. Re-amostragem
6. Tempo total de segmentação
7. Cálculo das distâncias
8. Cálculo das probabilidades

Capítulo 4

Implementação e Resultados

4.1 Implementação

A técnica descrita anteriormente foi implementada utilizando o *Matlab*, bem como em [4]. A implementação foi divida em partes para facilitar tanto o desenvolvimento quanto a manutenção do código fonte, estando este agrupado em 9 diferentes funções descritas abaixo com sua relação hierárquica mostrada na Figura 4.1. O fluxo do código e seus principais resultados utilizados ao longo do processamento são mostrados no diagrama da Figura 4.2, as cores utilizadas nesse diagrama correspondem às funções da Figura 4.1 que realizam as tarefas descritas.

- **segmenta.m:** Função que realiza a classificação dos *pixels* de acordo com a probabilidade.
- **getPixelsPosition.m:** Função que guarda em uma matriz a posição e os valores dos *pixels* marcados pelo usuário.
- **getChannels.m:** Função que calcula os canais.
  - **gaborFilter.m:** Função que implementa o filtro de Gabor.
  - **getPixelsDist.m:** Função que guarda em uma matriz a posição e os valores dos pixels marcados pelo usuário para todos os canais.
- **getChannelWeight:** Função que calcula o peso de cada canal seguindo a equação 3.1.
- **getGeodesicWeight:** Função que calcula o peso que irá ponderar a distância do *pixel* t para o *pixel* x, onde x pertence à uma região marcada.

- **resampleMatrix.m:** Função que faz a re-amostragem dos *pixels* marcados pelo usuário.
- **getMinDistance.m:** Função que calcula a menor distância de um *pixel* para uma determinada região marcada na imagem

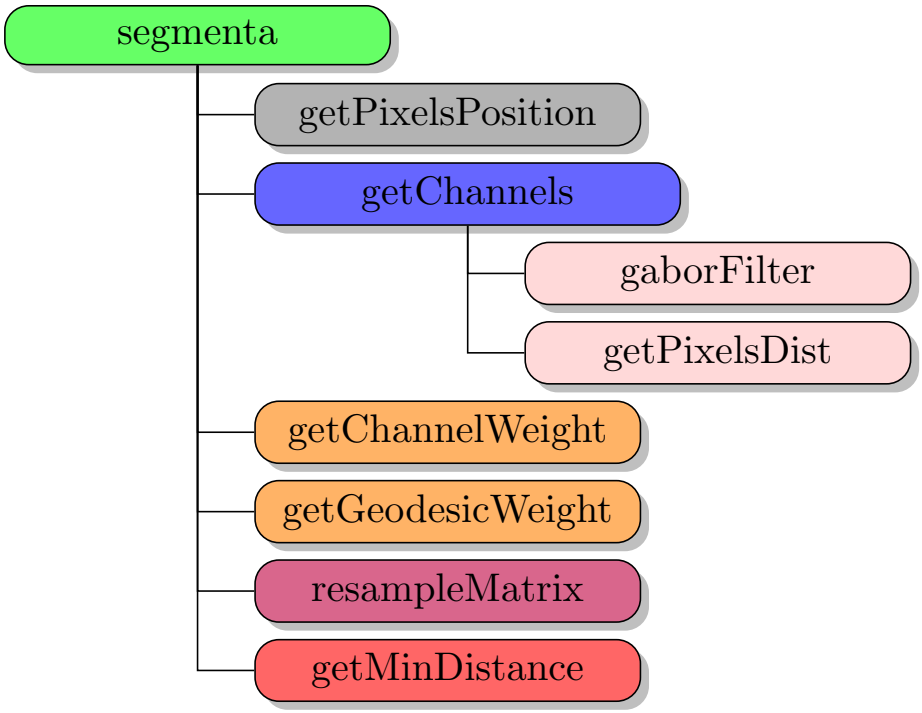

FIGURA 4.1: Hierarquia das funções criadas

A função *segmenta.m* recebe os parâmetros de entrada listados abaixo. É importante notar que o formato da imagem deve ser ".png" para que não haja compressão, ou seja, os valores dos *pixels* não sejam alterados após a imagem ser salva com as marcações desejadas, para cada região de interesse deve ser gerada uma imagem com as sub-regiões de interesse.

Para armazenar os valores dos pontos das regiões de interesse as imagens marcadas são comparadas ponto a ponto com a imagem original de tal forma que os valores que forem diferentes representam os *pixels* marcados, esses pontos então tem seu valor escrito em uma matriz guardando a mesma posição original. A partir desta matriz cada sub-região é armazenada em uma matriz diferente, estas são distinguidas verificando os *pixels* 8-conectados com seus vizinhos, assim cada imagem marcada é varrida apenas uma vez. Feito isto, é calculada a FDP das regiões de interesse, todo este processo é realizado pela função *getPixelsPosition.m*.

A função *getChannels.m* irá construir os filtros de gabor utilizando a função *gaborFilter.m*, uma vez construídos os filtros são utilizados no canal de luminância (Y) e sua saída (G_i) é utilizada na equação 4.1 para definição dos canais, os canais complementares são as crominâncias (Cb e Cr) da imagem. Após a criação do banco de canais, a função *getPixelsDist.m* é chamada para calcular a FDP dos *pixels* das regiões marcadas para cada um dos 19 canais utilizados.

Uma vez calculadas as FDP's das regiões de interesse de cada canal, é possível calcular o peso de cada canal em função dessas probabilidades conforme descrito anteriormente nas equações 3.1 e 3.2, este cálculo é feito pela função *getChannelsWeight.m* que utiliza a matriz contendo as FDP's das sub-regiões de todos os canais. O valor final da probabilidade de um *pixel* x pertencer a uma região é a soma das probabilidades de x pertencer a cada uma das sub-regiões existentes.

Cada *pixel* x no intervalo $[0, 255]$ tem um peso geodésico associado, descrito na equação 3.6b, que representa o complemento da probabilidade de x pertencer a uma determinada região ou sub-região. A função *getGeodesicWeight.m* utilizada as equações 3.5 e 3.6a para encontrar os valores de Ω de uma sub-região em comparação com outra. Para cada valor de pixel é criada uma matriz $\Omega(r, s)$, onde r é o número de regiões e s é o número de sub-regiões, a função *getGeodesicWeight.m* é chamada dentro de um laço para calcular o peso para cada sub-região, os pesos são calculados comparando uma a uma das sub-regiões, considerando o fato de que sub-regiões de uma mesma região não competem entre si, e a soma desses pesos é o peso geodésico final para o *pixel* em questão.

Por fim a classificação dos *pixels* propriamente ditos é feita diretamente na função *segmenta.m* que utiliza a função *getMinDistance.m* para encontrar a menor distância de um ponto entre todas as sub-regiões, essas distâncias, calculadas a partir da equação 3.8 são armazenas em uma matriz $D(r, s)$, conforme descrito anteriormente, e a partir dessa matriz é calculada um outra matriz $P(r, s)$ que armazena a probabilidade, segundo a equação 4.2, do *pixel* atual pertencer a cada uma das sub-regiões e então o menor valor da matriz P representa a qual sub-região pertence o *pixel* analisado:

$$\forall (x,y) \in \Omega \; : F_i(x,y) = \frac{1}{N^2} \int \int_{\Omega_{x,y}} \tanh\left(\alpha \frac{G_i(u,v)}{\sigma(G_i)}\right) dudv, \; onde \quad \alpha = 0.25 \, e \, N = 5 \tag{4.1}$$

- A taxa referente a quantos % dos pixels marcados serão usados, no intervalo [0,1]
- A escala de cor a ser utilizada, podendo esta ser: cinza, R, G ou B
- A imagem original
- As imagens marcadas.

$$Pr(t \in l_i) = \frac{d_i(t)^{-1}}{\sum_{j \in [1,N_l]} d_j(t)^{-1}} \tag{4.2}$$

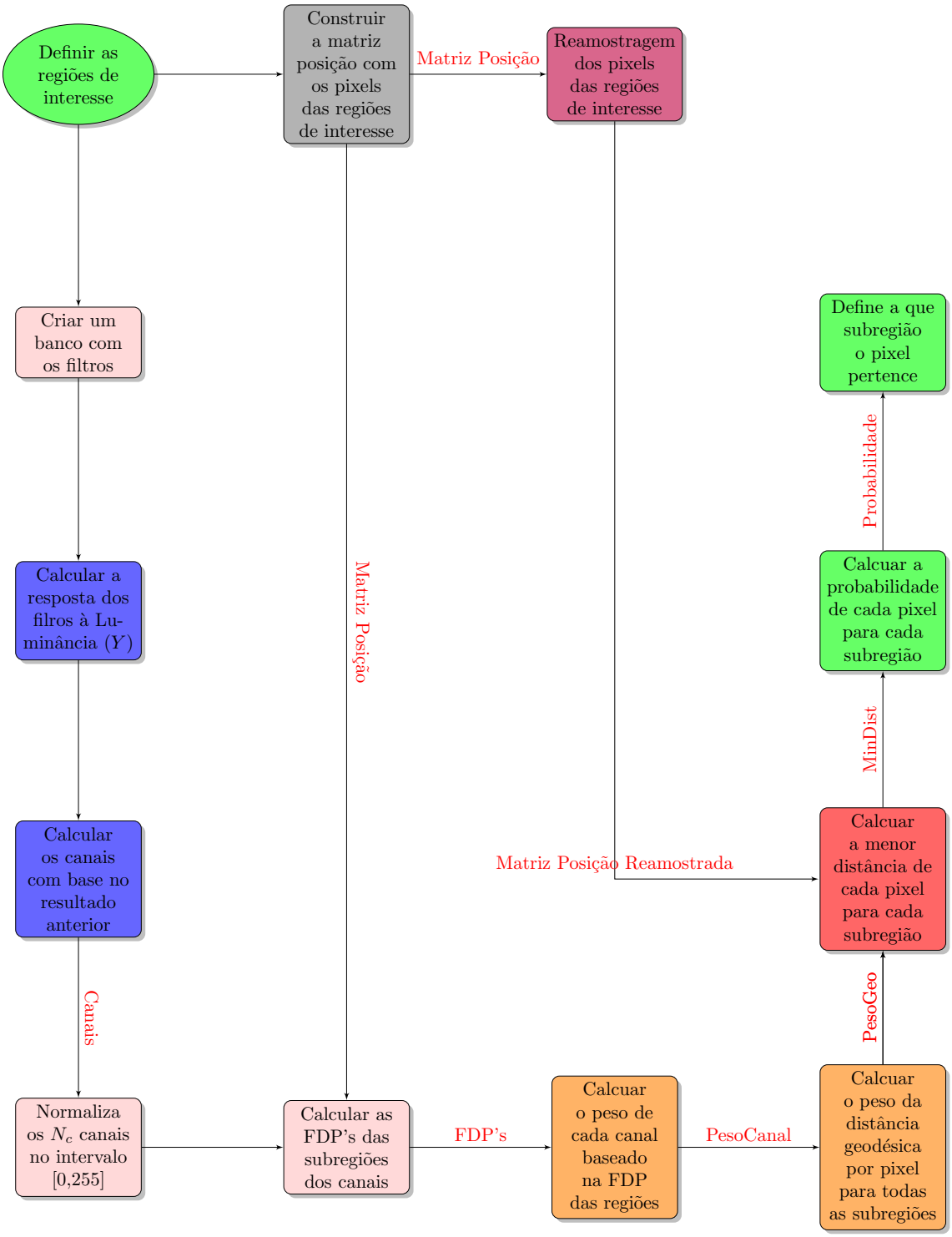

FIGURA 4.2: Fluxo do algoritmo

Capítulo 5

Considerações finais e Trabalhos futuros

Referências Bibliográficas

[1] Linda Shapiro and George Stockman. Computer Vision. 9:609, 2000.

[2] Rafael C Gonzalez, Richard E Woods, and Barry R Masters. Digital image processing, third edition. *Journal of biomedical optics*, 14(2):029901.

[3] S.S. HAYKIN and B. VAN VEEN. *Sinais e sistemas*. Bookman, 2001.

[4] Alexis Protiere and Guillermo Sapiro. Interactive image segmentation via adaptive weighted distances. *IEEE Transactions on Image Processing*, 16(4):1046–1057, 2007.

[5] B Manjunath and W Ma. Texture features for browsing and retrivieval of image data. *\mbox{IEEE} Trans. on Pattern Analysis and Machine Intelligence*, 8(18):837–842, 1996.

[6] M Sivalingamaiah and B D Venakramana Reddy. Texture Segmentation Using Multichannel Gabor Filtering. 2(6):22–26, 2012.

[7] Tadayoshi Shioyama, Haiyuan Wu, and Shigetomo Mitani. Segmentation and object detection with Gabor filters and cumulative histograms. *Proceedings - International Conference on Image Analysis and Processing, ICIAP 1999*, (1):412–417, 1999.