UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE TECNOLOGIA

FACULDADE DE ENGENHARIA DA COMPUTAÇÃO E TELECOMUNICAÇÕES

TITULO A DEFINIR

Autor: Danilo Henrique Costa Souza

Orientador: Prof. Dr. Ronaldo de Freitas Zampolo

Belém/PA, 28 de agosto de 2015.

UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE TECNOLOGIA

FACULDADE DE ENGENHARIA DA COMPUTAÇÃO E TELECOMUNICAÇÕES

TITULO A DEFINIR

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, 28 de agosto de 2015.

TITULO A DEFINIR

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

Redes de sensores ópticos baseados em FBG

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 [1] 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 [1] 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 [1], 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 [1] 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, [2] 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 ([3], [4]). (ENCONTRAR REFERENCIAS SOBRE ISSO)

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

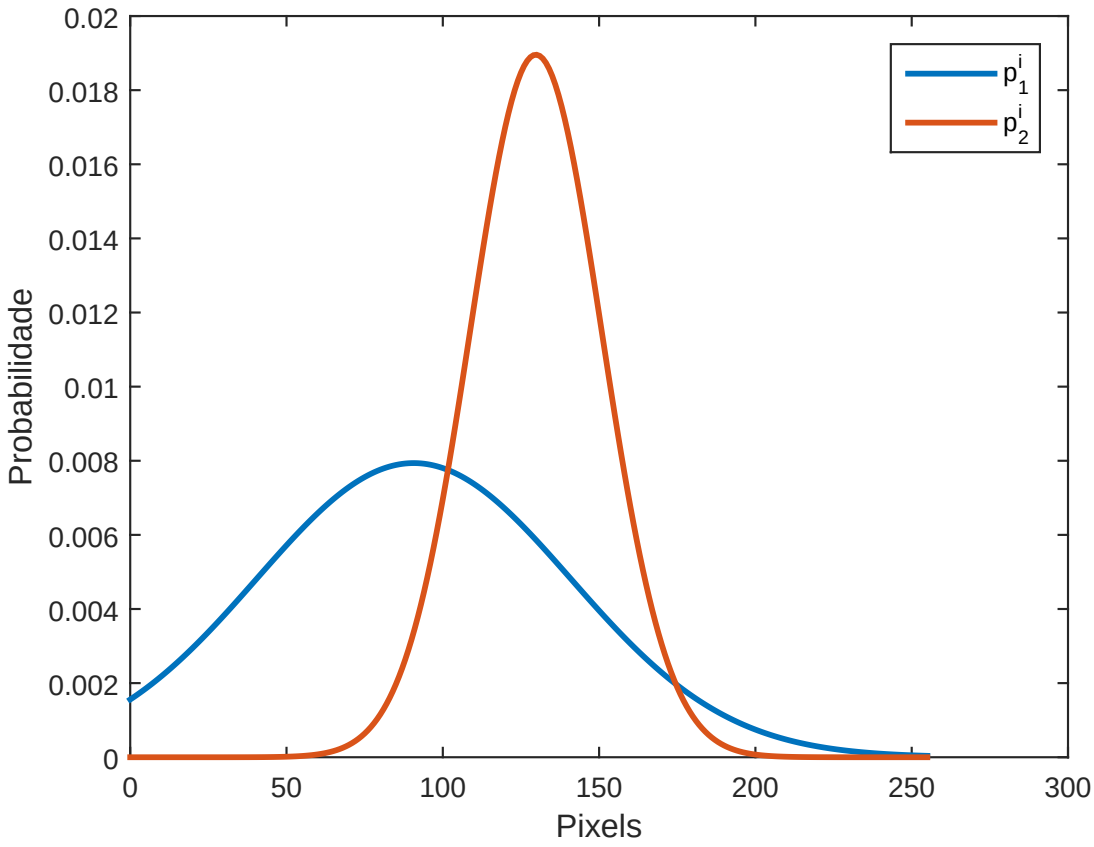

FIGURA 3.2: FDP de uma imagem com duas regiões

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

Conforme descrito no capítulo anterior a técnica escolhida é bastante robusta sendo capaz de segmentar objetos em imagens complexas, entretanto o tempo de execução do algoritmo é alto e esta diretamente associado ao número de *pixels* marcados

Capítulo 4

Implementação e Resultados

4.1 Implementação

A técnica descrita anteriormente foi implementada utilizando o *Matlab*, bem como em [1]. 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

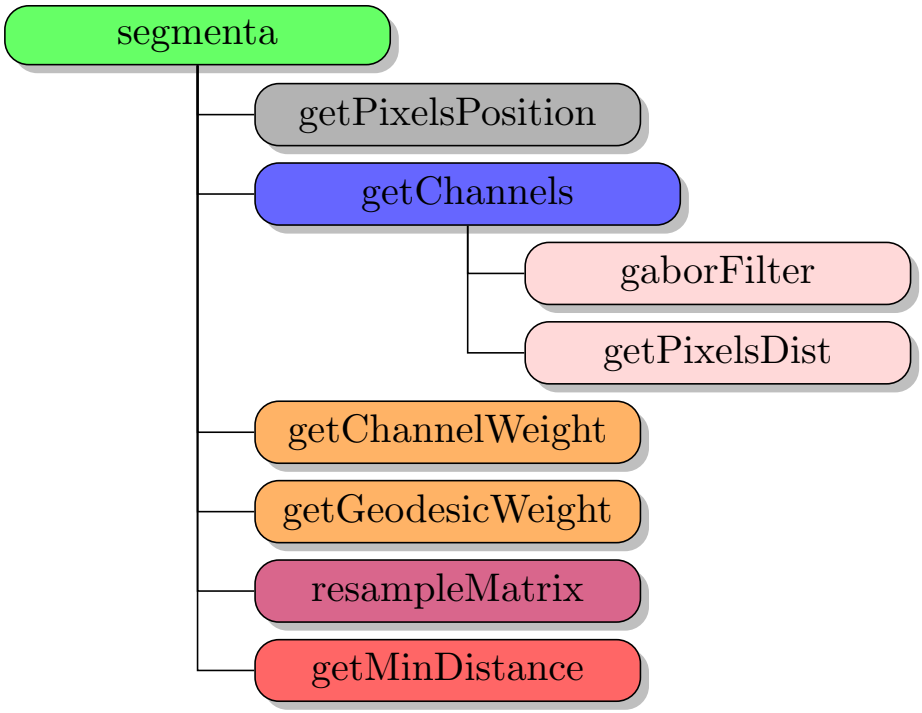

FIGURA 4.1: Hierarquia das funções criadas

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.

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

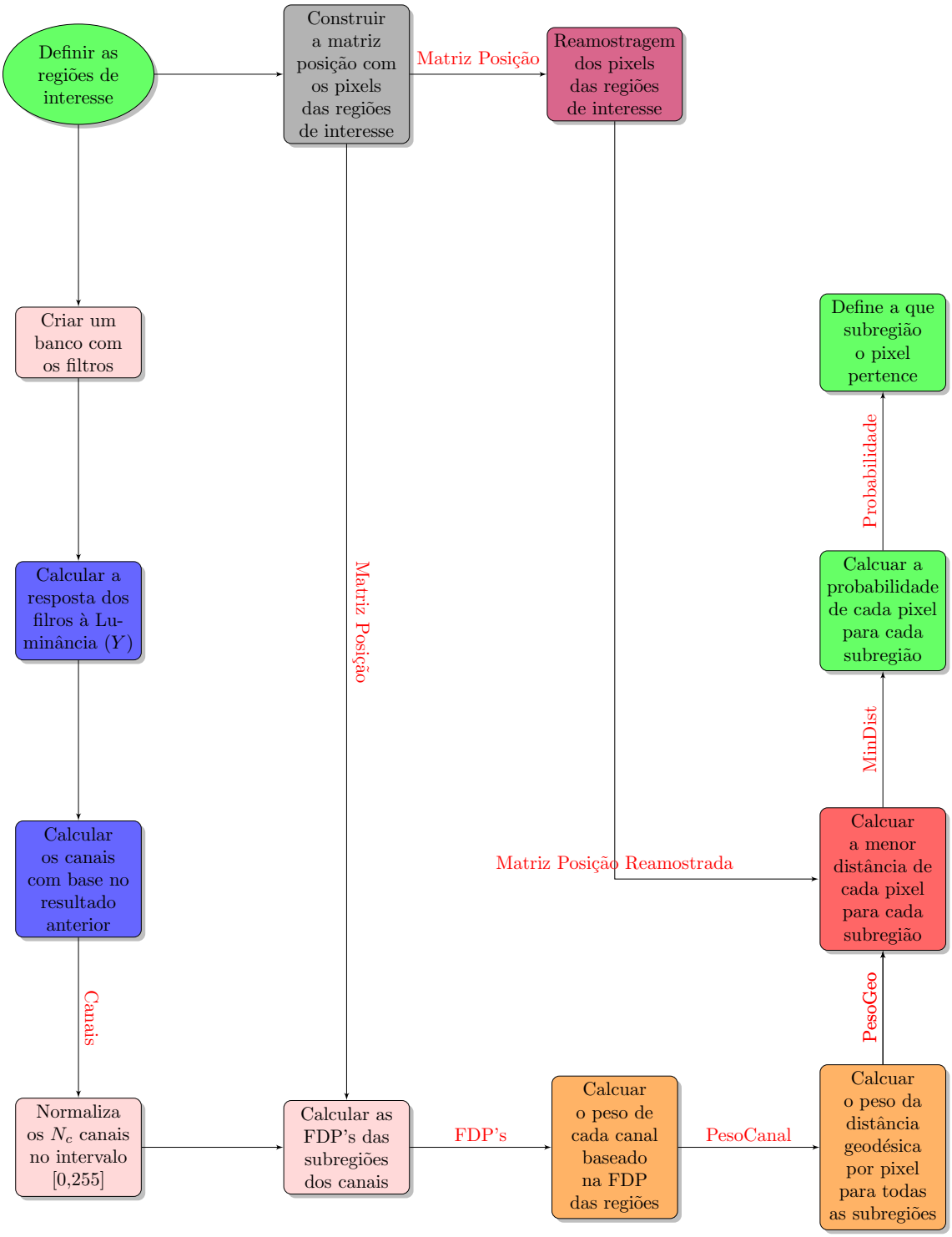

FIGURA 4.2: Fluxo do algoritmo

Capítulo 5

Considerações finais e Trabalhos futuros

Referências Bibliográficas

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

[2] 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.

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

[4] 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.