UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Segmentação de Imagens Utilizando Superpixels e Redes Complexas

Autora: Victória Maria Gomes Velame

Orientador: Prof. Dr. Adilson Gonzaga

São Carlos

2016

Victória Maria Gomes Velame

Segmentação de Imagens Utilizando Superpixels e Redes Complexas

Trabalho de Conclusão de Curso apresentado
à Escola de Engenharia de São Carlos, da
Universidade de São Paulo

Curso de Engenharia Elétrica

ORIENTADOR: Prof. Dr. Adilson Gonzaga

São Carlos
2016

Velame, Victória Maria Gomes
V432s Segmentação de Imagens Utilizando Superpixels e Redes Complexas / Victória Maria Gomes Velame; orientador Adilson Gonzaga. São Carlos, 2016.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2016.

1. Segmentação de imagens . 2. Superpixels. 3. Detecção de comunidades. 4. Redes complexas. I. Título.

FOLHA DE APROVAÇÃO

Nome: Victória Maria Gomes Velame

Título: "Segmentação de imagens utilizando Superpixels e Redes Complexas"

Trabalho de Conclusão de Curso defendido e aprovado em 25 / 11 / 2016,

com NOTA 10,0 (DEZ, ZERO), pela Comissão Julgadora:

Prof. Associado Adilson Gonzaga - Orientador - SEL/EESC/USP

Dra. Carolina Toledo Ferraz - SEL/EESC/USP

Mestre Raissa Tavares Vieira - Doutoranda-SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado José Carlos de Melo Vieira Júnior

Agradecimentos

Em primeiro lugar agradeço à Deus, quem me deu a oportunidade e as condições necessárias para desenvolver este trabalho.

Ao meu orientador Prof. Dr. Adilson Gonzaga, por aceitar-me como orientanda, por tudo o que me ensinou e pela amizade.

Ao meus pais, pelo amor e por todo o apoio que eles sempre me deram.

À João Ferreira, pelo amor, amizade, compreensão, incentivo e apoio incondicional.

À Ana Paula Fonsêca e Milla Dantas pela amizade e incentivo.

E a todos que direta ou indiretamente fizeram parte da minha formação.

Victória Maria Gomes Velame.

"O sucesso é ir de fracasso em fracasso sem perder entusiasmo."

Winston Churchill

Resumo

A segmentação de imagens é uma das tarefas mais importantes na análise de imagens em visão computacional. Muitas técnicas para a segmentação imagens já foram propostas. Destas, as baseadas em grafos têm se destacado. Neste trabalho, é proposto e avaliado um método de segmentação de imagens que combina extração de *superpixels* e detecção de comunidades de redes complexas. Neste método, os *superpixels* substituem os *pixels* na geração da rede complexa, com a finalidade de reduzir o número de nós da rede, permitindo o uso dos algoritmos de detecção de comunidades em imagens de grandes dimensões. Para a extração de *superpixels* utilizou-se o algoritmo *Simple Linear Iterative Clustering* e para detecção de comunidade, utilizou-se os algoritmos *Fast Greedy* e *Label Propagation*. Os experimentos realizados analisam os parâmetros envolvidos na metodologia proposta, principalmente os relacionados à criação da rede complexa, onde uma nova função peso foi proposta. Estes experimentos foram realizados com o uso de um banco de imagens com *ground-truth*. Os resultados dos experimentos e a comparação com o método de segmentação *Efficient Graph-Based Image Segmentation* mostraram que o método de segmentação proposto apresenta bons resultados de qualidade e número de segmentos, porém alto tempo de processamento.

Palavras-Chave: Segmentação de imagens; *Superpixels*; Detecção de comunidades; Redes complexas.

Abstract

Image segmentation is one of the most important tasks on the image analysis on computational vision. Many techniques for image segmentation were already proposed. From these, the ones based of graphs have been more succesful. In this work, it is proposed and evaluated a method of image segmentation that combines superpixels extraction and community detection of complex networks. In this method, superpixels replace the pixels on the generation of the complex network, aiming to reduce the number of network nodes, enabling the use of community detection algorithms on large scale images. For the extraction of superpixels, the Simple Linear Iterative Clustering algorithm was used and for the community detection, the Fast Greedy and the Label Propagation algorithms were used. The experiments analyze the the parameters for the proposed methodology, mainly the ones related to the creation of the complex network, on which a new weight function was proposed. These experiments were made with a database with ground-thruth. The results and the comparison with the Efficient Graph-Based Image Segmentation method showed that the proposed method for segmentation provides good quality and number of segments results, albeit a high processing time.

Keywords: Image Segmentation, Superpixels, Community Detection; Complex Networks.

Lista de Figuras

Lista de Tabelas

Siglas

BSDS300	*Berkeley Segmentation Dataset* 300
CIE	*Commission Internationale de l'Éclairage* (Comissão Internacional de Iluminação)
EGB	*Efficient Graph-Based Image Segmentation*
EM	*Expectation-maximization* (Maximização de expectativa)
FCM	*Fuzzy c-means*
Q	Modularidade
SEEDS	*Superpixels Extracted via Energy-Driven Sampling*
SLIC	*Simple Linear Iterative Clustering*
SUTP	*Speeded-up Turbo Pixels*

Sumário

Capítulo 1

Introdução

Um dos sentidos mais importantes para o ser humano é o da visão. Com ele, é possível analisar uma cena e extrair grande quantidade de informação dela. Há diversas tarefas que dependem da visão humana e, para otimizar algumas dessas tarefas, surgiu na ciência a área de visão computacional. Esta área se dedica a interpretar e extrair informações de imagens, na tentativa de emular as funcionalidades da visão humana. Segundo Ballard e Brown (1982), visão computacional é a construção de descrições explícitas e claras dos objetos em uma imagem. Embora seja natural para os seres humanos, a interpretação de imagens é um grande desafio para a ciência e, até hoje, nenhum sistema de visão automatizada apresenta desempenho similar ao do ser humano.

O interesse em métodos de processamento de imagens decorre de duas principais aplicações: melhoria de informação visual para a interpretação humana e o processamento de dados de cenas para a percepção automática através da máquina (GONZALEZ; WOODS, 2000). Uma das primeiras aplicações de técnicas de processamento de imagem foi o melhoramento de imagens digitalizadas para jornais. Em 1964, técnicas de computação para corrigir vários tipos de distorção foram utilizadas para o melhoramento de imagens da Lua (GONZALEZ; WOODS, 2000). Atualmente, verifica-se a utilização cada vez maior dos algoritmos de visão computacional nas tarefas de extração de informações de imagens para solucionar problemas práticos (UMBAUGH, 2016).

Visão computacional apresenta diversas aplicações em múltiplas áreas do conhecimento. Algumas destas são: análise de imagens médicas, automação industrial, robótica, sensoriamento remoto, reconhecimento de características, recuperação de imagem baseada em conteúdo, observações astronômicas, rastreamento de segurança, monitoramento ambiental, entre outros (JAIN, 1989; PARKER, 2010; COSTA; CESAR JR, 2000). A principal aplica-

bilidade de visão computacional decorre de processos que geram uma grande quantidade de imagens, de modo a tornar a automatização do processo de análise das imagens e extração de características destas indispensável.

O processo de visão geralmente é composto pelas seguintes etapas: pré-processamento, segmentação, extração de características, classificação e inferências. Na figura 1.1 há a representação do sistema de visão computacional segundo a interpretação de Jain (1989).

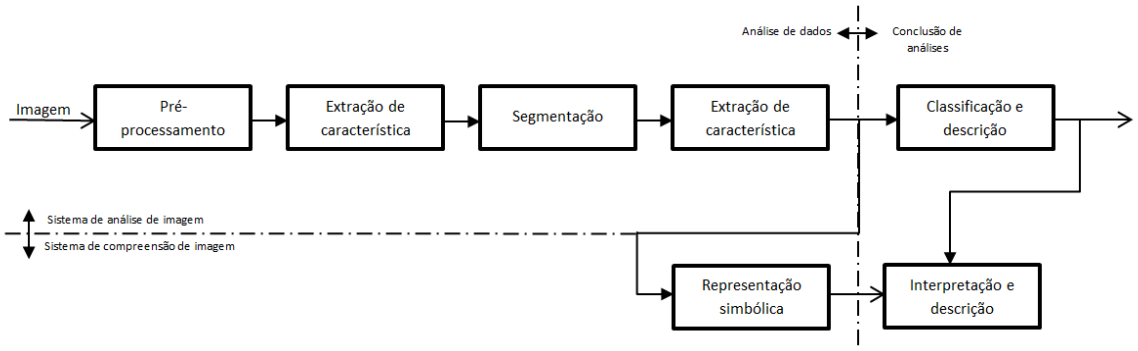


Figura 1.1: Representação dos processos que compõem o sistema de visão computacional. Adaptada de Jain (1989).

A segmentação, uma das etapas do sistema de visão computacional, consiste em identificar regiões de uma imagem, estabelecendo subdivisões em sua unidade básica (*pixel*). O nível de detalhe em que a subdivisão é realizada depende do problema a ser resolvido. Em muitas aplicações, a precisão da segmentação determina o sucesso ou fracasso final dos procedimentos de análise computadorizada (GONZALEZ; WOODS, 2000). A segmentação é de extrema importância para outros processos, principalmente a detecção e reconhecimento de padrões. Ela pode ser utilizada, por exemplo, em diagnósticos médicos para localizar patologias, em sistemas de controle de tráfego, descoberta automática de fissuras em materiais, entre outras aplicações.

Existem diversos métodos e algoritmos para segmentar uma imagem, que podem utilizar cor, forma ou textura da imagem. Segundo Gonzalez e Woods (2000), os algoritmos para a segmentação são usualmente baseados em duas propriedades da imagem: descontinuidade e similaridade. A segmentação baseada em descontinuidade gera partições quando ocorrem alterações bruscas dos níveis da função imagem. Já a segmentação baseada na similaridade particiona regiões com características semelhantes. Dentre os principais algoritmos para segmentação de imagem estão os métodos de limiarização ou histogramas, de crescimento de regiões e de grafos. Neste trabalho serão explorados dois métodos de segmentação: extração de *superpixel* e segmentação por reconhecimento de comunidade de redes complexas.

O método de *superpixels* surgiu como forma de reduzir a informação presente em uma imagem, sendo um *superpixel* constituído por um conjunto de *pixels* com características semelhantes. Deste modo, os *superpixels* devem formar um agrupamento de *pixels* resultando em uma segmentação da imagem em regiões. Com este método, pode-se obter um aumento substancial na velocidade de processamento da imagem, desde que o número de *superpixels* não seja muito grande, de 25 a 2500 em média (NEUBERT; PROTZEL, 2012). Existem várias formas de gerar *superpixels*, cujo algoritmo original foi introduzido por Ren e Malik (2003) e teve como base o algoritmo *Normalized Cut* de Shi e Malik (2000).

O método de superpixel é utilizado, em geral, como uma pré segmentação capaz de capturar redundância na imagem e reduzir significativamente a complexidade das tarefas subsequentes de processamento de imagem, por conseguinte, reduzindo o custo computacional total do processo (SHI; MALIK, 2000). Algumas de suas aplicações são: estimativa de profundidade (HOIEM; EFROS; HEBERT, 2005), de segmentação de imagens (CUADROS, 2013; BOTELHO, 2014; TORRES, 2012; SARATH, 2014), esqueletização (LEVINSHTEIN; DICKINSON; SMINCHISESCU, 2009), localização de objeto (FULKERSON et al., 2009), *tracking* (WANG et al., 2011) e biomédicas (AKBAR et al., 2015). Esse método é muito utilizado quando pretende-se segmentar a imagem utilizando algoritmos de grafos ou redes complexas. Nestes últimos, ao invés de cada nó ser representado por um *pixel*, utiliza-se os superpixels como pré-processamento para reduzir a cardinalidade do grafo ou da rede.

Os algoritmos de segmentação baseados em grafos têm-se tornado populares no início deste século (SHI; MALIK, 2000). Estes métodos geralmente criam um grafo G onde cada nó representa um *pixel* da imagem e os pesos das arestas são computados de acordo com uma função de similaridade. Algumas limitações destes métodos são o alto tempo de processamento e consumo excessivo de memória. Assim, sua limitação é o tamanho da imagem, pois uma imagem de tamanho $N \times N$ é mapeada em um grafo com N^2 vértices. Com as recentes pesquisas na teoria de redes complexas, as técnicas de reconhecimento de padrões baseadas em grafos melhoraram consideravelmente (OLIVEIRA; ZHAO, 2008).

A detecção de comunidades tem como base a teoria de redes complexas, o uso dessa técnica em segmentações de imagens vem produzindo particionamentos acurados e com custo computacional satisfatório (NEWMAN; GIRVAN, 2004). Essa técnica identifica comunidades, que são grupos nos quais os vértices são mais intensamente conectados entre si do que com o restante da rede. Além disso, a técnica de detecção de comunidades fornece partições mais precisas que os métodos tradicionais baseados em grafos, como *Graph Partitioning*,

Hierarchical Clustering, *Partitional Clustering* e *Spectral Clustering* (FORTUNATO, 2010). Uma grande vantagem da detecção de comunidades é que o número de comunidades (objetos segmentados) é definido pelo algoritmo, não sendo uma constante fixa, como ocorre na maioria dos algoritmos de segmentação (RAGHAVAN; ALBERT; KUMARA, 2007).

1.1 Justificativa

A principal motivação deste trabalho é a proposição de técnicas de segmentação de imagens automáticas capazes de se aproximar das segmentações manuais, de modo a auxiliar tarefas que envolvem a interpretação de grande quantidade de imagens.

Além disso, outra motivação do trabalho é a busca por técnicas de segmentações de imagem que mantenham o comprometimento entre acurácia e custo computacional, sendo capazes de segmentar imagens de diferentes contextos e tamanhos automaticamente e definir de modo automático a melhor quantidade de segmentos para cada imagem. Por fim, há a motivação de se estudar e analisar técnicas de *superpixels* em conjunto com algoritmos de detecção de comunidades, e cada uma delas individualmente.

1.2 Objetivos

O objetivo principal deste trabalho é estudar, avaliar e implementar métodos de segmentação de imagens baseados em detecção de comunidades e *superpixels*. Para isso, é realizado um pré-processamento com o algoritmo de extração de *superpixels*, o qual servirá para reduzir os nós da rede gerada, reduzindo o custo computacional total do processo.

Além disso, pretende-se, neste trabalho, estudar e avaliar quais os melhores algoritmos de extração de *superpixels* e de detecção de comunidades para esta aplicação, avaliando quais os parâmetros envolvidos no processo, principalmente os relacionados à geração da rede complexa. Pretende-se, também analisar a segmentação gerada somente pelo algoritmo de extração de *superpixels*, avaliando a influência em qualidade e custo computacional dos parâmetros envolvidos em sua técnica.

1.3 Organização do trabalho

Os capítulos seguintes estão organizados na seguinte disposição:

- No Capítulo 2, são descritos conceitos relacionados à segmentação de imagens digitais. Neste capítulo, é apresentada a revisão bibliográfica de algumas técnicas básicas de segmentação de imagens.

- No Capítulo 3, são descritos conceitos relacionados aos espaços e modelos de cores. O foco deste capítulo são os modelos CIELAB e RGB.

- No Capítulo 4, é apresentada a revisão bibliográfica de algoritmos de extração de *superpixels*.

- No Capítulo 5, são abordados conceitos relacionados à teoria das redes complexas e revisão bibliográfica de algoritmos de detecção de comunidades.

- No Capítulo 6, é apresentada a metodologia proposta para segmentação de imagens, baseada em *superpixels* e detecção de comunidades. Neste capítulo, são descritas as etapas do método de segmentação proposto e a metodologia utilizada para avaliar os resultados.

- No Capítulo 7, são apresentados os experimentos realizados, seus resultados e a discussão dos mesmos.

- Finalmente, no Capítulo 8, são apresentadas as considerações finais, as limitações, contribuições e sugestões para trabalhos futuros.

Capítulo 2

Segmentação de imagens

A segmentação de imagens é uma etapa de visão computacional que consiste em identificar regiões (objetos ou partes) de uma imagem, estabelecendo subdivisões em sua unidade básica (*pixel*). O nível de detalhe em que a subdivisão é realizada depende do problema a ser resolvido. Ou seja, a segmentação deve parar quando os objetos de interesse tiverem sido isolados (GONZALEZ; WOODS, 2000). Em muitas aplicações, a precisão da segmentação determina o sucesso ou fracasso final dos procedimentos de análise computadorizada (GONZALEZ; WOODS, 2000; CUADROS, 2013).

Geralmente, a segmentação é o primeiro passo na análise de imagens. Entretanto, antes de segmentar uma imagem, normalmente é realizado um pré-processamento para reduzir o ruído e aumentar a qualidade das imagens. O objetivo da segmentação é reduzir ao máximo as informações para que as etapas de reconhecimento e/ou caracterização obtenham maior taxa de sucesso. Desse modo, a segmentação é de extrema importância para outros processos, principalmente na detecção e reconhecimento de padrões. Ela pode ser utilizada, por exemplo, em diagnósticos médicos para localizar patologias, em sistemas de controle de tráfego, entre outras aplicações.

Uma segmentação deve ser tal que obedeça as seguintes condições: todos os *pixels* da imagem devem pertencer a uma região, as regiões devem ser conectadas e todos os *pixels* de uma dada região devem ser considerados similares. Haralick e Shapiro (1985) estabeleceram os seguintes critérios qualitativos para uma boa segmentação.

- Regiões segmentadas de uma imagem devem ser uniformes e homogêneas com respeito a características como tons de cinza ou textura.

- Regiões interiores devem ser simples e sem muitos buracos pequenos.

- Regiões adjacentes a uma segmentação devem ter valores significativamente diferentes com respeito às características nas quais elas são uniformes.
- Fronteiras de cada segmento devem ser simples, não irregulares e devem ser acuradas espacialmente.

Existem algumas dificuldades inerentes ao processo de segmentação de imagens. As principais são: controle da luminosidade, bordas de regiões irregulares e imprecisas, controle da qualidade da imagem, presença de ruído e escolha do algoritmo mais adequação à aplicação desejada. Em ambientes externos, a dificuldade é maior devido às variáveis externas como clima, iluminação e ruído. Em muitos casos, opta-se por ambientes controlados para facilitar a interpretação de imagens, promovendo maior sucesso nas aplicações.

A escolha do método de segmentação é uma tarefa importante na obtenção de um algoritmo apropriado para o problema a ser resolvido. Existem diversos métodos e algoritmos para segmentar uma imagem, que podem utilizar cor, forma ou textura da imagem. Segundo Gonzalez e Woods (2000), os algoritmos para a segmentação são usualmente baseados em duas propriedades da imagem: descontinuidade e similaridade. A segmentação baseada em descontinuidade gera partições quando ocorrem alterações bruscas dos níveis da função imagem. Já a segmentação baseada na similaridade particiona regiões com características semelhantes. Dentre os principais métodos para segmentação de imagem estão os de limiarização (ou *thresholding*), de crescimento de regiões, detecção de fronteiras e de grafos. Esses método apresentam limitações, de modo que se faz necessário um conjunto destes métodos em busca um algoritmo de maior acurácia e menor custo computacional. Neste capítulo é apresentada uma revisão bibliográfica de alguns métodos de segmentação com aplicações e o estado da arte do mesmo.

2.1 Limiarização ou *Thresholding*

O processo de limiarização se baseia na análise do histograma da imagem, que é a representação da imagem no espaço de cores. Portanto, a limiarização consiste em segmentar uma imagem em N níveis baseados em limiares (*thresholds*) encontrados com cálculos estatísticos a partir do histograma. A limiarização foi desenvolvida, inicialmente, para segmentar imagens em escala cinza (monocromáticas), porém, posteriormente, foi generalizada para imagens multicromáticas. Normalmente, a técnica de limiarização é utilizada para segmentar a imagem em apenas duas regiões, preto e branco. Quando mais regiões são incorporadas o

cálculo de limiar fica bastante susceptível a erros, assim sendo, este último caso só é utilizado em ambientes controlados nos quais há contraste significativo entre os objetos. Segundo Gonzalez e Woods (2000), limiarização multinível é geralmente menos confiável que a de limiar único.

Na figura 2.1 há um exemplo de histograma limiarizado com dois *thresholds* (T_1 e T_2), este histograma ilustra o caso de uma imagem com alto contraste e com dois vales, para estes casos, o uso de mais de um limiar é benéfico. Porém, como a maioria dos histogramas não tem vales, normalmente é utilizado apenas um limiar.

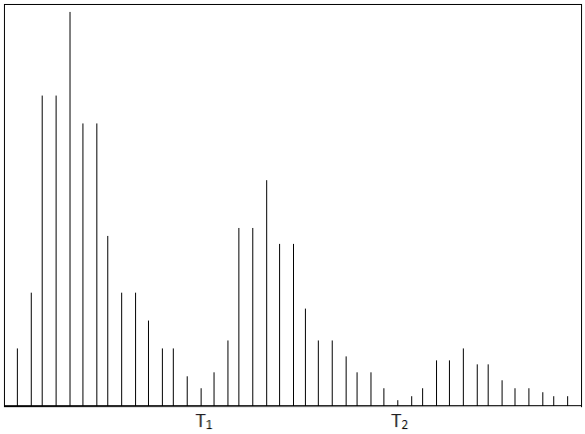


Figura 2.1: Histograma particionado por limiares múltiplos.

Matematicamente, para uma imagem de entrada $f(x,y)$, a operação de limiarização para N limiares pode ser descrita como:

$$\begin{aligned}
&R_1, \quad caso \quad f(x,y) \leq T_1 \\
&R_2, \quad caso \quad T_1 < f(x,y) \leq T_2 \\
&\vdots \\
&R_N, \quad caso \quad T_{N-1} < f(x,y) \leq T_N \\
&R_{N+1}, \quad caso \quad f(x,y) > T_N
\end{aligned}$$

Para apenas um limiar, a saída da técnica de segmentação por limiarização é uma imagem em dois tons $g(x,y)$, geralmente, preto e branco. O equacionamento neste caso é definido por:

$$g(x,y) = \begin{cases} 1, & \text{caso } f(x,y) \leq T_1 \\ 0, & \text{caso } f(x,y) > T_1 \end{cases} \tag{2.1}$$

A técnica de limiarização não funciona bem em imagens com iluminação não uniforme e com baixo contraste entre as regiões. Ela é muito utilizada quando pretende-se segmentar objetos com tons bastantes distintos do fundo. Na figura 2.2 há o exemplo de uma imagem na qual foi realizada a limiarização pelo método de Otsu (1975) com e sem iluminação uniforme. Como é possível perceber, essa imagem apresenta alto contraste dos objetos com relação ao fundo, logo a limiarização funciona bem para este caso. Porém, com a inserção de iluminação não uniforme, percebe-se que há perda de objetos e diminuição da eficácia deste método.

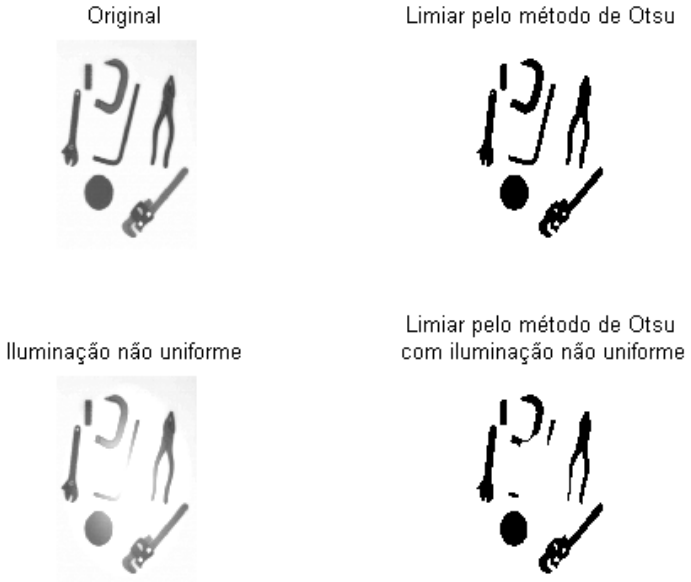


Figura 2.2: Imagem limiarizada pelo método de Otsu (1975) com e sem efeito de iluminação não uniforme.

Para a determinação do limiar existem processos manuais e automáticos. Os manuais se baseiam na procura de vales no histograma e os automáticos, por sua vez, utilizam uma função matemática para determinar o limiar. Segundo Gonzalez e Woods (2000) essa operação pode ser definida como: $T = T[x,y,p(x,y),f(x,y)]$, onde $f(x,y)$ é o nível de cinza do ponto (x,y) e $p(x,y)$ denota alguma propriedade local deste ponto. De modo que, se o limiar não depender de $p(x,y)$, ele é global, e, caso dependa, ele é local.

Dentre os métodos automáticos o mais comum é o método de Otsu (1975) ou limiarização ótima. Esse é um método global que trata o histograma como uma função densidade de

probabilidade e procura um limiar capaz de maximizar a variância das classes segmentadas. Porém, existem vários outros métodos que são empregados a depender das características da aplicação. Como exemplos, temos os métodos de limiarização global simples de Gonzalez e Woods (2000), limiarização baseada em características de fronteira, fuzzy, entropia, média, entre outros (PARKER, 2010). Na figura 2.3 há exemplos de alguns dos métodos utilizados para obter o *threshold* com a respectiva imagem binarizada (limiarizada) e seu valor de *threshold*.

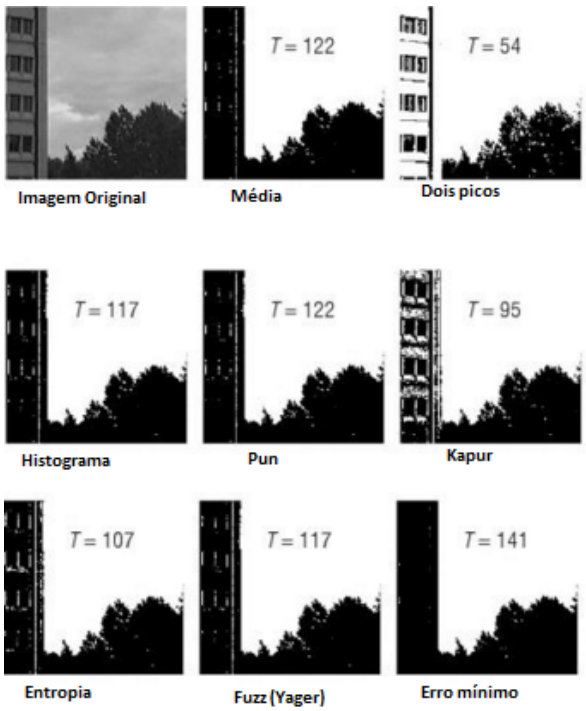


Figura 2.3: Limiarização de imagem em escala cinza com os respectivos métodos para obtenção do *threshold*. Adaptada de Parker (2010).

2.1.1 Trabalhos relacionados

Com o intuito de viabilizar o uso da técnica de limiarização para aplicações em tempo real, LOPES (2003), na sua dissertação de mestrado, desenvolveu um modelo de limiarização de imagens baseado na percepção de visão humana. Seu modelo foi resultante do treinamento de uma Rede de Funções de Base Radial e utilizou dados de limiarização manual obtida em um experimento psicofísico. Como resultados ele demonstrou que seu método pode ser usado como referência para avaliação de métodos automáticos de limiarização, devido a sua capacidade de quantizar a qualidade de um método de limiarização.

Aboud Neta, Dutra e Erthal (2008) elaboraram uma técnica de limiarização automática multiníveis baseada em histogramas, que consiste em encontrar cada vale entre os picos do

histograma. O algoritmo implementado para a detecção de picos e vales é baseado no cálculo das áreas de regiões onde há transição de sinal dos valores do histograma. Devido a grande quantidade de ruído, esse algoritmo propõe o uso de filtros de convolução piramidal ou *box* com parâmetros ajustáveis capazes de definir os níveis de ruídos desejados. Essa técnica de limar automático de multiníveis se mostrou eficiente e com baixo custo computacional.

Bertholdo e Araújo (2007) em sua dissertação de mestrado propôs um algoritmo de limiarização para melhorar a qualidade e legibilidade das imagens digitalizadas de documentos históricos, degradados devido ao desgaste natural do tempo. Para esta aplicação, normalmente é utilizada o método de Otsu. O algoritmo de Bertholdo e Araújo (2007) obteve resultados eficientes, sendo capaz de melhorar a qualidade visual e legibilidade de 80% das imagens do acervo Dops/MG de documentos históricos. Seu algoritmo utilizou uma abordagem híbrida, que combina características globais e locais.

Seixas et al. (2008) analisaram diferentes métodos de segmentação em imagens de Ressonância Magnética da seção sagital do crânio. Os métodos utilizados foram: limiarização em multiníveis de Otsu modificado por algoritmos genéticos, limiarização em multiníveis de Niblack, entropia máxima, limiarização em multiníveis de Otsu com função GGM e limiarização em multiníveis de Rosin. Os resultados experimentais mostraram que, para as imagens testadas, o método mais eficiente foi o de Otsu otimizado por algoritmo genético.

Beuren, Pinheiro e Facon (2011) aplicaram a segmentação por limiarização global Renyi para a extração automática do câncer do melanoma. Esse método de limiarização foi escolhido pelos autores por se mostrar o mais eficiente no resgate das lesões sem prejudicar a geometria e o formato e sem adicionar ou remover partes das mesmas, dentre os outros métodos de binarização disponíveis na literatura.

2.2 Segmentação Orientada a Região

A segmentação orientada a região utiliza propriedades espaciais da imagem para determinar a segmentação. Dentre os métodos de segmentação orientada a região, os mais comuns são o de crescimento de regiões e o de divisão e fusão de regiões.

A segmentação orientada a região segue os seguintes critérios:

- $\bigcup_{i=1}^{n} R_i = R$,
- R_i é uma região conexa, $i = 1, 2, ..., n$,

- $R_i \cap R_j = \varnothing$ para todo i e j, $i \neq j$,

- $P(R_i) = true$ para $i = 1,2,...,n$, e

- $P(R_i \cup R_j) = false$ para $i \neq j$,

onde R é a região completa da imagem, a imagem é segmentada em n partições (R_1,R_2,...,R_n) e P(R_i) é um operador lógico, tal que, é verdadeiro caso a propriedade de similaridade definida seja satisfeita por todos os pixels da região R_i e falsa caso contrário. As propriedades de similaridades mais usuais são intensidade de nível de cinza, cor e textura.

2.2.1 Crescimento de Regiões (*Region Growing*)

A segmentação por crescimento de regiões é um procedimento que agrupa *pixels* ou subregiões em regiões maiores (GONZALEZ; WOODS, 2000). O algoritmo mais simples é o de agregação de *pixels*. Este algoritmo se inicia com um conjunto de sementes (*pixels*) dos quais crescem regiões pela junção de cada semente com *pixels* vizinhos[1] que compartilham propriedades semelhantes. Nessa técnica somente regiões adjacentes podem ser agrupadas, conforme critérios definidos anteriormente, pois cada semente se inicia como uma região distinta que pode-se unir somente a regiões vizinhas.

O primeiro método de crescimento de região foi desenvolvido por Brice e Fennema (1970), ele foi baseado simplesmente num conjunto de regras de crescimento. Primeiramente ele utilizava os critérios de 4-conectada e mesma amplitude de intensidade para gerar grupos chamados de regiões atômicas, e depois utilizava duas regras heurísticas para fundir regiões atômicas com fronteiras fracas (PRATT, 1991).

Os problemas desta técnica são: a definição das sementes (pontos iniciais), as escolhas dos critérios de similaridade e conectividade, e a formulação de uma regra de parada do algoritmo. Na figura 2.4 há um exemplo de segmentação por crescimento de região que se inicia com apenas uma semente.

[1] Segundo o critério de conectividade determinado.

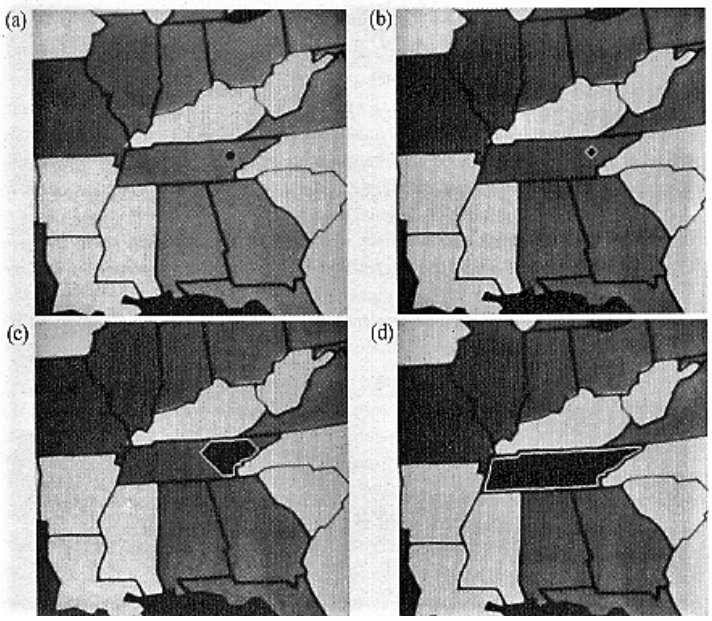
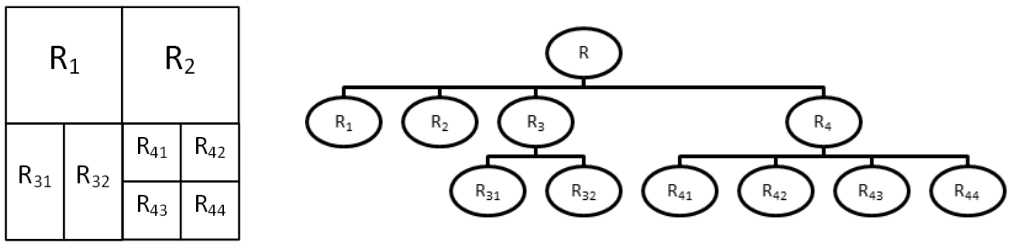


Figura 2.4: Imagem segmentada por crescimento de região com um ponto semente, 8-conectada e similaridade pelo menos de 90% do valor do nível de cinza. (a) Imagem original. (b) Estágio primário. (c) Estágio intermediário. (d) Região final. Fonte: Gonzalez e Woods (2000).

2.2.2 Divisão e Fusão de Regiões *(Split and Merge)*

A segmentação por divisão e fusão de regiões consiste em subdividir uma imagem em um conjunto arbitrário de regiões distintas e então juntar ou separar essas regiões segundo o critério de similaridade definido.

Os algoritmos mais comuns da técnica *split and merge* são baseados na representação da imagem por uma árvore de dados quadrada, chamada de *quadtree*. Esse algoritmo consiste em dividir (*split*) a imagem em quatro quadrantes e depois dividir cada um destes em mais quatro quadrantes caso eles não satisfaçam o critério de similaridade ou, ainda, unir (*merge*) quatro quadrados adjacentes caso eles sejam similares, repetindo esse procedimento até que as regiões se estabilizem. Na figura 2.5 há um exemplo de uma imagem particionada e sua *quadtree*. Na representação *quadtree* a raiz da árvore representa a imagem inteira e cada nó representa um subdivisão na imagem, de modo que todos os nós juntos formam a imagem completa.



(a) (b)

Figura 2.5: (a) Imagem particionada, (b) *Quadtree* da imagem

Caso o processo se inicie no nível da imagem completa, essa técnica tem boa precisão, porém, alto custo computacional. Por outro lado, caso o processo se inicie com uma árvore excessivamente ramificada tem-se custo computacional pequeno, porém, atrelado ao aumento dos erros decorrentes do fato das medidas de similaridade, inicialmente, estarem baseadas apenas em pontos vizinhos. Desse modo, o mais indicado é se iniciar o algoritmo em um estágio intermediário entre esses casos para conseguir um equilíbrio entre custo computacional e acurácia. Há muitas alterações proposta para esta técnica na literatura, estas muitas vezes pretendem encontrar melhores critérios de similaridades para definir a uniformidade dos quadrantes ou encontrar o melhor ponto para iniciar o algoritmo.

2.2.3 Trabalhos relacionados

Dlugosz et al. (2005) utilizaram a técnica crescimento de região do *software* SPRING para discriminar tipos de floresta em imagens de alta resolução do satélite IKONOS. Esta aplicação faz parte de um grande projeto que visa o zoneamento ambiental da Embrapa/Epagri Reserva Florestal localizado em Caçador, sudoeste do Estado de Santa Catarina, Brasil, e inclui plano de gestão e a acompanhamento da vegetação nativa. A área representa um importante fragmento da Floresta de Araucária no sul do Brasil e contém espécies ameaçadas, como o pinheiro-do-paraná (Araucaria angustifolia). A cobertura vegetal existente na reserva é constituída por um dos últimos remanescentes da Floresta Ombrófila Mista, onde se destacam algumas das espécies constantes da lista oficial de espécies em extinção do IBAMA (DLUGOSZ et al., 2005). Como resultado constatou-se que o algoritmo de segmentação mostrou-se eficiente para a discriminação e delimitação de diferentes porções da tipologia florestal típica da Floresta Ombrófila Mista.

Bins et al. (1996) apresentaram um método de segmentação baseado em crescimento de região, que foi implementado no software SPRING desenvolvido pela Divisão de Processa-

mento de Imagens (DPI) do Instituto Nacional de Pesquisas Espaciais (INPE). Esse método foi aplicado em imagens de satélite (Landsat/TM) usadas para avaliar mudanças no solo da região Amazônica. Os resultados preliminares dos testes realizados em imagens de áreas florestadas e agrícolas da região Amazônica mostraram-se satisfatórios apesar da simplicidade da técnica. Os autores pretendem, no futuro, melhorar o desempenho do algoritmo e aplicá-lo em outros tipos de imagens.

Fan et al. (2001) propuseram um novo método automático de segmentação de imagens híbrido baseado em crescimento de regiões com sementes iniciais determinadas por extração de bordas no espaço de cores YUV. O algoritmo se inicia com a obtenção automática de bordas coloridas por meio da combinação de um detector de bordas isotrópico aprimorado e uma técnica rápida de limiarização por entropia. As bordas obtidas fornecem as principais estruturas geométricas de uma imagem, os centroides entre regiões com fronteiras adjacentes são escolhidos como sementes iniciais do algoritmo de crescimento de região. Esse método foi aplicado em detecção automática de face, onde demonstrou bons resultados.

2.3 Agrupamento (*clustering*)

A segmentação por *clustering* refere-se ao agrupamento de um dado conjunto de objetos em subconjuntos de acordo com as propriedades de cada objeto. Um dos primeiros exemplos de segmentação de imagens, realizado por Haralick e Kelly (1969) com o uso de *clustering*, foi a subdivisão de imagens aéreas multi espectrais de terrenos agrícolas em regiões com o mesmo tipo de relevo (PRATT, 1991). O conceito de segmentação por *clustering* é simples, porém seu custo computacional é alto.

A técnica de *clustering* é um método de estatística multidimensional baseado em *Data Mining* que identifica grupos automaticamente segundo o grau de semelhança de suas características. Cada *pixel* da imagem de coordenada (j,k) é representado por um vetor de características N dimensional, $x = [x_1, x_2, ..., x_N]^T$, onde cada componente desse vetor contém uma métrica da imagem para o *pixel* cujo o qual representa. Essas métricas podem ter dimensões variáveis e podem ser desde uma característica individual do *pixel* como da região (vizinhança). Os algoritmos de *clustering* inicialmente desconhecem o número de grupos e suas características. Desse modo, geralmente esses algoritmos tem como parâmetro a ser definido o número de *clusters* e tem como medida de similaridade (semelhança) a distância euclidiana entre os vetores de características de dois pontos.

Existem basicamente dois tipos de clustering: divisivo e aglomerativo. No primeiro, a imagem é vista como um *cluster*, e então são realizadas sucessivas divisões até formarem *cluster* significativos. No segundo, cada *pixel* é visto como um *cluster*, e então *clusters* similares são unidos recursivamente até formarem um *cluster* significativo. Como resultado espera-se que os elementos pertencentes ao mesmo *cluster* sejam os mais semelhantes possíveis entre si, e os que pertencentes a *clusters* distintos apresentem pouca semelhança.

Três algoritmos comuns de *clustering* são: *K-means* ou ISODATA (COLEMAN; ANDREWS, 1979), *fuzzy c-means* (FCM) (BEZDEK; HALL; CLARKE, 1992) e o algoritmo maximização de expectativa (EM) (LIANG; MACFALL; HARRINGTON, 1994). O *K-means* agrupa dados computando iterativamente uma intensidade média para cada classe e segmentando a imagem pela classificação de cada pixel da classe com a média mais próxima. O FCM generaliza o algoritmo *K-means*, permitindo segmentações baseadas na teoria *fuzzy* de conjuntos. O algoritmo EM, por sua vez, aplica os mesmo princípios de *clustering* com o pressuposto de que os dados seguem o modelo Guassiano (PHAM; XU; PRINCE, 2000).

2.3.1 *K-means*

O algoritmo *K-means* foi um dos primeiros métodos de *clustering* desenvolvidos. Ele é um algoritmo simples de ser implementado, no entanto, tem a grande desvantagem de ser muito sensível às sementes iniciais, pois sementes diferentes podem resultar em *clusters* distintos. Desse modo, não há uma garantia de convergência para uma solução ótima. O algoritmo *K-means* tem como parâmetro a ser definido o número de *clusters* (grupos) que devem ser gerados, porém, em alguns casos, essa é uma tarefa difícil. Um número de *clusters* errado pode gerar resultados com uma estrutura errônea.

O algoritmo *K-means* consiste em agrupar os objetos de entrada em K grupos baseados na sua proximidade no espaço de características. Esse agrupamento é realizado em torno de K centroides, um para cada grupo (*clusters*). Um algoritmo *K-means*, cujas entradas são:

- uma imagem com N *pixels* de coordenadas (j,k);
- os N vetores de características de cada pixel (j,k), descritos por $x^{(j,k)}$; e
- o parâmetro número de *clusters* com valor K,

é constituído pela sequência de passos descritos abaixo.

1. Especificar inicialmente K sementes, cada uma delas como um *cluster*, representados por $C_i \; \forall \; i = 1, 2, ..., K$.

2. Calcular os K centroides, um para cada *cluster*, sendo eles descritos por $\mu_i \quad \forall i = 1, 2, ..., K$.

3. Calcular a distância euclidiana do vetor de características de cada pixel (j,k) para o centroide de cada *cluster*.

4. Atribuir cada *pixel* ao *cluster* mais próximo.

5. Recalcular os K centroides baseando-se nas atribuições mais recentes, através da média dos *pixels* pertencentes a cada *cluster*.

6. Repetir os passos 3, 4 e 5 até que todos os *clusters* e centroides permaneçam inalterados.

Um maneira de otimizar o algoritmo e impedir que ele entre em um fluxo infinito de repetições é com o cálculo do erro dado pela equação 2.2. Desse modo, as repetições do passo 6 devem parar quando o erro E for menor que um certo valor ou até se completarem um número máximo de iterações.

$$E = \sum_{i=1}^{K} \sum_{x^{(j,k)} \in C_i} ||x^{(j,k)} - \mu_i||^2 \tag{2.2}$$

Em alguns casos existe uma estimativa da posição dos centroides. Nesses casos, é possível melhorar a convergência do algoritmo ao iniciar as K sementes nesses locais previamente conhecidos. Para reduzir o impacto da inicialização aleatória, uma opção simples é realizar a execução do algoritmo diversas vezes, isso, contudo, aumenta bastante o custo computacional do algoritmo.

2.3.2 Trabalhos relacionados

Furuie, Moura e Udupa (1996) propõem uma técnica para a classificação de *voxels* de imagens médicas 3D baseado em *fuzzy clustering*. O algoritmo apresentado pelos autores foi aplicado em imagens tridimensionais de ressonância magnética do coração na diástole e tinha por objetivo segmentá-la em objetos pertinentes, como ventrículo esquerdo (VE) e ventrículo direito (VD), por exemplo. Os resultados encontrados apresentaram problemas em casos complexos, com falsos positivos isolados e com falhas em alguns *voxels*. A causa dessas falhas, segundo os autores, deve-se ao fato da técnica utilizar apenas informações locais.

Furuie, Moura e Udupa (1996) concluem ao final do trabalho que a classificação pode ser significativamente melhorada através de refinamento do pós-processamento ou da inclusão de informações prévias sobre os objetos, tais como morfologia e conectividade.

Bonventi Junior, Smith e Moreira (2015) aplicam o algoritmo FCM para segmentar imagens aéreas coloridas. O único atributo utilizado para realizar o *clustering* foi a cor da imagem no espaço RGB, pois a cor é invariante à rotação e a ampliação. A métrica de distância utilizada foi a Mahalanobis, cuja escolha se deve a necessidade de detectar grupos alongados de *pixels* com cores similares. O número ideal de *clusters* foi encontrado por minimização do índice de XieBeni. Como resultado da segmentação, os autores obtiveram uma boa separação entre estruturas ambientais visualmente diferentes, tais como florestas, água e solo exposto. Concluindo que esse método mostrou-se robusto para realizar a separação entre diferentes regiões de imagens aéreas coloridas.

O trabalho de Pham, Xu e Prince (2000) analisou as vantagens e desvantagens dos métodos de segmentação comumente utilizados para segmentar imagens médicas. Dentre os métodos de segmentação *clustering*, foram analisados *K-means*, FCM e EM. Com o algoritmo *K-means*, foi realizada a segmentação de uma imagem corte de ressonância magnética de um cérebro, onde optou-se por três *clusters* que representariam: fluido cerebral, massa cinzenta e massa branca. Segundo os autores, a vantagem dos algoritmos *clustering* é que eles não necessitam de treinamento, porém, como desvantagens, eles requerem parâmetros iniciais. Comparando os métodos EM, FCM e *K-means*, notou-se que o algoritmo EM foi o que demonstrou maior sensibilidade na inicialização.

Capítulo 3

Modelo de cores

A luz, ao atravessar os meios ópticos oculares, excita moléculas fotossensíveis dos fotorreceptores da retina que iniciam o processo de codificação da informação dos raios luminosos, ocorrendo, assim, a percepção das cores (BACKHAUS et al., 1998). Cores são definidas como ondas eletromagnéticas descritas pelo seu comprimento de onda, geralmente especificado em nanômetros (*nm*). A percepção de cores é um fator subjetivo que varia de pessoa para pessoa. Desse modo, com o objetivo de especificar um padrão de cores, foram definidos inúmeros modelos de cores. Alguns modelos são orientados para *hardware*, como, por exemplo, o RGB (*red*, *green*, *blue*), que é voltado para monitores coloridos e câmeras, o CMYK (*cyan*, *magenta*, *yellow*, *key*), para impressoras, e o YIQ, que é o padrão utilizado para transmissão de TV colorida (GONZALEZ; WOODS, 2000). Outros são orientados para aplicações envolvendo manipulações de cores, como, por exemplo, HSI (*hue*, *saturation*, *intensity*) e HSV (*hue*, *saturation*, *value*) (GONZALEZ; WOODS, 2000). Cada modelo de cor pode ter vários espaços de cores, como, por exemplo, o RGB, que tem o sRGB, o Adobe RGB e outros. Espaço de cores é definido como um modelo matemático usado para a representação geométrica das cores no espaço, usualmente em três dimensões.

A Comissão Internacional de Iluminação (CIE - *Commission Internationale de l'Éclairage*) definiu padrões de colorimetria baseados na percepção da cor pelo olho humano, determinadas pela sistematização das funções de misturas de cores necessárias para um observador padrão dentro de um campo visual de 2°, em condições específicas de iluminação. Dentre eles, estão os padrões CIELAB, CIERGB, CIExyY e CIEXYZ. Os modelos de cor padrões de referência usuais são o CIELAB ou o CIEXYZ, que foram projetados para abranger todas as cores que os seres humanos podem ver. O diagrama de cromaticidade do sistema CIERGB é mostrado na figura 3.1 e o do sistema CIEXYZ na figura 3.2. É possível notar pela com-

paração dessas figuras que o CIERGB perde um região cromática considerável se comparada ao CIEXYZ.

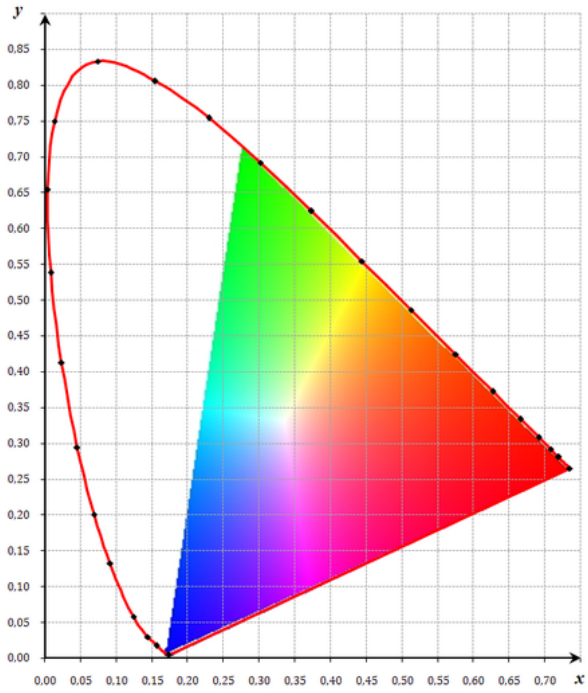


Figura 3.1: Representação do diagrama de cromaticidade CIERGB, dentro do diagrama CIE (linha vermelha).

Em 1931, a CIE elaborou os sistemas colorimétricos CIEXYZ e CIExyY. O CIExyY é derivado do CIEXYZ e representa as cores de acordo com a sua cromaticidade (eixos x e y) e a sua luminância (Y). Na figura 3.2 há o diagrama de cromaticidade da CIE, onde os eixos x e y representam a cromaticidade do modelo CIExyY. O diagrama de cromaticidade representa todas as cores visíveis para uma pessoa padrão, região chamada de *gamut* (gama) da visão humana. A extremidade da curva *gamut* é chamada de locus espectral e corresponde a luz monocromática do comprimento de onda indicado, ou seja, cada ponto representa um único comprimento de onda do espectro visível - uma cor pura. A linha reta da borda inferior do *gamut* é chamada de linha dos púrpuros, pelo fato dessas cores, mesmo estando na borda do *gamut*, não terem luz monocromática equivalente - elas são composta por duas radiações monocromáticas: azul e vermelha. O branco está localizado na posição central $(x = 0,33; y = 0,33)$, e é resultado da combinação dos três comprimentos de onda adotados como primárias pelo CIE, 700 nm (vermelho), 546,1 nm (verde) e 435,8 nm (azul).

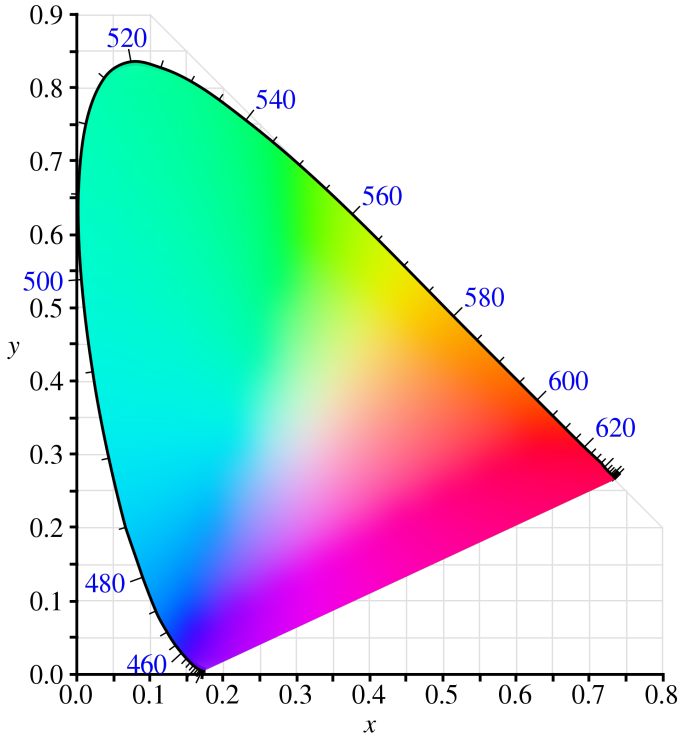


Figura 3.2: Diagrama de cromaticidade da CIE (1931).

No sistema CIEXYZ as quantidades de vermelho, verde e azul necessárias para formar qualquer cor em particular são denominadas valores triestímulos e são denotados por X, Y e Z, respectivamente (GONZALEZ; WOODS, 2000). Uma cor é especificada por seus valores tricromáticos (x,y,z), que representam as coordenadas de cromaticidade e são obtidas pelas seguintes equações em 3.1. Estas três coordenadas (x,y,z) correspondem às proporções de cada uma das três primárias para constituir cada cor espectral.

$$x = \frac{X}{X+Y+Z} \quad y = \frac{Y}{X+Y+Z} \quad z = \frac{Z}{X+Y+Z} \tag{3.1}$$

Somando essas equações tem-se que $x+y+z=1$, portanto, o coeficiente tricromático z é obtido pela equação 3.2.

$$z = 1-(x+y) \tag{3.2}$$

O sistema CIExyY caracteriza as cores pelo seu brilho e cromaticidade. A cromaticidade é representada por x e y do diagrama da figura 3.2 e o brilho pela luminância (Y). Desta forma, as cores têm informações nas três dimensões (x,y,z), enquanto o branco e o preto estão no eixo Y que representa o nível de luminância, expresso em cd/m^2.

O CIE 1931 vem sendo utilizado sem alterações substanciais desde a sua concepção. Alguns pontos foram interpolados ou aprimorados, seus valores numéricos foram expressos com aproximação de sete dígitos, mas os fundamentos continuam os mesmos (SCHANDA, 1998).

3.1 CIELAB

Em 1976, para preencher as lacunas do modelo CIExyY, o CIE desenvolveu o espaço de cor CIELAB, também chamado de CIEL*a*b*. O espaço de cor CIELAB é uma aproximação de escala uniforme de cor, onde diferenças no espaço correspondem a diferenças visuais. Ele foi desenvolvido com o objetivo de ser independente de dispositivos, por isso, como foi mencionado no capítulo 3, ele é puramente matemático. Nesse modelo, uma cor é localizada por três valores: a luminância (L) e as duas gamas de cor (a e b). A luminância contém as informações de luminosidade e é expressa em porcentagem, variando de 0 (preto) a 100 (branco). As gamas de cor contêm a informação de cor, elas não tem valores máximos numéricos, porém utilizam como referência o valor de 60 unidades de cor. Os componentes de cor verde são representado por valores negativos do gama a, os de cor vermelha pelo positivo de gama a, os de cor azul pelo negativo do gama b e os de cor amarelo pelos positivo do gama b.

Na figura 3.3 é possível visualizar o espaço de cor CIELAB, no qual a saturação é determinada pelas coordenadas das duas gamas de cor (a e b) e o brilho pela coordenada Y.

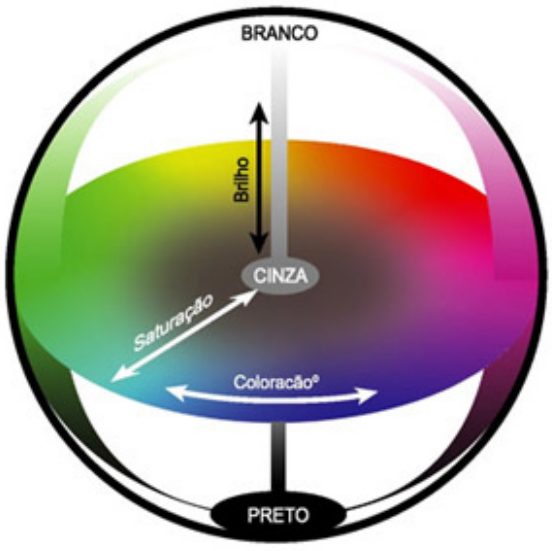


Figura 3.3: Espaço de cor tridimensional CIELAB.

Desse modo, o CIELAB cobre a integralidade do espectro visível pelo olho humano e representa-o de maneira uniforme, permitindo, por conseguinte, descrever o conjunto das cores visíveis, independentemente de qualquer tecnologia gráfica. Como o CIELAB compreende a totalidade das cores dos modelos RGB e CMYK, alguns *softwares* como o *PhotoShop* utilizam este modo para passar de um modelo de representação a outro.

A conversão do modelo XYZ para o LAB é dado pelo conjunto de equações 3.3. Essa conversão tem alto custo computacional, o que faz com que alguns *softwares* evitem-na.

$$
\begin{aligned}
L* &= 116\left(\frac{Y}{Y_0}\right)^{1/3} - 16, && \text{para} \quad \frac{X}{X_0} > 0,008856 \\
a* &= 500\left(\left(\frac{X}{X_0}\right)^{1/3} - \left(\frac{Y}{Y_0}\right)^{1/3}\right), && \text{para} \quad \frac{Y}{Y_0} > 0,008856 \\
b* &= 200\left(\left(\frac{Y}{Y_0}\right)^{1/3} - \left(\frac{Z}{Z_0}\right)^{1/3}\right), && \text{para} \quad \frac{Z}{Z_0} > 0,008856 \\
X_0 &= 95,047 \quad Y_0 = 100,000 \quad Z_0 = 108,883
\end{aligned}
\tag{3.3}
$$

3.2 RGB

O modelo RGB de cores teve origem na teoria de Thomas Young, baseada no princípio de que diversos efeitos cromáticos são obtidos pela projeção da luz branca através dos filtros vermelho, verde e azul.

No modelo RGB cada cor aparece nos seus componentes espectrais primários de vermelho, verde e azul. Esse modelo é um modelo aditivo de cores que se baseia num sistema de coordenadas cartesianas (GONZALEZ; WOODS, 2000). Na figura 3.4 é mostrado o cubo RGB. Nos vértices do cubo estão as cores vermelho, verde, azul, ciano, magenta e amarelo. A cor preta está localizado na origem do cubo e a branca na extremidade oposta. A diagonal do cubo entre as cores preto e branco é a escala de cinza.

O RGB é, atualmente, o modelo de cor mais utilizado. Existem diversos espaços de cores que utilizam esse modelo, sendo os mais comuns o sRGB e o Adobe RGB. Por isso, ele é reproduzido de diferentes maneiras, dependendo das capacidades do sistema utilizado. O mais comum é o uso de 24 bits, com 8 bits ou 256 níveis para cada canal. Qualquer espaço de cor com base no modelo RGB de 24 bits é restrita a faixa de, aproximadamente, $256 \times 256 \times 256 \approx 16,7$ milhões de cores. Algumas implementações usam 48 bits (16 bits por

canal), resultando no mesmo *gamut*, porém com um número maior de cores reproduzíveis. O aumento da quantidade de bits é importante para espaços de cores com maior *gamut* e para implementações com um grande número de filtragens digitais.

O modelo RGB é utilizado por câmeras e monitores. O sRGB é o espaço de cor mais utilizado atualmente, sendo usado em câmeras digitais, câmeras de vídeo HD e monitores de computador. É conveniente utilizar o mesmo espaço de cores em quase todos os equipamentos, o que evita a necessidade de conversão entre espaços de cores ao mudar a imagem de um equipamento para outro. O espaço sRGB tem a desvantagem de ter um *gamut* limitado, sendo preterido em aplicações de alta qualidade, como no caso de HDTV. O espaço Adobe RGB apresenta um *gamut* maior que o sRGB, o que o faz ser preferido por artistas gráficos.

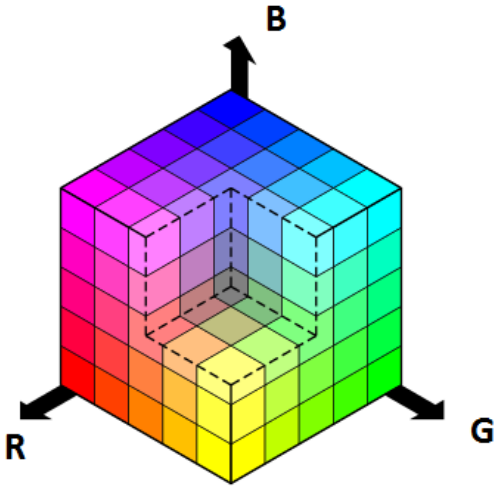


Figura 3.4: Cubo RGB.

Capítulo 4

Superpixel

A segmentação de imagens baseada em redes complexas é limitada pelo custo computacional, pois cada *pixel* da imagem é mapeado como um nó na rede. Logo, uma imagem de tamanho $M \times M$ é mapeada em uma rede composta por M^2 nós. No processamento de imagens é usual empregar técnicas de pré-segmentação para reduzir o número de *pixels* em uma imagem. Estas técnicas têm como objetivo reduzir o custo computacional das tarefas subsequentes e, por conseguinte, do processo total. Uma delas é a de *superpixels*.

Superpixels é uma técnica de segmentação de imagens baseada em regiões que tem por objetivo representar imagens com um número limitado de grupos de *pixels* (REN; MALIK, 2003). Um *superpixel* é constituído por um conjunto de *pixels* com características semelhantes, capaz de representar a imagem em regiões mais naturais e significantes. Muitas aplicações em visão computacional são otimizadas ao trabalhar com *superpixels* no lugar de *pixels* (WANG et al., 2011; FULKERSON et al., 2009). *Superpixels* são de interesse especial para segmentação semântica, área em que há relatos sobre vantagens desta aplicação (BERGH et al., 2012). Nesta área eles são capazes de reduzir o número de entidades semânticas rotuladas e de permitir a computação de características em regiões maiores e mais significantes.

Em segmentações baseadas em grafos ou redes, é comum o uso de *superpixels* como pré-processamento, pois eles fornecem uma eficiente e compacta representação do grafo ou da rede devido à redução do número de nós, ou seja, da cardinalidade. Logo, pode-se obter um aumento substancial na velocidade de processamento da imagem, desde que o número de *superpixels* não seja muito grande, de 25 a 2500 em média (NEUBERT; PROTZEL, 2012).

O método de *superpixel* pode ser utilizado como uma segmentação ou como uma pré-segmentação para reduzir a quantidade de pixels. Essa última é capaz de capturar a redundância na imagem e reduzir significativamente a complexidade das tarefas subsequentes de

processamento de imagem, reduzindo o custo computacional total do processo (SHI; MALIK, 2000). Algumas de suas aplicações são: estimativa de profundidade (HOIEM; EFROS; HEBERT, 2005), segmentação de imagens (CUADROS, 2013; BOTELHO, 2014; TORRES, 2012; SARATH, 2014), esqueletização (LEVINSHTEIN; DICKINSON; SMINCHISESCU, 2009), localização de objeto (FULKERSON et al., 2009), *tracking* (WANG et al., 2011) e biomédicas (AKBAR et al., 2015).

O uso de *superpixel* como pré-processamento exige um maior esforço computacional devido à necessidade de construção das unidades. Ao mapear os *superpixels* de uma imagem, é importante evitar a perda de dados relativos às bordas da imagem, evitando colocá-las dentro de um *superpixel*. Existem várias formas de gerar *superpixels*, cada qual com suas vantagens e desvantagens, sendo importante escolher qual algoritmo e parâmetros melhor se adaptam a aplicação requerida. Métodos baseados em grafos, por exemplo, são indicados para aplicações nas quais a aderência às fronteiras da imagem são de grande importância. Já em aplicações onde *superpixels* são usados para construir grafos, métodos que gerem uma grade de *superpixels* mais regular são mais recomendados. Achanta et al. (ACHANTA et al., 2012) definiu as seguintes propriedades para qualificar *superpixels*:

- *Superpixels* devem aderir bem às fronteiras da imagem;
- Quando usado para reduzir a complexidade computacional como um passo de pré-processamento, *superpixels* devem ser computados rapidamente, ter eficiência de memória e serem simples de usar;
- Quando usados para propósito de segmentação, *superpixels* devem aumentar a velocidade e a qualidade dos resultados;

O algoritmo original de *superpixel* foi introduzido por Ren e Malik (REN; MALIK, 2003) como uma solução em agrupamentos de *pixels* em regiões coerentes (características semelhantes de cor, textura e brilho), ele foi baseado no *Normalized Cut* de Malik e Shi (SHI; MALIK, 2000). O algoritmo de Ren e Malik (REN; MALIK, 2003) utiliza o algoritmo *Normalized Cut* para gerar um grafo ponderado da imagem, no qual são realizados cortes para particionar a imagem em um grande número de regiões pequenas, compactas, quasi-uniformes e disjuntas, obtendo, assim o mapa de *superpixel* da imagem com coerência dos atributos de contorno e textura. No grafo da imagem, cada *pixel* gera um nó e os pesos das arestas ligando os nós são definidos por uma função de similaridade entre os *pixels*, esta considerando características de contorno e textura. Para gerar o particionamento (corte) do grafo,

é calculada a discrepância entre os diferentes grupos e a semelhança entre os elementos do mesmo grupo. Atualmente, este método é considerado lento e possui eficiência razoável em relação à aderência de fronteiras.

Os métodos de extração de *superpixel* são recentes e, de acordo com a literatura, os algoritmos que merecem destaque são: *Normalized cuts* (SHI; MALIK, 2000), *Mean Shift* (COMANICIU; MEER, 2002), *Felsenszwalb* (FELZENSZWALB; HUTTENLOCHER, 2004), *Quickshift* (VEDALDI; SOATTO, 2008), *Turbopixel* (LEVINSHTEIN et al., 2009), *Simple linear iterative clustering* (ACHANTA et al., 2010), *Superpixels Extracted via Energy-Driven Sampling* (BERGH et al., 2012) e *Speeded-up turbopixels* (CIGLA; ALATAN, 2010). Segundo o estado da arte da literatura em aplicações de *superpixel* como pré-processamento para posterior segmentação em grafos ou redes, os métodos *Simple linear iterative clustering* (SLIC), *Superpixels Extracted via Energy-Driven Sampling* (SEEDS) e *Speeded-up turbopixels* (SUTP) são considerados mais eficientes. Os métodos SLIC e o SUTP são baseados em *K-means* e apresentam divisão inicial da imagem em regiões retangulares, que intercambiam seus pixels até formar os superpixels. O método SEEDS, por sua vez, tem como objetivo gerar *superpixel* em tempo real. Para isso, ele utiliza uma otimização de *hill-climbing*, função de energia e divisão inicial da imagem em regiões retangulares. Na figura 4.1 há um exemplo de extração de superpixel com os métodos de *Felzenszwalbs*, SLIC e *Quickshift*.

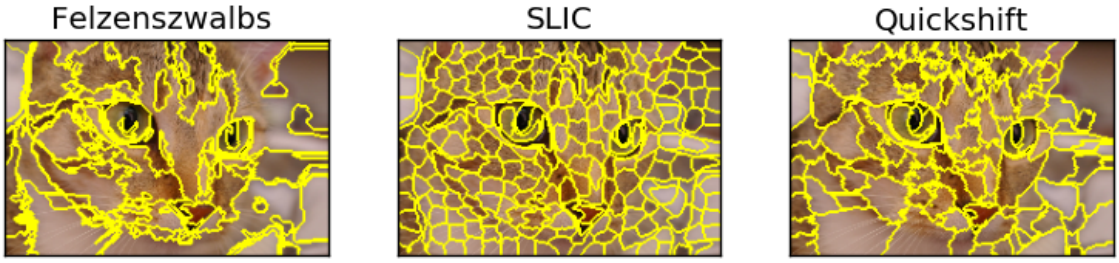
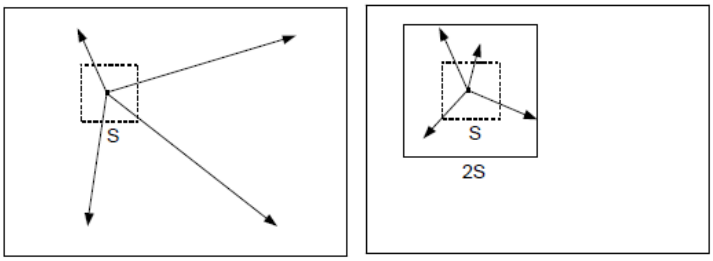


Figura 4.1: Respectivamente método de *Felzenszwalbs*, SLIC e *Quickshift*.

4.1 *Simple Linear Iterative Clustering*

No artigo de Achanta et al. (2010) é proposto o algoritmo SLIC. Este é um algoritmo iterativo com complexidade de ordem $O(N)$, onde N é o número de *pixels*, para extração de *superpixel* em imagens coloridas, considerando o espaço de cores CIELAB. O SLIC é uma adaptação do algoritmo *K-means* com espaço de busca reduzido à uma proporção do tamanho de um superpixel como é mostrado na figura 4.2. Essa adaptação do SLIC reduz o seu custo

computacional e faz com que a complexidade do algoritmo seja independente do número de *superpixels*.



(a) Espaço de busca *k-means*. (b) Espaço de busca SLIC.

Figura 4.2: Redução do espaço da busca do *superpixel*. A complexidade de SLIC é linear no número de *pixels* da imagem, $O(N)$, enquanto o algoritmo *K-means* convencional é $O(kNI)$, onde I é o número de iterações. (a) Espaço de busca empregado por um algoritmo *K-means* convencional, as distâncias são computadas de cada centro para todos os *pixels* da imagem. (b) Espaço de busca reduzido do algoritmo SLIC, de cada centro para *pixels* dentro da região $2S \times 2S$. Adaptado de Achanta et al. (2012).

O algoritmo SLIC gera *superpixels* mediante o agrupamento de *pixels* baseado na combinação de similaridade de cor e proximidade espacial. O algoritmo forma agrupamentos locais de *pixels* no espaço *Labxy*, no qual L, a, b é a cor do *superpixel* no espaço CIELAB e x, y sua posição.

O algoritmo tem como entrada uma imagem de N *pixels* e os parâmetros k e m, respectivamente, o número de *superpixels* e a constante de compacidade. Primeiramente, ele realiza um particionamento da imagem em k regiões retangulares, onde cada região compõe um *superpixel* inicial de extensão espacial aproximadamente $S = \sqrt{\frac{N}{k}}$, portanto, dimensões $S \times S$ e área S^2.

Na sequência, é atribuído para cada *superpixel* j um vetor de 5 dimensões definido por $C_j = [L_j, a_j, b_j, x_j, y_j]$, onde L_j, a_j, b_j correspondem à média da cor, no espaço CIELAB, dos *pixels* que compõem o *superpixel* e x_j, y_j à média das coordenadas dos *pixels* que compõem o *superpixel*. O vetor C_j é o *cluster center* do *superpixel* j. Os agrupamentos centrais iniciais são movidos para posições correspondentes ao menor gradiente em uma vizinhança de 3×3. Esse passo é realizado para evitar colocar *cluster center* em uma borda e reduzir as chances de se escolher um pixel ruidoso. Os gradientes da imagem são computados calculados pela equação 4.1, onde $I(x,y)$ é o vetor de cor no espaço CIELAB:

$$G(x,y) = \|I(x+1,y) - I(x-1,y)\|^2 + \|I(x,y+1) - I(x,y-1)\|^2 \tag{4.1}$$

O algoritmo segue com iterações em que são percorridos os agrupamentos centrais dos

k superpixels e é realizado o intercâmbio dos *pixels* localizados na região de busca de cada *superpixel*, esta é formada pela área $2S \times 2S$ em torno do centroide do *superpixel*. Neste intercâmbio de *pixels*, cada *pixel i* é associado ao *superpixel* que minimize a função de distância *D*, representada pelas equações em 4.2. Após percorrido os *k cluster center*, estes são recalculados levando em consideração o novo conjunto de *pixels* que compõem os *superpixels*. O algoritmo, então, repete iterativamente esses passos de associar *pixels* com os agrupamentos centrais mais próximos, até sua convergência. Finalmente, após as finalizar as iterações, existem alguns *pixels* que não correspondem a uma mesma componente conexa, estes são reagrupados em uma operação de pós-processamento.

A função de distância *D* é função do cálculo de duas distâncias: d_{lab} (distância no espaço de cor CIELAB) e d_{xy} (distância euclidiana). Nas equações 4.2, *j* representa o *superpixel* atual e *i* os *pixels* que pertencem a região de busca do *superpixel j*. A distância Euclidiana comparada a distância no espaço de cor CIELAB é perceptivamente significativa para distâncias pequenas. Desse modo, é necessário normalizá-las, para evitar que as distâncias espaciais dos *pixels* excedam o limite perceptual de cor, resultando em *superpixels* que não respeitam limites da região, mas apenas sua proximidade. A razão $\frac{m}{S}$ é responsável por essa normalização e acrescenta o parâmetro *m*, que permite o controle da compacidade dos *superpixels*. Quando *m* é pequeno, os *superpixels* têm maior aderência à bordas e fronteiras, porém têm tamanho e forma menos regular. Para espaço de cor CIELAB, *m* pertence ao intervalo $[1,40]$ (ACHANTA et al., 2012).

$$\begin{aligned} d_{lab} &= \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2} \\ d_{xy} &= \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \\ D &= \sqrt{d_{lab}^2 + \left(\frac{d_{xy}}{S}\right)^2 m^2} \end{aligned} \tag{4.2}$$

Segundo Achanta et al. (2010) os *superpixels* acurados podem ser obtidos com, no máximo, 10 iterações. Além disso, há uma adaptação para imagens em escala cinza, alterando a equação de distância no espaço de cor para: $d_{xy} = \sqrt{(l_j - l_i)^2}$. O algoritmo SLIC está descrito em algoritmo 1.

Algoritmo 1: SLIC *Superpixel*. Adaptado de Achanta et al. (2010).

Entrada:
I: imagem (*N pixels*);
k : número de *superpixels*;
m: constante de compacidade;
Saída:
$l(i)$: *label* (*i pixels* agrupados em *k superpixels*);

início
  // Inicialização
  $S = \sqrt{\frac{N}{k}}$; /* comprimento aproximado do lado dos superpixels */
  grid(*I*,*S*) ; /* particiona a imagem em k regiões retangulares */
  $l(i) = -1$; /* inicializa *label* para cada *pixel* i */
  $d(i) = \infty$; /* inicializa distâncias para cada *pixel* i */
  $C_j = [L_j, a_j, b_j, x_j, y_j]$; /* inicializa os k agrupemntos centrais */
  Mover os agrupamento centrais ao menor gradiente em uma vizinhança de 3×3
  repita
    // Iterações
    para cada *cluster center* C_j *(j=1 até j=k, passo 1)* **faça**
      para cada *pixel i da região* $2S \times 2S$ *em torno do* C_j **faça**
        $d_{lab} = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2}$
        $d_{xy} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$
        $D = \sqrt{d_{lab}^2 + \left(\frac{d_{xy}}{S}\right)^2 m^2}$
        se $D < d(i)$ **então**
          $d(i) = D$; /* atualiza distância mínima entre Cj e i */
          $l(i) = j$; /* agrupa *pixel* i no *superpixel* j */
        fim
      fim
    fim
    Atualizar os *k* agrupamentos centrais;
    Atualizar Erro;
  até *Erro* $\leq$ *threshold*;
fim
retorna $l(i)$

4.2 *Speeded-up Turbo Pixels*

No artigo de Cigla e Alatan (2010) é proposto o algoritmo SUTP. Este é um algoritmo iterativo para extração de *superpixels* com complexidade de ordem $O(N)$, onde N é o número de *pixels*. Ele é baseado no algoritmo de agrupamento *k-means* e é capaz de gerar *superpixels* quase uniformes com baixo custo computacional.

O SUTP tem como entrada uma imagem de N *pixels* e os parâmetros k, λ_1 e λ_2, respectivamente, o número de *superpixels*, o coeficiente de similaridade e o coeficiente de restrição de

convexidade. O algoritmo SUTP se inicia com um particionamento da imagem em k regiões retangulares, denominados segmentos (*superpixels*), de extensão espacial aproximadamente $S = \sqrt{\frac{N}{k}}$, portanto, dimensões $S \times S$ e área S^2. No passo seguinte, os *pixels* sobre as bordas dos segmentos são testados iterativamente e intercambiados ao segmento que minimize a função de custo apresentada na equação 4.3.

$$C_{x,y}(i) = \lambda_1 |I(x,y) - I_i| + \lambda_2 |(x - C_x^i)^2 + (y - C_y^i)^2| \tag{4.3}$$

Na equação 4.3, o termo I_i indica a intensidade média do i-ésimo segmento, x e y são as coordenadas do *pixel* testado entre os diferentes segmentos, $I(x,y)$ é a intensidade do *pixel* testado, C_x^i e C_y^i são as coordenadas do centroide do i-éssimo segmento. O primeiro termo da equação 4.3 garante a similaridade da intensidade de cor dos *pixels* que serão agrupados, enquanto o segundo termo garante a convexidade dos *superpixels*, evitando que *pixels* distantes sejam agrupados. Os parâmetros λ_1 e λ_2 são restritos ao intervalo $[0,1]$. Após testados todos os *pixels* das bordas, as intensidades médias (I_i) e os centroides dos segmentos são atualizados. As iterações têm como critério de parada um limite máximo de permutações dos *pixels*, de modo que, se o número de *pixels* intercambiados está abaixo do estabelecido, o algoritmo é finalizado. O algoritmo SLIC está descrito em algoritmo 2.

Como essa técnica testa apenas os *pixels* nas bordas dos segmentos, ela garante a conectividade dos *superpixels*, ou seja, não existiram *pixels* que não pertencem a nenhum *superpixel*. A técnica de extração de *superpixel* SUTP fornece um algoritmo rápido, pois um número limitado de *pixels* é testado entre os segmentos vizinhos (máximo de 4 conetados) (CIGLA; ALATAN, 2010).

Algoritmo 2: SUTP *Superpixel*. Adaptado de Cigla e Alatan (2010).

Entrada:
I: imagem (N *pixels*);
k : número de *superpixels*;
λ_1: constante de similaridade;
λ_2: constante de convexidade;
Saída:
$l((x,y))$: *label* ((x,y) *pixels* agrupados em k *superpixels*);

início
  // Inicialização
  $S = \sqrt{\frac{N}{k}}$; /* comprimento aproximado do lado dos superpixels */
  grid(I,S) ; /* particiona a imagem em k segmentos C_i */
  para cada *segmento* C_i *(*$i = 1$ *até* $i = k$*, passo* 1*)* **faça**
    $num = 0$; ; /* conta número de *pixels* do segmento i */
    para cada *pixel pertencente ao segmento* C_i **faça**
      $l((x,y)) = i$; /* inicia cada *pixel* (x,y) no *superpixel* i */
      $c_{x,y}(i) = \lambda_1 |I(x,y) - I_i| + \lambda_2 |(x - C_x^i)^2 + (y - C_y^i)^2|$; /* custo atual */
      $I_i = \sum I(x,y)$, $C_x^i = \sum x$, $C_y^i = \sum y$, $num = num + 1$;
    fim
    $I_i = \frac{I_1}{num}, C_x^i = \frac{C_x^i}{num}, C_y^i = \frac{C_y^i}{num}$; ; /* inicia I_i, C_x^i, C_y^i segmento i */
  fim
  repita
    // Iterações
    para cada *segmento* C_i *(*$i = 1$ *até* $i = k$*, passo* 1*)* **faça**
      para cada *pixel* (x,y) *pertencente a borda do segmento* C_i **faça**
        $C_{x,y}(i) = \lambda_1 |I(x,y) - I_i| + \lambda_2 |(x - C_x^i)^2 + (y - C_y^i)^2|$
        se $C_{x,y}(i) < c((x,y))$ **então**
          $c((x,y)) = C_{x,y}(i)$; /* atualiza o custo mínimo */
          $l((x,y)) = i$; /* agrupa o *pixel* (x,y) no *superpixel* i */
        fim
      fim
    fim
    Atualizar os k centroides
  até *número de pixels intercambiados está abaixo do estabelecido*;
fim
retorna $l((x,y))$

4.3 Trabalhos relacionados

Torres (2012) propõe dois métodos automáticos de segmentação de imagens para auxiliar a sua árdua tarefa de segmentar manualmente imagens de Tomografia Computorizada. Os métodos propostos são baseado na partição de grafo orientada à micro região, sendo, no primeiro método, obtidas por transformada *watershed* e, no segundo, por *superpixels*. Como

resultado, o método baseado em *superpixels* produziu melhores resultados, segundo a autora, devido à maior uniformidade em tamanho e forma das micro regiões resultantes.

Sarath (2014) aplicou a técnica SLIC de extração de *superpixel* para mensurar áreas foliares. Ele utilizou essa aplicação como uma ferramenta de medição automática dos danos causados pela herbivoria nas folhas de soja e obteve resultados condizentes com as medidas de referência.

Wang et al. (2011) propõem um método para *tracking* de objetos a partir da perspectiva de visão de nível médio, com informações estruturais capturadas em *superpixels*. Seu modelo visa distinguir objeto e fundo, assim facilitando o *tracking*. A extração de *superpixels* foi realizada pelo algoritmo SLIC. Este método mostrou-se capaz de lidar com fortes oclusões do objeto e recuperar-se de seus desvios. Além disso, apresentou bons resultados quando comparado a métodos de *tracking* de objetos já existentes.

Fulkerson et al. (2009) propõem um método para identificar e localizar classes de objetos em imagens, que utiliza *superpixels* como unidade básica. Seu classificador foi construído no histograma de características locais encontrados em cada *superpixel*, estes foram gerados pelo algoritmo *quick shift*. Os resultados desse método se mostraram superiores aos métodos do estado da arte da época.

Capítulo 5

Redes Complexas

O experimento de Milgram (MILGRAM, 1967) é um conhecido exemplo de rede. Em 1967, Milgram enviou cartas para uma série de pessoas nos Estados Unidos com informações básicas sobre uma pessoa aleatória do mesmo país e perguntando se o receptor a conhecia. Caso o destinatário não a conhecesse, ele deveria enviar a mesma carta, com acréscimo do seu nome, para um amigo que ele julgasse ter a maior probabilidade de a conhecer, repetindo esse procedimento até alcançar o destinatário. Esse experimento deu início à expressão "Seis graus de separação", que afirma que cada indivíduo pertencente a um grande grupo conhecerá qualquer outro membro arbitrário do mesmo grupo a partir de uma cadeia de amizades de tamanho médio seis.

O conceito utilizado por Milgram para descrever sua rede social é semelhante ao de rede complexa, pois cada pessoa se conecta com outra de acordo com uma regra pré estabelecida, assim como, numa rede, cada nó (elemento) se conecta a outro por uma regra definida, gerando arestas para representar as interações entre os nós.

Diversos sistemas e dados são compostos por uma quantidade de elementos que possuem relações entre si. É possível, portanto, organizá-los por meio de suas relações (ligações) como um modelo de rede. Redes têm sido objeto de estudo em inúmeros campos da ciência, tais como matemática, sociologia e ciência da computação. Diversas relações do mundo real podem ser representadas como uma rede: uma célula pode ser descrita como uma rede de substâncias químicas ligadas por reações químicas, ou a Internet, como uma rede de roteadores e computadores conectados (CUADROS, 2013). Redes com muitos nós ou com estruturas muito dinâmicas (que evoluem ao longo do tempo) são denominadas na literatura de redes complexas.

Redes complexas é uma área de pesquisa que surgiu recentemente com o intuito de es-

tudar grande volume de dados modelados como redes e tem atraído a atenção de pesquisadores, sendo aplicada em diversos domínios. O termo redes complexas refere-se a um grafo que apresenta uma estrutura topográfica não trivial, composto por um conjunto de vértices (nós) que são interligados por meio de arestas (BARABÁSI, 2003). O primeiro problema modelado em forma de um grafo data de 1736, quando Leonhard Euler publicou um artigo demonstrando que não existia solução para o problema das Sete Pontes de Königsberg (EULER, 1741). Em 1998, Bollobás publicou *Modern Graph Theory*, onde estão reunidas as principais propriedades matemáticas da Teoria dos Grafos (BOLLOBÁS, 1998). A teoria das redes complexas nasceu da aplicação de medidas desenvolvidas pela teoria dos grafos e conceitos provenientes da mecânica estatística, física não-linear e sistemas complexos (NEWMAN, 2003; RODRIGUES, 2007).

As redes complexas possuem um caráter multidisciplinar e cobrem aplicações em diversas áreas do conhecimento, utilizando a computação como ferramenta para modelagem, tratamento e análise de dados. Seu estudo tornou-se viável apenas recentemente, devido ao aumento do poder de processamento e armazenamento dos sistemas computacionais. Muitas relações existentes, tais como a Internet, redes sociais entre indivíduos, redes organizacionais ou de negócios entre companhias, redes neurais, redes metabólicas e redes biológicas, foram modeladas como redes complexas. Alguns sistemas reais podem dar origem a redes com milhões ou mesmo bilhões de vértices.

A descoberta de que a topologia das redes reais apresenta propriedades organizacionais bastante robustas e distintas das redes aleatórias intensificaram o estudo desta área. As redes geradas a partir de dados reais podem envolver estrutura de comunidade, distribuição de graus de potência e *hubs*, e outras características estruturais. Essa é a principal razão de serem chamadas redes complexas.

É importante ressaltar que nem todo grafo pode ser considerado uma rede complexa. Essa classificação só é possível se o grafo apresentar algumas propriedades topográficas características de redes complexas.

5.1 Conceitos Teóricos

Esta seção apresenta os principais conceitos referentes a grafos e redes complexas. Esses conceitos são importantes para a compreensão das técnicas de segmentação de imagens baseada em detecção de comunidades em redes complexas.

5.1.1 Teoria de Grafos

Na Teoria dos Grafos, um grafo é a representação de um conjunto de objetos onde alguns pares de objetos são conectados por linhas. Esses objetos são representados por abstrações chamadas vértices e arestas (PORTO, 2014; TRUDEAU, 1993). Um grafo é definido por $G = (V, E)$, onde V é o conjunto não-vazio composto por elementos denominados vértices ou nós e um conjunto E de subconjuntos dos elementos de V, denominados arestas. Cada aresta de E é definida por um par $e = \{u, v\}$, sendo u e v vértices pertencentes a V, ou seja, $u, v \in V$. O número de vértices e arestas de G, também chamado de cardinalidade, é dado por $|V|$ e $|E|$, respectivamente. A ordem n de um grafo é o tamanho do conjunto de vértices, ou seja o número de vértices, $|V|$. O tamanho m de um grafo é o tamanho do conjunto de arestas, ou seja, o número total de arestas do grafo, $|E|$.

Um grafo pode ser orientado, chamado de dígrafo, ou não orientado, indicando respectivamente, se existe ou não uma direção entre os dois nós de cada aresta. Um grafo também pode ser definido como um grafo ponderado quando as arestas apresentam valores numéricos, chamados de pesos. Um grafo pode conter laços, que são arestas ligando um vértice a ele mesmo.

Para um grafo ponderado existem uma função peso $w(u, v)$, que atribui um peso à aresta que liga u a v. Para grafos não orientados ela obedece a propriedade: $w(u, v) = w(v, u)$. Caso não exista uma aresta que ligue u e v é atribuído peso nulo, ou seja, $w(u, v) = 0$. Grafos não ponderados consideram todas as arestas existentes com peso 1, $w(u, v) = 1$.

Um grafo também pode ser representado pela matriz de adjacência, W, que é uma matriz quadrada de ordem n (dimensão $n \times n$), onde $n = |V|$, e cujos elementos w_{ij} representam as arestas e são dados pela função peso $w(i, j)$, representado a ligação do nó i para o nó j. Devido a propriedade $w(u, v) = w(v, u)$ de grafos não orientados, sua matriz de adjacência é simétrica. Caso o grafo não tenha laços, a diagonal principal de W é nula.

A somatória dos pesos das arestas incidentes em um determinado vértice i é chamado de grau do nó, $deg(i) = \sum_{n}^{j=1} W_{ij}$. No caso de grafos orientados, o grau de um nó pode ser dividido em grau de entrada e grau de saída. Um grafo é dito regular se todos os seus vértices têm o mesmo grau, e é dito k-regular se $deg(v) = k$ para todo vértice $v \in V$. Um grafo é dito completo quando há uma aresta entre cada par de seus vértices. O grafo completo de n vértices é frequentemente denotado por K_n e possui o número máximo possível de arestas, dado pela equação 5.1, sendo $(n-1)$-regular.

$$\binom{n}{2} = \frac{(n)(n-1)}{2} \tag{5.1}$$

Vizinhança de um nó consiste no conjunto de nós que formam ligação com o mesmo, ou seja, os nós adjacentes a ele. Um grafo conexo é aquele em que há um caminho entre qualquer par de vértices e, se não for o caso, ele é chamado desconexo.

Seja $G = (V, E)$ um grafo qualquer, $G' = (V', E')$ é dito subgrafo de G se $V' \subset V$ e $E' \subset E$, ou seja, um subgrafo é um grafo cujos conjuntos de vértices e arestas estão contidos em outro grafo. Um particionamento de grafo é o conjunto de subgrafos disjuntos que unidos formam o grafo. Matematicamente, seja um grafo $G = (V, E)$ k-particionado nos conjuntos $V_1, V_2, ..., V_k$, tem-se que $V_1 \cup V_2 \cup ... \cup V_k = V$ e $V_i \cap V_j = \varnothing$, para $i, j \in \{1, 2, ..., k\}$. Esses subgrafos são denominados partições do grafo e as arestas que ligam os vértices em diferentes partições são chamadas arestas de corte. O corte de uma partição é igual a soma dos pesos das arestas de corte. O corte mínimo em um grafo $G = (V, E)$ é definido como o corte que resulta no valor mínimo da soma dos pesos das arestas que compõem o corte.

5.1.2 Medidas para caracterização de redes complexas

Redes complexas, assim como grafos, são descritas por um conjunto de vértices e arestas e também é representada matematicamente por $R = (V, E)$. As redes complexas podem apresentar diversas topologias e características de acordo com o domínio estudado e, utilizando algumas medidas, podem ser analisadas, caracterizadas e modeladas (COSTA et al., 2007; ALMEIDA, 2009). Elas têm uma série de propriedades locais e globais que já foram alvo de sucessivos estudos (NEWMAN, 2003).

Uma das propriedades mais utilizadas é a conectividade local de um vértice (NEWMAN, 2003). A conectividade de um vértice i, chamada de k_i, é número de arestas diretamente conectadas a ele, ou seja, o número de vizinhos do vértice. A conectividade é dada pela equação 5.2:

$$k_i = \sum_{j=1}^{N} a_{ij} \tag{5.2}$$

onde N é o número de vértices da rede e a_{ij} são os elementos da matriz de adjacência da rede. A conectividade média de uma rede é dada pela média da conectividade de cada vértice da rede, portanto:

$$\langle k \rangle = \frac{1}{N} \sum_{i=1}^{N} k_i \tag{5.3}$$

Algumas medidas importantes relacionadas à conectividade dos vértices são a conectividade máxima da rede e a distribuição da conectividade $P(k)$, que são, respectivamente, a conectividade do vértice mais conectado $k_{max} = \max_i k_i$ e a probabilidade de que um vértice escolhido aleatoriamente tenha grau ou conectividade k. A segunda medida é utilizada para determinar qual o modelo da rede.

Em uma rede ponderada (de conexões com pesos) é utilizada a medida força (*strength*), que é dada pela soma de todos dos pesos w_{ij}, que são os elementos da matriz de adjacência, das arestas incidentes no vértice. Para um vértice v_i a força s_i é dada por:

$$s_i = \sum_{j=1}^{N} w_{ij} \tag{5.4}$$

A força média de uma rede é dada pela média da força de cada vértice da rede, portanto:

$$\langle s \rangle = \frac{1}{N} \sum_{i=1}^{N} s_i \tag{5.5}$$

As medidas de força máxima e distribuição de força são calculadas como as respectivas medidas de conectividade.

O coeficiente de agrupamento (*clustering coefficient*), cc_i, expressa a presença de ciclos mínimos em uma rede, ou seja, ciclos com apenas três vértices formando "triângulos". Seja v_i um vértice qualquer que possui k_i vizinhos: caso esses k_i vértices formassem um subgrafo completo, teriam $\frac{k_i(k_i-1)}{2}$ arestas. Sendo e_i o número de arestas que realmente existem entre os k_i vértices, o coeficiente de agrupamento é dado por:

$$cc_i = \frac{2e_i}{k_i(k_i-1)} \tag{5.6}$$

Um alto coeficiente de agrupamento indica que um vértice central v_i e os seus vizinhos possuem coesão, de modo que v_i pode ser um bom representante do agrupamento. O coeficiente de agrupamento médio é dado por:

$$\langle cc \rangle = \frac{1}{N} \sum_{i=1}^{N} cc_i \tag{5.7}$$

Analisando esta distribuição de probabilidade, é possível extrair diversas medidas, tais

como entropia, energia e a média do grau de junção. Elas podem ser calculadas com a força do vértice, s_i ou a conectividade k_i. São definidas, respectivamente, por:

$$H = -\sum_{i=1}^{N} P(k',k') \log_2 P(k',k') \tag{5.8}$$

$$E = \sum_{i=1}^{N} (P(k',k'))^2 \tag{5.9}$$

$$P = \frac{1}{N} \sum_{i=1}^{N} P(k',k') \tag{5.10}$$

A média do grau de junção é definida como a probabilidade de uma rede ter dois vértices com o mesmo grau.

O comprimento do caminho que conecta dois vértices é obtido pela quantidade de arestas ao longo deste caminho. O caminho mínimo entre dois vértices, $d_{i,j}$, é definido como o conjunto de arestas que ligam os dois vértices com o menor custo total. Em uma rede que não tem pesos, esse caminho é dado diretamente pela quantidade de arestas. Caso não haja um caminho ligando esses vértices, tem-se que $d_{i,j} = \infty$. O maior valor da matriz em uma rede conexa, $d_{max} = \max_{i,j} d_{i,j}$, é denominado diâmetro da rede.

Existem medidas de centralidade, que servem para quantificar a importância de um vértice ou aresta para a comunicação na rede. O grau de proximidade (*closeness centrality*) de um vértice é a média dos caminhos mínimos desse vértice para qualquer outro vértice da rede. O grau de intermediação de um vértice u (*betweenness centrality*) definido por Freeman (1977) é calculado pela equação 5.11:

$$b(u) = \sum_{s \neq u \neq t} \frac{\sigma_{st}(u)}{\sigma_{st}} \tag{5.11}$$

onde $\sigma_{st}(u)$ é o número de menores caminhos entre s e t que passaram por u, e σ_{st} é o total de caminhos mínimos entre s e t. Portanto, se o valor de $b(u)$ for alto, o vértice u exerce uma posição central na rede, logo tem grande influência na comunicação da rede. A média do grau de intermediação é dado pela equação 5.12:

$$\langle B \rangle = \frac{1}{N} \sum_{u \in V} b(u) \tag{5.12}$$

A partir dessa medida, é possível calcular a dominância do ponto central pela equa-

ção 5.13:

$$c_D = \frac{1}{N-1} \sum_{u \in V} (B_{max} - b(u)) \tag{5.13}$$

A resistência da rede indica a capacidade de resistência da rede quanto às remoções de alguns vértices, sem que haja perda de sua funcionalidade. A remoção de vértices pode resultar na perda de conexão entre pares de vértices ou, ainda, aumentar significativamente o caminho de um vértice a outro.

5.1.3 Modelos de redes complexas

Ao longo da segunda metade do século XX, um dos objetivos em teoria dos grafos era propor novos modelos e catalogar as suas propriedades. Assim sendo, com o objetivo de estudar propriedades topológicas de redes reais, diversos modelos de redes complexas foram introduzidas na literatura.

Em Costa et al. (2011) são apresentadas diversas aplicações e modelos de redes complexas em problemas reais. Nesta seção, são apresentados três modelos de redes complexas importantes historicamente para o desenvolvimento desta área: redes aleatórias, redes de pequeno mundo e redes livres de escala.

Redes Aleatórias

O primeiro modelo de rede foi sugerido por Erdös e Rényi em 1959, chamado de Redes Aleatórias (*Random Networks*) (ERDÖS; RÉNYI, 1959, 1960, 1961). Esse pode ser considerado o modelo mais básico de redes complexas (COSTA et al., 2007). Nesse modelo existem dois parâmetros, o número total de vértices N e uma probabilidade p ($0 \leq p \leq 1$). Cada par de vértices é conectado de acordo com essa probabilidade, independentemente dos outros vértices. Nessas redes não existem qualquer hierarquia de organização das arestas, ou seja, não existem critérios que privilegiem uma ligação em relação a outra – as ligações são aleatórias. O número esperado de arestas é $m = p\left(\frac{N(N-1)}{2}\right)$ e a conectividade média é $\langle k \rangle = p(N-1)$ (ALBERT; BARABÁSI, 2002).

A distribuição de graus, $P(k)$, desse modelo é bem aproximada por uma distribuição de Poisson de valor médio $\langle k \rangle$, de modo que todos os vértices possuem grau próximo de $\langle k \rangle$ (NEWMAN, 2003). O caminho mínimo médio nesse tipo de rede tende a ser bem pequeno, crescendo proporcionalmente ao logaritmo de n: $l \propto ln(n)$ (RODRIGUES, 2007).

Quando a probabilidade de formação de ligações for zero, $p = 0$, a rede será completamente desconectada. Quando $p = 1$, a rede será um grafo completo e, portanto, o valor médio do coeficiente de agrupamento será máximo, $\langle cc \rangle = 1$ (ROCHA, 2007).

Na figura 5.1 há exemplos de redes aleatórias com número de vértices fixos em $N = 8$ e com valor de probabilidade variando para ilustrar sua influência na rede. Como é possível observar nessa figura, quando $p = 0$ a rede é desconexa – não tem arestas – quando $p = 0,1$ a rede tem poucas arestas, quando $p = 0,3$ há a adição de mais arestas, o que possibilita a formação de ciclos no grafo, por fim, quando $p = 0,8$ a rede gerada aproxima-se de um grafo completo (OLIVEIRA, 2008). Na figura 5.2 há um exemplo de rede aleatória e o gráfico da distribuição de graus de 10 redes desse modelo, onde pode-se notar a semelhança com a distribuição de Poisson.

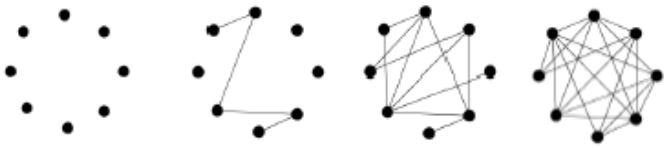
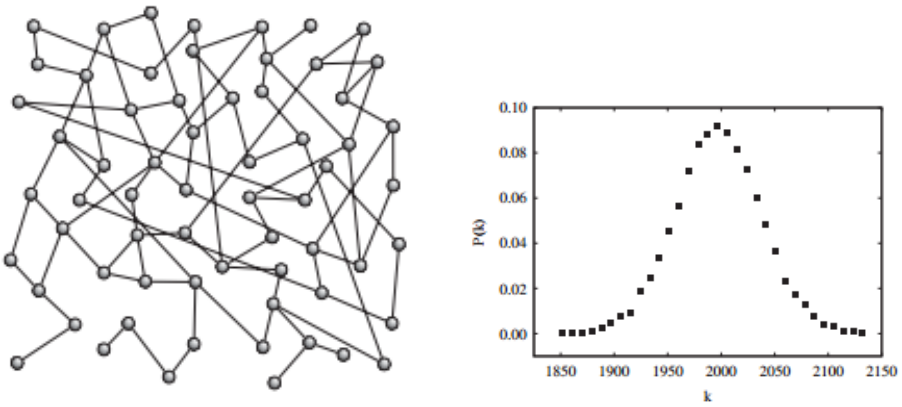


Figura 5.1: Exemplos de redes aleatórias com $N = 8$ e, respectivamente, $p = 0$, $p = 0,1$, $p = 0,3$ e $p = 0,8$. Adaptada de (OLIVEIRA, 2008).



(a) (b)

Figura 5.2: Rede aleatória de Erdös e Rényi: (a) um exemplo (b) distribuição média dos graus de 10 redes aleatórias formadas por 10.000 vértices usando uma probabilidade $p = 0,2$. Adaptado de Costa et al. (2007).

Redes pequeno mundo

Duncan Watts e Steven Strogatz observaram que em algumas redes reais foram encontrados muito mais caminhos fechados com apenas três vértices do que nos modelos de redes aleatórias com o mesmo número de vértices e arestas (WATTS; STROGATZ, 1998; WATTS,

1999a, 1999b). Esse fato mostrou que as redes reais não são completamente aleatórias e que existe algum tipo de lei de formação por trás da sua construção (RODRIGUES, 2007). Essa descoberta os motivou a idealizar um novo modelo de rede capaz de contemplá-la, tal efeito é conhecido como mundo pequeno (*small world*) e foi observado pelo experimento de Milgram em 1967.

Em 1998, Watts e Strogatz formalizaram o conceito de redes de pequeno mundo (*small world networks*), modelo de rede no qual existem poucas ligações entre os vértices da rede, porém a maioria dos vértices pode ser alcançado com poucos passos a partir de qualquer outro vértice (WATTS; STROGATZ, 1998). A maioria das redes complexas apresentam o efeito de mundo pequeno, por exemplo, a internet possui[1] $\ell \approx 10$, a World Wide Web $\ell \approx 11$, as moléculas nas células $l \approx 3$ e as espécies em cadeias alimentares $\ell \approx 2$ (BARABÁSI, 2003; NEWMAN, 2003; ALBERT; BARABÁSI, 2002; ALMEIDA, 2009).

Rede pequeno mundo é um modelo simples que apresenta características complementares às redes aleatórias, sendo semelhante ao de modelo de Erdös e Rényi. Essas redes têm as seguintes propriedades: baixa média de comprimento dos caminhos mínimos da rede e vértices com um alto coeficiente de agrupamento, ou seja, grande quantidade de ciclos curtos, o que caracteriza o efeito de vizinhança. Desse modo, a distância média entre quaisquer dois vértices de uma rede muito grande não ultrapassa um número pequeno de vértices. Para isso, basta que algumas conexões aleatórias entre grupos sejam estabelecidas (NEXUS, 2002).

Para construir uma rede de pequeno mundo, o modelo se inicia com N vértices organizados na forma de um anel, com cada vértice ligado aos seus κ vizinhos mais próximos. A seguir, as arestas são redirecionadas de acordo com a probabilidade p atribuída a cada uma delas de ter o vértice destino alterado. As arestas ao serem redirecionadas obedecem duas restrições: nenhum vértice pode se ligar com ele mesmo e não pode haver mais de uma conexão entre um par qualquer de vértices da rede. Para $p = 0$ nenhuma aresta é modificada, resultando na rede regular original. A escolha aleatória de uma fração $p > 0$, gera algumas arestas aleatórias, resultando em uma rede de pequeno mundo. Para valores pequenos de p, $p \approx 0,01$, é produzida uma rede na qual o coeficiente de agrupamento ainda é elevado, mas que ao mesmo tempo é uma rede mundo pequeno. Se $p \approx 1$ e $p = 1$, a rede gerada será equivalente a uma rede aleatória. Logo, o parâmetro p interpola entre um comportamento completamente regular e um completamente aleatório.

No exemplo de rede pequeno mundo apresentado na figura 5.3 a rede inicial possui $N =$

[1] O símbolo ℓ representa média dos caminhos mínimos

20 vértices e $\kappa = 4$, rede para $p = 0$. Nessa figura é ilustrada o efeito de p, quando $p = 0$ tem-se uma estrutura ordenada regular com alto número de laços, porém com grandes distâncias entre os vértices: quando $p \to 1$ a rede torna-se aleatória com distâncias curtas mas poucos laços. Assim, para valores intermediários de p o modelo da rede é de pequeno mundo apresentando tanto distâncias curtas quanto um grande número de laços.

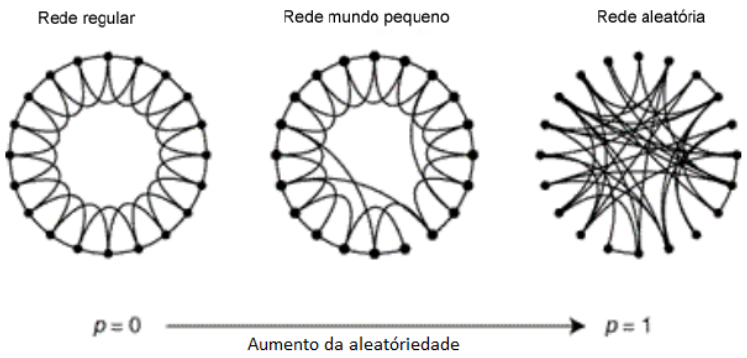


Figura 5.3: Modelo de rede pequeno mundo demonstrando o efeito da variação de valores de probabilidade(p). Adaptada de Watts (1999a) e Albert e Barabási (2002).

Redes livres de escala

Depois do modelo de redes de pequeno mundo de Watts (1999b), Barabási e Albert (1999) mostraram que muitos sistemas reais são caracterizados por uma distribuição desigual, muitas redes do mundo real apresentam vértices com mais conexões do que outros. Ao invés dos vértices destas redes terem um padrão aleatório de conexões com um grau característico, como acontece nas redes aleatórias e nas redes mundo pequeno, o grau dos vértices não tem um valor típico representativo. Mais especificamente, foi encontrado que para redes reais a probabilidade $P(k)$ de que um vértice na rede esteja conectado a outros k vértices segue a lei de potência, descrita como $P(k) \sim k^{-\gamma}$, onde $\gamma > 0$ é o expoente de escala. Redes que obedecem a lei de potência também são chamadas de redes livre de escala (*Scale- Free Networks*).

Em 1999, Barabási e Albert propuseram um modelo promissor para modelar redes livre de escala, que foi baseado no estudo de ponteiros dentro da *World Wide Web* (BARABÁSI; ALBERT, 1999). O modelo de rede de Barabási-Albert é baseado em duas regras: crescimento da rede ao longo do tempo e ligação preferencial a vértices com grau elevado. A rede é gerada iniciando com a adição de novos vértices. Para cada novo vértice, m novas arestas são inseridas entre os novos vértices e alguns vértices já existentes. Os vértices que receberam as novas arestas são escolhidos seguindo uma regra de ligação preferencial linear,

ou seja, a probabilidade do novo vértice i estar conectado com um vértice j já existente é proporcional ao grau de j, matematicamente:

$$P(i \rightarrow j) = \frac{k_j}{\sum_{u=1}^{N} k_u} \tag{5.14}$$

Na figura 5.4 é exemplificado o processo de geração de uma rede livre de escala, onde os nós com mais ligações são representado por círculos maiores e os nós novos são brancos.

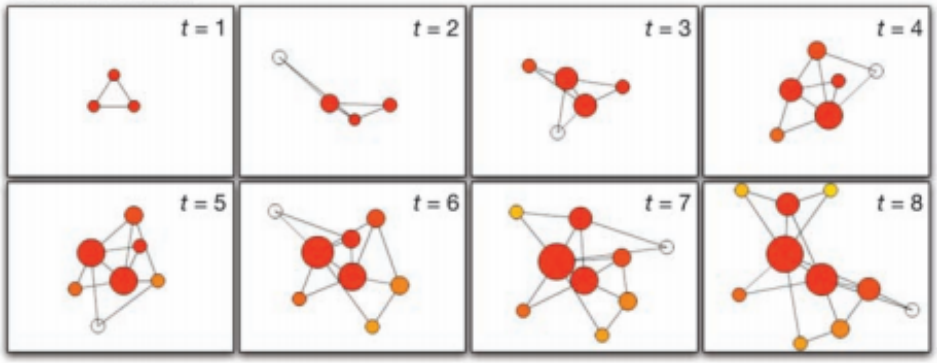


Figura 5.4: Exemplo de geração de uma rede livre de escala. Adaptada de Barabási (2009).

Uma das principais características das redes de livre escala é a denominada conexão preferencial, ou seja, a tendência de um novo vértice se conectar a um vértice da rede que tem um grau elevado de conexões. Desse modo, os vértices mais bem conectados adquirem ainda mais conexões do que aqueles que são pouco conectados, gerando a concentração das conexões em apenas alguns nós, denominados *hubs* ou vértice centralizador (BARABÁSI, 2009). Segundo Newman (2003), essas redes têm sido observadas em vários sistemas, como: internet, Web, redes de metabolismos e redes de citações de artigos científicos. Na figura 5.5 é apresentado um exemplo de rede livre de escala.

Outros modelos de redes livre de escala têm sido propostos visando flexibilizar o γ, que no modelo de Barabási e Albert é igual a três quando $N \rightarrow \infty$, e manter um coeficiente de agrupamento similar ao de redes reais (ALMEIDA, 2009; DOROGOVTSEV; MENDES, 2002).

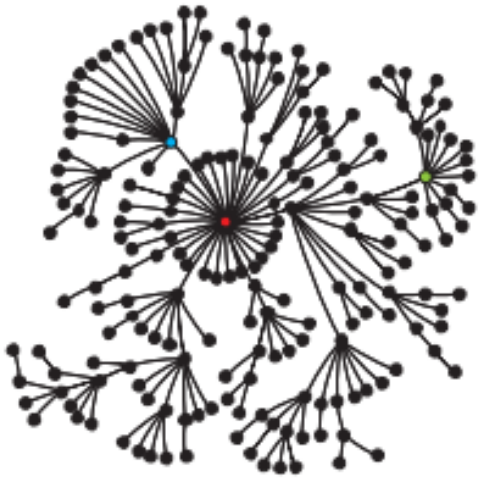


Figura 5.5: Rede Livre de Escala com 200 vértices, 199 arestas e presença de *hubs*. Os vértices que possuem maior grau são: vermelho $k = 33$; azul $k = 12$ e verde $k = 11$. Adaptada de Strogatz (2001).

5.2 Detecção de Comunidades

Uma propriedade bastante comum em redes complexas, que ocorre na maioria dos sistemas reais, é a presença de estruturas modulares chamadas de comunidades (LANCICHINETTI; FORTUNATO, 2009; ALMEIDA, 2009). A estrutura de comunidades é a organização dos nós de uma rede ou grafo em grupos (subgrafos) onde existe uma maior densidade de arestas entre os nós da mesma comunidade, e uma baixa densidade de arestas entre nós de diferentes comunidades (OLIVEIRA; ZHAO, 2008). O conceito de detecção de comunidades aproxima-se do conceito de corte ou particionamento de grafos (KARGER, 2000). A habilidade de encontrar e analisar tais comunidades podem prover valorosa ajuda para entender e visualizar a estrutura global do sistema analisado (NEWMAN; GIRVAN, 2004). A decomposição de uma rede em comunidades, é importante para a compreensão das relações entre diferentes componentes, além de permitir identificar funções de um componente com base nas funções de seus membros (OLIVEIRA, 2008).

As técnicas de detecção de comunidades vêm produzindo particionamentos acurados e com custo computacional satisfatório (NEWMAN; GIRVAN, 2004; RAGHAVAN; ALBERT; KUMARA, 2007; PORTO, 2014; ALMEIDA, 2009). Uma das aplicações recentes da detecção de comunidades que vem produzindo resultados satisfatórios é a segmentação de imagens (BOTELHO, 2014; CUADROS, 2013; BELIZARIO, 2012). Para isso, uma imagem é mapeada em uma rede complexa, de modo que a identificação de comunidades é utilizada para identificar os objetos da imagem. A grande limitação dessa aplicação é o tamanho da ima-

gem, pois como, cada *pixel* da imagem é representado como um nó, uma imagem $N \times N$ terá N^2 nós. Uma solução encontrada para isso é a utilização de *superpixel*, ao invés de apenas *pixels* (BOTELHO, 2014; CUADROS, 2013; BELIZARIO, 2012).

O objetivo do particionamento em grafos é dividir qualquer grafo em grupos de tamanho similar. A detecção de comunidades, por outro lado, procura grupos com similaridade entre seus nós. Além disso, em geral, o número de comunidades em uma rede e seu tamanho não são conhecidos, sendo estes normalmente estabelecidos pelo próprio algoritmo de detecção de comunidades.

Dada uma rede com n nós e m arestas, qualquer algoritmo de detecção de comunidades é capaz de encontrar subgrupos de nós. Sejam $C_1, C_2, ..., C_k$ as k comunidades encontradas, então temos as seguintes propriedades (FIDUCCIA; MATTHEYSES, 1982):

- $C_i \cap C_j = \varnothing$ para $i \neq j$;
- $\bigcup C_i$ abrange todos os n nós da rede.

Nas comunidades, vértices com uma posição central (que possuem muitas arestas ligando-o à própria comunidade) têm papel fundamental no controle e estabilidade do grupo. Já vértices localizados em posição de fronteira, ou seja entre os módulos, conduzem as comunicações entre as comunidades (FORTUNATO, 2010).

Uma medida muito utilizada para a caracterização dos vértices na estrutura de comunidades é a integração. Integração é a medida do quanto o vértice está relacionado ou integrado à comunidade que pertence, ou seja, é a quantidade relativa de vizinhos de um elemento que pertencem à sua comunidade. O grau interno k_{int} de um nó é o número de conexões daquele nó que pertencem à mesma comunidade. O grau externo k_{ext}, por sua vez, é o número de conexões com vértices de outras comunidades (ORMAN; LABATUT; CHERIFI, 2011). A integração de um vértice pode ser definida como pela razão entre seu grau interno e seu grau, conforme a equação 5.15.

$$e = \frac{k_{int}}{k_{ext} + k_{int}} = \frac{k_{int}}{k} \tag{5.15}$$

Quando um vértice tem valor de integração 1, isso significa que todos os seus vizinhos estão em sua própria comunidade. O valor da integração igual a 0, só é possível se o vértice em questão for uma comunidade isolada. Em redes do mundo real, a maioria dos vértices têm grau total baixo, e um valor de integração alto (ORMAN; LABATUT; CHERIFI, 2011).

Detectar a melhor divisão em comunidades de uma rede é um grande problema da detecção de comunidades. Visto que em redes reais geralmente não apresentam nenhuma informação sobre o número e/ou tamanho das comunidades existentes. Para solucionar esse problema, vem sendo propostas soluções que baseiem-se em heurísticas como, por exemplo, medidas de qualidade. Newman e Girvan (2004) propuseram uma medida denominada modularidade, que mede a qualidade de uma divisão particular da rede.

Na literatura existem várias propostas para descobrir a estrutura das comunidades de uma rede (NEWMAN; GIRVAN, 2004; CLAUSET; NEWMAN; MOORE, 2004; RAGHAVAN; ALBERT; KUMARA, 2007). Um resumo de propostas para a identificação de comunidades pode ser encontrado em Fortunato (2010). Muitos algoritmos de detecção de comunidades foram desenvolvidos baseados na medida da modularidade, como, por exemplo, o *Fast Greedy*. Porém devido a seu alto custo computacional, têm sido desenvolvidos algoritmos que não fazem uso da modularidade e que apresentem um baixo custo computacional, por exemplo, o *Label Propagation*. Nas próximas subseções, são apresentadas algumas abordagens para a detecção de comunidades com cálculo automático da quantidade de comunidades.

5.2.1 Modularidade

A métrica modularidade Q proposta por Newman e Girvan (2004) é uma das métricas mais utilizadas para avaliar a qualidade do agrupamento da rede. Com base nesta medida, pode-se quantificar a qualidade do particionamento obtido por um determinado algoritmo de detecção de comunidades.

Considerando a divisão de uma rede em k comunidades, é definida uma matriz simétrica E de ordem $k \times k$, onde o elemento e_{ij} é a fração de todas as arestas da rede que unem os vértices da comunidades i e com os vértices da comunidade j. O traço dessa matriz $Tr(E) = \sum_i e_{ii}$, portanto, representa a fração de arestas da rede que conectam os vértices da mesma comunidade i. A fração das arestas que conectam os vértices da comunidade i com as outras comunidades j é dada pela somatória das i linhas ou colunas da matriz E, sendo definida por $a_i = \sum_j e_{ij}$. A modularidade é definida por:

$$Q = \sum_{i}^{k} (e_{ii} - a_i^2) \tag{5.16}$$

De acordo com a medida de modularidade, valores muito próximos de 0, $Q \approx 0$, indicam baixa probabilidade da rede estar dividida em comunidades reais, visto que a estruturas de co-

munidades encontrada pelo algoritmo não é melhor do que se as arestas fossem simplesmente colocadas de maneira aleatória. Conforme o valor de Q aproxima-se de 1, $Q \approx 1$, a modularidade representa uma boa qualidade de agrupamento. Na prática, valores de $Q \geq 0,3$ indicam uma boa qualidade na divisão e que a estrutura de comunidades encontrada compensa ser considerada (NEWMAN; GIRVAN, 2004).

Encontrar o arranjo de comunidades para uma rede que maximize Q é um problema NP-completo (BRANDES et al., 2006). Portanto, considerando a tecnologia atual, a otimização da modularidade é inviável computacionalmente, devido a seu alto custo computacional e, consequentemente, alto tempo de execução. Diante disso, surgiram vários algoritmos capazes de encontrar boas aproximações do valor de modularidade máximo em um tempo razoável (BOTELHO, 2014).

5.2.2 *Fast Greedy*

Clauset, Newman e Moore (2004) propuseram um algoritmo hierárquico aglomerativo baseado na modularidade para detectar comunidades de uma rede, que foi chamado de *Fast Greedy* (FG) e apresenta complexidade computacional de ordem $O(md \log n)$, onde m é o número de arestas da rede, n é o número de vértices e d é a profundidade do dendrograma. Ele é uma otimização do algoritmo proposto por Newman e Girvan (2004), que utiliza uma técnica gulosa. Muitas redes reais são esparsas e hierárquicas, de modo que $m \sim n$ e $d \sim \log n$, para esses casos, a complexidade computacional é da ordem $O(n \log^2 n)$, portanto, se aproxima do tempo linear (CLAUSET; NEWMAN; MOORE, 2004).

O *Fast Greedy* utiliza de heurísticas para obter as comunidades de uma rede que fornecem o maior valor de modularidade (Q), para com isso determinar uma divisão ótima da rede. O algoritmo envolve encontrar as mudanças (Δ) na modularidade (Q) que resultem da união de cada par de comunidades, escolhendo o maior valor de ΔQ. Para implementar este método, a rede é vista como um multigrafo, onde uma comunidade inteira é representada por um vértice, pacotes de arestas conectam um vértice à outro e as arestas internas são representadas por autoarestas (*self-edges*). Para uma rede de n vértices, após $(n-1)$ uniões, o algoritmo terá apenas uma comunidade, sendo, portanto, concluído em no máximo $(n-1)$ iterações. O método pode ser representado como uma árvore, chamada de dendrograma, com a decomposição hierárquica da rede em comunidades, onde as folhas são os vértices da rede original e os nós internos são as respectivas uniões de comunidades.

A união de duas comunidades i e j na matriz de adjacência para um multigrafo é feita

substituindo as i_{th} e j_{th} linhas e colunas pela sua soma. No algoritmo Newman e Girvan (2004), essa operação é executada explicitamente na matriz. No entanto, se a matriz de adjacência é esparsa, a operação pode ser feita mais eficientemente usando estruturas de dados para matrizes esparsa (CLAUSET; NEWMAN; MOORE, 2004). O algoritmo *Fast Greedy*, ao invés de continuamente computar a matriz de adjacência e calcular ΔQ_{ij}, apenas armazena e atualiza o valor ΔQ_{ij} da matriz. Como juntar duas comunidades sem arestas entre elas não produz incremento em Q, é suficiente armazenar ΔQ_{ij} para os pares i, j unidos por pelo menos uma aresta. Além disso, o algoritmo usa a eficiente estruturas de dados *max-heap* para armazenar os maiores valores de ΔQ_{ij}. No total, o algoritmo mantém três estruturas de dados, sendo elas:

- Uma matriz esparsa contendo ΔQ_{ij} para cada par i, j de comunidades com pelo menos uma aresta entre elas.
- Um *max-heap H* que contém o maior valor de cada linha d matriz ΔQ_{ij}, com os rótulos i, j das correspondentes comunidades.
- Um vetor com os elementos a_i (equação apresentada na seção 5.2.1).

O algoritmo se inicia com cada vértice da rede sendo um membro único de uma comunidade, de modo que $e_{ij} = \frac{1}{2m}$ caso os nós i e j estejam conectados e $e_{ij} = 0$ caso contrário. E $a_i = \frac{k_i}{m}$, onde k_i é o grau do vértice i. Na sequência, ele define uma matriz esparsa simétrica com elementos ΔQ_{ij}, inicialmente de ordem $n \times n$. Os elementos iniciais dessa matriz são calculados pela equação 5.17.

$$\Delta Q_{ij} = \begin{cases} \frac{1}{2m} - \frac{k_i k_j}{(2m)^2} & \text{, se } i, j \text{ estão conectados} \\ 0 & \text{, caso contrário} \end{cases} \tag{5.17}$$

Depois de calculada a matriz ΔQ e o vetor com os valores de a_i, o algoritmo insere no *max-heap H* o maior elemento de cada linha da matriz ΔQ. Na sequência, é criada uma nova comunidade combinando-se as comunidades i e j com maior valor de ΔQ_{ij} encontrado em H. Para a rede com a nova comunidade, a matriz ΔQ, o *max-heap H* e o vetor a_i são atualizados. Além disso, incrementa-se Q mediante ΔQ_{ij}. A atualização da matriz ΔQ para a união das comunidades i e j formando uma nova comunidade j é realizada pela equação 5.18. Nesse caso considera-se que os membros da comunidade i são inclusos na comunidade j, $C_i \rightarrow C_j$, de modo que só é preciso eliminar a linha e a coluna de i na matriz ΔQ (os outros membros

permanecem inalterados). Os valores da nova comunidade $\Delta Q'_{jk}$ são atualizadas aproveitando a linha e coluna de j.

$$\Delta Q'_{jk} = \begin{cases} \Delta Q_{ik} + \Delta Q_{ik} & \text{, se } k \text{ é conectado a ambos } i \text{ e } j \\ \Delta Q_{ik} - 2a_j a_k & \text{, se } k \text{ é conectado a } i\text{, mas não a } j \\ \Delta Q_{jk} - 2a_i a_k & \text{, se } k \text{ é conectado a } j\text{, mas não a } i \end{cases} \tag{5.18}$$

A equação 5.18 implica que Q tem um único máximo, desse modo, após a ocorrência do maior ΔQ, todos os demais valores de Δ tendem a decrescer (CLAUSET; NEWMAN; MOORE, 2004). O algoritmo irá unir os pares de comunidades e repetir os passos subsequentes do algoritmo até todos os nós pertencerem a mesma comunidade ou até ΔQ começar a decrescer. O algoritmo 3 apresenta o algoritmo *Fast Greedy*.

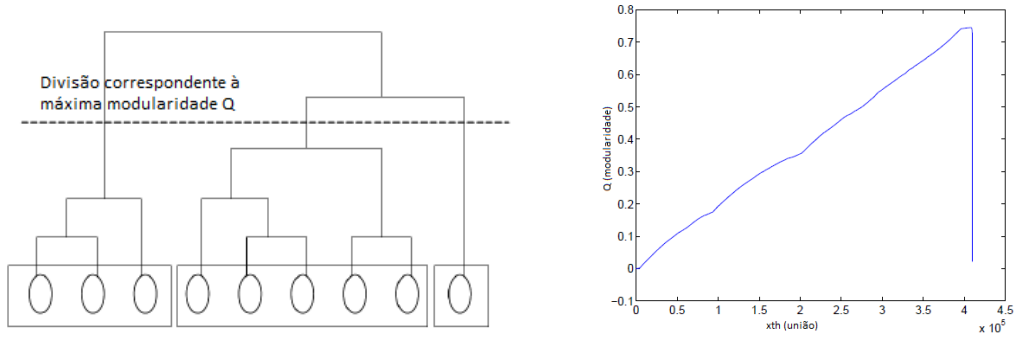
Algoritmo 3: *Fast Greedy*. Adaptado de Clauset, Newman e Moore (2004).

Entrada:
$G = (V, E)$: grafo G, com conjunto de vértices V e arestas E;
Saída:
$V_j, ..., V_k$: k grupos de nós (comunidades);

início
  // Inicialização
  $init(\Delta Q)$; /* inicializa matriz ΔQ $n \times n$ com equação 5.17 */
  $init(a_i)$; /* inicializa vetor a_i */
  $init(H)$; /* inicializa max-heap */
  $C = (C_1, C_2, ..., C_n)$; /* inicializa cada nó em uma comunidade */
fim
repita
  // Iterações
  $\Delta Q_i j = max(H)$; /* encontra maior valor de H */
  $C_i \rightarrow C_j$; /* inclui (uni) a comunidade i na j */
  update(ΔQ) ; /* atualiza matriz ΔQ com equação 5.18 */
  update(H) ; /* atualiza max-heap H */
até $\Delta Q_{ij}^{(t)} > \Delta Q_{ij}^{(t+1)}$ *ou* $C == 1$; /* todos os nós pertencerem a mesma comunidade ou ΔQ_{ij} decrescer */
;
retorna $V_j, ..., V_k$

Na figura 3(a) é apresentado um dendrograma com vértices (círculos) formando comunidades e com o corte do valor máximo de modularidade. Na figura 3(b) é apresentado um gráfico da modularidade ao longo do algoritmo *Fast Greedy*. Ao longo da execução, as comunidades são unidas formando novas comunidades até todos os nós formarem uma única

comunidade. Nesse gráfico, a modularidade máxima é $Q = 0,745$ e ocorre em 1684 comunidades.



(a) (b)

Figura 5.6: (a) Representação do dendrograma com o corte que representa o valor máximo da modularidade e vértices representando comunidades. Adaptada de Raghavan, Albert e Kumara (2007). (b) Gráfico da modularidade pelo número de uniões ao longo do algoritmo *Fast Greedy*, com pico em $Q = 0,745$ e 1684 comunidades. Adaptada de Clauset, Newman e Moore (2004).

5.2.3 *Label Propagation*

Raghavan, Albert e Kumara (2007) propuseram um método iterativo de detecção de comunidades baseado na propagação de rótulos, chamado *Label Propagation* (LP), cuja complexidade computacional é quase linear ($\sim O(n)$). Como já foi dito, esse algoritmo é baseado na propagação de rótulos, de modo que todos os nós recebem inicialmente um rótulo diferente e os rótulos se propagam pela rede, sendo o rótulo de um vértice determinado pelo rótulo com maior frequência entre seus vizinhos. Ao fim do processo, grupos de nós com o mesmo rótulo são agrupados numa mesma comunidade. Assim, se um nó x tem vizinhos $x_1, x_2, ..., x_k$ e cada vizinho tem um rótulo que indica a comunidade a qual ele pertence, a comunidade do nó x é determinada com base na frequência das comunidades dos seus vizinhos. Quando os rótulos se propagam, grupos densamente conectados alcançam em poucas iterações o mesmo rótulo.

A Figura 5.7 exemplifica a propagação de rótulos executada pelo algoritmo e a atualização dos vértices, neste exemplo, a rede é completa (tem densidade máxima), por isso, todos os vértices obtiveram o mesmo rótulo.

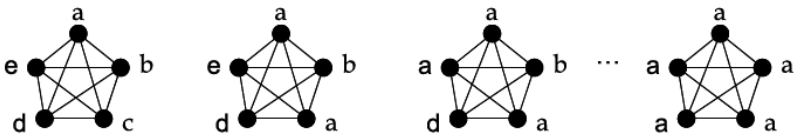


Figura 5.7: Atualização dos vértices no algoritmo *Label Propagation*, um a um, da esquerda para a direita. Devido a alta densidade das arestas (a maior possibilidade neste caso), todos os vértices adquirem o mesmo rótulo. Adaptada de Raghavan, Albert e Kumara (2007).

No *Label Propagation*, a divisão da rede está baseada na propagação dos rótulos dos vértices que é realizada iterativamente. O modo de atualização desses rótulos pode ser síncrono ou assíncrono. A atualização de modo assíncrono surgiu para solucionar o problema de oscilações de rótulos, problema comum em redes bipartidas ou quase bipartidas (RAGHAVAN; ALBERT; KUMARA, 2007). Na atualização síncrona, o vértice x em uma iteração t é função dos rótulos dos seus vizinhos na iteração $(t-1)$. A equação 5.19 apresenta esse modo de atualização, onde $C_x(t)$ é o rótulo do nó x no tempo t.

$$C_x(t) = f(C_{x_1}(t-1), ..., C_{x_k}(t-1)) \tag{5.19}$$

Na atualização assíncrona, por sua vez, o vértice x em uma iteração t é atualizado segundo a equação 5.20, onde $x_{i1}, ..., x_{im}$ são os vizinhos de x cujos rótulos foram atualizados na iteração t (atual), enquanto $x_{i(m+1)}, ..., x_{ik}$ são vizinhos de x cujos rótulos ainda não foram atualizados na iteração t (atual).

$$C_x(t) = f(C_{x_{i1}}(t), ..., C_{x_{im}}(t), C_{x_{i(m+1)}}(t-1), ..., C_{x_{ik}(t-1)}) \tag{5.20}$$

A função f nas equações 5.19 e 5.20 determina o rótulo com maior frequência dos vizinhos de x, e esse rótulo será atribuído ao nó x. Se não há um único rótulo majoritário, um dos rótulos majoritários é selecionado aleatoriamente. A ordem em que os n nós da rede são atualizados em cada iteração é definida aleatoriamente. No início do algoritmo tem-se n rótulos distintos, eles vão se propagando através da rede, onde a maioria dos rótulos desaparece e alguns predominam, de modo que o número de rótulos se reduz ao longo das iterações. Esses rótulos geram as comunidades que compõem a rede.

O algoritmo *Label Propagation* é inicializado ($t = 0$) atribuindo a cada nó um rótulo distinto – matematicamente para um nó x tem-se $C_x = x$. Na sequência, iniciam-se as iterações para cada $t = t + 1$. Em cada iteração, todos os nós são dispostos em uma ordem aleatória X, então o rótulo de cada vértice $x \in X$ (na sequência de X), é atualizado de forma assíncrona

segundo a equação 5.20. São selecionados m vizinhos a serem atualizados na iteração atual (t) e $k-m$ vizinhos cujos rótulos foram atualizados na iteração anterior ($t-1$). O algoritmo segue com as iterações até que todos os vértices apresentam o mesmo rótulo da iteração imediatamente anterior. O algoritmo 4 apresenta o algoritmo *Label Propagation*.

Algoritmo 4: *Label Propagation*. Adaptado de Raghavan, Albert e Kumara (2007).

Entrada:
$G = (V, E)$: grafo G, com conjunto de vértices V e arestas E;
Saída:
$V_j, ..., V_k$: k grupos de nós (comunidades);

início
  // Inicialização
  $i = 0$;
  $t = 0$;
  para cada $x \in V$ **faça**
    $C_x(t) = i$; /* inicializa cada nó com um rótulo distinto */
    $i = i + 1$;
  fim
fim
repita
  // Iterações
  $t = t + 1$;
  $X = random(V)$; /* ordem dos nós nessa iteração */
  para cada $x \in V$ **faça**
    $C_x(t) = f(C_{x_{i1}}(t), ..., C_{x_{im}}(t), C_{x_{i(m+1)}}(t-1), ..., C_{x_{ik}(t-1)})$;
  fim
  // todos os vértices tenha o mesmo rótulo da iteração anterior
até $C_x(t) = C_x(t-1), \forall\, x \in V$;
retorna $V_j, ..., V_k$

A complexidade de tempo de cada iteração é $O(n)$ (n é o número de vértices) e o número de iterações para atingir a convergência parece independente do tamanho do grafo ou cresce muito lentamente com ele, podendo ser considerado aproximadamente constante (RAGHAVAN; ALBERT; KUMARA, 2007).

Embora todos os nós da rede formando apenas uma comunidade satisfaça o critério de parada do algoritmo, uma rede heterogênea com estrutura de comunidade subjacente dificilmente convergirá para a formação uma comunidade única. No caso de redes homogêneas, como as rede aleatórias de Erdös-Renyi (que não apresentam estrutura de comunidade), o algoritmo *Label Propagation* identifica toda a rede como uma única comunidade.

A *Label Propagation* é uma das técnicas mais rápidas de detecção de comunidades e

pode ser utilizada para analisar grandes redes (BOTELHO, 2014; RAGHAVAN; ALBERT; KUMARA, 2007). Um dos problemas dessa técnica é que ela não resulta em uma única solução: cada execução do algoritmo pode apresentar um resultado diferente, mesmo mantendo as condições iniciais inalteradas (RAGHAVAN; ALBERT; KUMARA, 2007). Isso ocorre devido ao fato da ordem de visitação dos vértices ser aleatória no início de cada iteração e da seleção dos rótulos também ser aleatória, em casos que existem mais de um rótulo majoritário entre os vizinhos de um vértice. Portanto, é possível derivar diferentes partições a partir da mesma condição inicial.

5.3 Trabalhos relacionados

Rubinov e Sporns (2010) propõem o uso de redes complexas para a construção de redes cerebrais a partir de dados de conectividade estrutural (anatômica) e funcional. Nele são descritas medidas capazes de detectar a integração funcional e segregação, quantificar a centralidade das regiões cerebrais individuais ou caminhos, caracterizar padrões da anatomia local e testar a robustez das redes a ruídos. Os autores concluíram que a análise de redes complexas é uma importante ferramenta para caracterização anatômica e funcional do cérebro, capaz de extrair propriedades locais e globais do mesmo.

Watanabe (2013) propõe métodos de auxílio ao diagnóstico médico, *Computer-aided diagnosis* (CAD), por imagens usando regras de associação e redes complexas. Em seus métodos, foi utilizada a teoria das redes complexas com a finalidade de modelar imagens médicas em redes livres de escala, de onde são extraídas medidas topológicas para constituir os descritores das imagens. A classificação foi realizada por meio do uso de regras de associação estatísticas. Os seus métodos foram aplicados em um sistema de auxílio ao diagnóstico de mama. Os resultados destes mostraram-se superiores aos métodos do estado da arte da época. Nesta tese, também foram sugeridas novas medidas para caracterizar imagens de pacientes com epilepsia no lobo temporal mesial.

Oliveira (2008) propõe um método para clusterização de dados, baseado na identificação de comunidades em redes complexas e modelos computacionais biologicamente inspirados. Em sua técnica, primeiramente é criada a rede e depois esta é particionada para obtenção dos *clusters*. Os resultados dos seus experimentos mostraram que seu método é capaz de detectar *clusters* de formas arbitrárias e a representá-los com diferentes níveis de refinamento.

Botelho (2014) propõe um método de segmentação para imagens de alta resolução que

combina detecção de comunidades de redes complexas com *superpixels*. Esse método foi aplicado em imagens de ninhais de aves do Pantanal em conjunto com a técnica baseada em *Markov Random Fields* para se realizar a contagem de aves. Os resultados de contagens encontrados mostram-se condizentes com o método manual.

Capítulo 6

Metodologia

Neste trabalho, foi adotada uma metodologia para segmentação de imagem que combina algoritmos de extração de *superpixels* com algoritmos de detecção de comunidades da teoria de redes complexas. O diagrama da figura 6.1 mostra as etapas que constituem esse método. O uso da técnica de *superpixel* tem por objetivo viabilizar o algoritmo de detecção de comunidade, reduzindo seu tempo de processamento e o uso de memória. Essa metodologia foi desenvolvida e avaliada com o uso da linguagem de programação *open source* Python [1] e das bibliotecas: igraph [2] (CSARDI; NEPUSZ, 2006), OpenCV [3] (BRADSKI, 2000) e scikit-image [4] (WALT et al., 2014).

A primeira etapa consiste na geração dos *superpixels* da imagem de entrada. A segunda etapa consiste da extração de característica dos *superpixels* gerados. A terceira etapa consiste na criação da rede complexa, onde cada *superpixel* será um nó, e as arestas que ligam os nós são construídas por meio da características extraídas dos *superpixels*. A quarta, e última, etapa consiste em aplicar os algoritmos de detecção de comunidades na rede complexas gerada, para gerar a segmentação final da imagem. Nas seções 6.1, 6.2, 6.3 e 6.4 essas etapas são detalhadas.

Para avaliar essa metodologia, utilizou-se a base de dados *Berkeley Segmentation Dataset* 300 (BSDS300) (MARTIN et al., 2001), pelo extenso repertório de publicações que a utilizam (BOTELHO, 2014; CUADROS, 2013; BELIZARIO, 2012). Essa base é constituída por 300 imagens naturais com os respetivos *ground-truth*, obtidos pela segmentação manual de, no mínimo, 5 indivíduos para cada imagem. Na seção 6.5 é detalhado o processo de avaliação

[1] https://www.python.org/
[2] http://www.igraph.org/
[3] http://opencv.org/
[4] http://scikit-image.org/

da segmentação final.

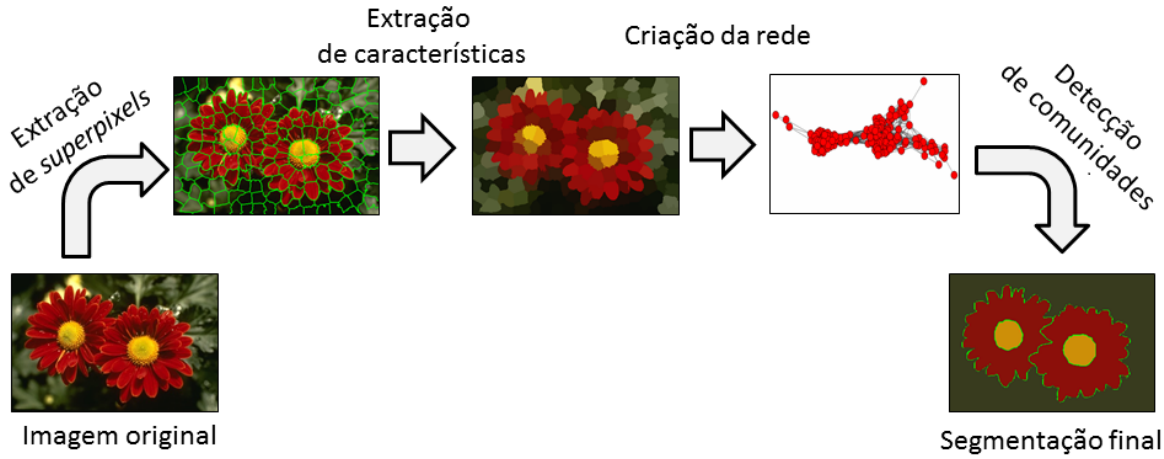


Figura 6.1: Etapas do método de segmentação de imagens proposto que combina extração de *superpixels* e detecção de comunidades de redes complexas.

6.1 Extração de *superpixels*

A extração de *superpixels* para cada imagem foi obtida com a aplicação do algoritmo SLIC proposto por Achanta et al. (2010, 2012), detalhado na seção 4.1 e descrito pelo algoritmo 1. Sua aplicação foi realizada com o uso da função *slic* do módulo *segmentation* [5], contido na biblioteca scikit-image (WALT et al., 2014). A escolha do algoritmo SLIC se deve aos bons resultados obtidos por este na literatura (BELIZARIO, 2012; SARATH, 2014; WANG et al., 2011) e à sua uniformidade, característica importante para a fase de criação da rede complexa.

Os parâmetros de entrada do algoritmo SLIC são o número de *superpixels* (k), a constante de compacidade (m) e o número máximo de iterações. A função *slic* altera o parâmetro m para um equivalente denominado com *compactness*. Além disso, esta função provê um parâmetro de modo do algoritmo SLIC. Este parâmetro é booleano, de modo que verdadeiro significa modo SLIC0 e falso modo SLIC. O modo SLIC0, também chamado de ASLIC (ACHANTA et al., 2012), constitui o mesmo algoritmo de SLIC, porém com compacidade adaptativa, de modo que o parâmetro *compactness* para este modo é apenas o valor inicial deste.

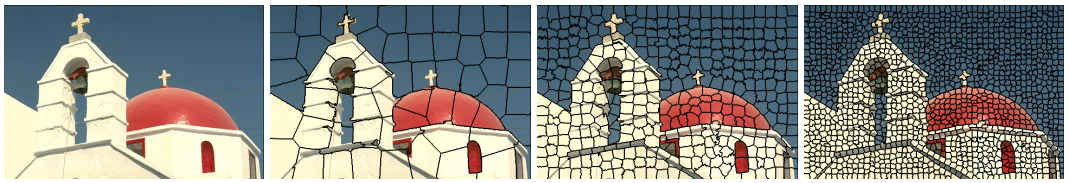
O parâmetro número de *superpixels* foi alterado para tamanho (comprimento do lado) do *superpixels* (S), com o objetivo de torná-lo independente do tamanho da imagem de entrada. A equação 6.1 foi utilizada para realizar essa alteração – nela N é o número total de *pixels* da

[5] http://www.scikit-image.org/docs/dev/api/skimage.segmentation.html

imagem.

$$k = \frac{N}{S^2} \tag{6.1}$$

Como saída, o algoritmo SLIC apresenta uma pré-segmentação da imagem de entrada, representada em pequenos segmentos denominados *superpixels*. A Figura 6.2 apresenta um exemplo do resultado da pré-segmentação do algoritmo SLIC para tamanhos distintos, 50, 20 e 10, respectivamente.



(a) (b) (c) (d)

Figura 6.2: (a) Original, (b) SLIC ($m = 0,01$ $S = 50$) (c) SLIC ($m = 0,01$ $S = 20$) (d) SLIC ($m = 0,01$ $S = 10$)

Na figura 6.2, é possível notar que a pré-segmentação fornecida pelos *superpixels* é uma super-segmentação da imagem que, em alguns casos, pode ser suficiente. Os *superpixels* são mais adequados para identificar objetos com, pelo menos, metade do seu tamanho. Objetos muito menores passam despercebidos, como aconteceu com a cruz da catedral para $S = 50$ na figura 6.2(b).

Na metodologia proposta, os *superpixels* têm como finalidade representar de forma eficiente e compacta a imagem, permitindo a geração de uma rede compacta (com redução de nós). Desse modo, a qualidade do *superpixel* para essa aplicação é definida por sua regularidade e pela aderência às bordas dos objetos contidos na imagem. Assim sendo, quanto menor o tamanho do *superpixel*, maior sua aderência as bordas, porém maior também o número de nós da rede. O parâmetro de compacidade tem grande influência na aderência as bordas do algoritmo SLIC. Dessa maneira, foi realizado um experimento para avaliar os parâmetros do algoritmo SLIC, buscando a extração de *superpixel* mais adequada para a segmentação final. Vale ressaltar que a qualidade dos superpixels gerados influencia diretamente na qualidade da segmentação final da imagem.

6.2 Extração de características dos *superpixels*

Na sequência da metodologia proposta, após o cálculo dos *superpixels*, suas características são extraídas. As características extraídas de cada *superpixel* neste trabalho foram o centroide e a intensidade média de cada um dos seus canais de cor. Dois modelos de cor foram testados para compor esta etapa, RGB e CIELAB. A escolha destes modelos se deve ao fato da imagem de entrada utilizar o modelo RGB, enquanto que o algoritmo SLIC utiliza o modelo CIELAB. Um experimento foi realizado para analisar qual a influência destes modelos de cor na qualidade da segmentação final, em busca de qual seria mais adequado para o método proposto.

A equação 6.2 computa o centroide (x,y) do *superpixel SP* composto por N_{SP} *pixels*. E a equação 6.3 computa a média do seu vetor tridimensional de cor (c_1,c_2,c_3).

$$(x,y)_{SP} = \frac{1}{N_{SP}} \sum_{p \in SP} (x(p), y(p)) \tag{6.2}$$

$$(c_1,c_2,c_3)_{SP} = \frac{1}{N_{SP}} \sum_{p \in SP} (c_1(p), c_2(p), c_3(p)) \tag{6.3}$$

6.3 Geração da rede

Na sequência da metodologia proposta, uma rede é criada a partir das características extraídas dos *superpixels*. Nesta etapa, é construída uma rede ponderada não-orientada, com nós formados pelos *superpixels* calculados e arestas com pesos computados por funções de similaridade de suas características. A criação da rede é um dos processos mais importantes para o método de segmentação proposto, devido a sua grande influência nos resultados finais da segmentação.

As funções utilizadas para calcular a similaridade têm como entradas a distância espacial (d) e a distância euclidiana normalizada do vetor de cores tridimensional (I), cuja equação se encontra em 6.4. O cálculo de I para os *superpixels* i e j é realizado com os canais de cores normalizados no intervalo entre $[0,1]$, para que I também esteja normalizado.

$$I(i,j) = \sqrt{\frac{(c_{1i}-c_{1j})^2 + (c_{2i}-c_{2j})^2 + (c_{3i}-c_{3j})^2}{3}} \tag{6.4}$$

Para o calculo da distância espacial (entre os centroides) foram testadas três funções de

distância: *Euclidean*, Manhattan e Chebyshev, cujas equações estão apresentadas, respectivamente, em 6.5, 6.6 e 6.7. Essas distância são normalizadas pelo tamanho inicial dos *superpixels* (S), para que o raio seja mensurado em unidades de *superpixels*.

$$dEU(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{6.5}$$

$$dMA(i,j) = |x_i - x_j| + |y_i - y_j| \tag{6.6}$$

$$dCH(i,j) = \max(|x_i - x_j|, |y_i - y_j|) \tag{6.7}$$

Para estabelecer as conexões entre os vértices da rede, são empregados dois parâmetros: *threshold* (t) e raio (Rp). Haverá uma aresta entre dois vértices apenas se o valor computado para $(1-I)$ entre os *superpixels* for maior que t e a sua distância espacial menor que Rp. O parâmetro *threshold* evita a existência de arestas entre nós com baixa similaridade de cor. O raio, por sua vez, é utilizado para evitar conexões entre *superpixels* com centroides muito distantes. Desse modo, os nós da rede só fazem conexões com nós dentro de uma região aproximadamente circular de raio R. A escolha do raio é afetada pelo tamanho da imagem e tamanho dos *superpixels*.

Neste trabalho, foram avaliadas três funções de similaridade (peso): WG, WE e WI, respectivamente equacionadas em 6.8, 6.9 e 6.10. A função peso dada pela equação 6.8 é baseada na função de similaridade Gaussiana, muito utilizada pela literatura de agrupamento em grafos para segmentação de imagens (SHI; MALIK, 2000; CIGLA; ALATAN, 2010; TUNG; WONG; CLAUSI, 2010; SAKAI; IMIYA, 2009; YAN; HUANG; JORDAN, 2009). A função peso 6.9 é uma exponencial obtida empiricamente pela autora. A função peso 6.10 desconsidera a distância espacial (d) entre os *superpixels*, desse modo, ela deve ser utilizada com raios pequenos para evitar que regiões distantes com a mesma cor sejam agrupadas, ou ainda contornar esse problema com o uso de um algoritmo para separar regiões não conectadas da segmentação encontrada pela detecção de comunidades da rede. Para a função peso WG, os valores dos parâmetros σ_d e σ_I são valores entre 10% e 20% de toda a faixa de distância do d e I, respectivamente (SHI; MALIK, 2000). Assim sendo, utilizou-se $\sigma_I = 0,15$ e $\sigma_d = 2,5$, que são respectivamente, 15% da faixa de $1-I$ e 10% da faixa de raio.

$$WG(i,j) = e^{-\frac{I^2}{\sigma_I^2} - \frac{d^2}{\sigma_d^2}} \tag{6.8}$$

$$WE(i,j) = (1-I)\frac{1}{d} \tag{6.9}$$

$$WI(i,j) = 1 - I \tag{6.10}$$

6.4 Detecção de Comunidades

Após a construção da rede, é aplicado um algoritmo de detecção de comunidades para segmentar a imagem, *i.e.*: agrupar os *superpixels* calculados. Dois algoritmos de detecção de comunidades foram avaliados: *Fast Greedy* (FG) e *Label Propagation* (LP), descritos nas seções 5.2.2 e 5.2.3, respectivamente. A escolha do *Fast Greedy* se deve ao fato de seu custo computacional ser menor que outros algoritmos de detecção de comunidades baseados na modularidade, e a escolha do *Label Propagation* deve-se ao fato de seu custo computacional ser baixo e aproximadamente linear (BOTELHO, 2014).

A aplicação dos algoritmos *Fast Greedy* e *Label Propagation* foi realizada em Python, com o uso da biblioteca igraph (CSARDI; NEPUSZ, 2006). Esses algoritmos apresentam como saída a divisão dos vértices da rede em comunidades, em outras palavras, o agrupamento dos nós da rede, portanto dos *superpixels*. Cada uma das comunidades representa as diferentes regiões segmentadas da imagem. Em alguns casos, regiões não conectadas são consideradas pertencentes à mesma segmentação. Este problema pode ser resolvido aplicando algum algoritmo para separar regiões não conectadas como pós-processamento da segmentação encontrada pela detecção de comunidades da rede – o resultado deste é a segmentação final.

O resultado da detecção de comunidades é muito dependente da rede gerada. Como os algoritmos FG e LP utilizam abordagens diferentes para a detecção de comunidades, as rede ótimas para cada algoritmo podem ser distintas. Desse modo, os experimentos realizados buscam os melhores parâmetros para cada um dos algoritmos de detecção de comunidades.

A partir do resultado da detecção de comunidades, avaliou-se a aplicação de um pós-processamento para separar regiões não conectadas. Sua aplicação foi realizada em Python com o uso da função *label* do módulo *measure* [6], contido na biblioteca scikit-image (WALT et al., 2014). Portanto, o resultado da metodologia proposta para segmentação de imagens,

[6] http://www.scikit-image.org/docs/dev/api/skimage.measure.html

que utiliza *superpixels* e redes complexas, é o resultado dos algoritmos de detecção de comunidades sobre a rede construída com os *superpixels* da imagem, caso não haja o pós-processamento, e, caso haja, é o resultado deste.

6.5 Avaliação dos resultados

Após encontrar os resultados da segmentação da imagem para a metodologia proposta, é realizado o processo de avaliação dos resultados obtidos. São avaliadas a qualidade da segmentação, tanto qualitativa quanto quantitativamente, e o tempo de processamento para cada etapa do algoritmo.

A segmentação de imagens é uma tarefa subjetiva, dado que o ponto de vista dos usuários é muito variado. Desse modo, avaliar os resultados é também um processo complicado e subjetivo, afinal uma imagem pode ter diversos resultados "verdadeiros" e uma mesma segmentação pode ser considerada de qualidade boa ou ruim, a depender do usuário.

Para reduzir a subjetividade na avaliação qualitativa de segmentação, foram desenvolvidas técnicas de avaliação baseada em *ground-truth*. O *ground-truth* de uma imagem é um conjunto de imagens segmentadas manualmente por diferentes usuários, utilizado como referência para a comparação com a segmentação automática.

Arbelaez et al. (2009) propôs uma métrica baseada em *ground-truth* para medir quantitativamente a qualidade de segmentações de uma imagem. Esse métrica calcula a cobertura (*covering*) entre duas imagens segmentadas, *i.e.*: a sua similaridade. A métrica de *covering* é bastante conhecida e utilizada na literatura, sendo sugerida pelo grupo de visão computacional da Universidade de Berkeley da Califórnia, que disponibiliza uma base de dados de imagens com seus respectivos *ground-truth*.

A equação 6.11 computa a qualidade da segmentação baseada no método de *covering* (cobertura de regiões), proposto por Arbelaez et al. (2009), entre as segmentações S' e S, onde as regiões de S' tentam cobrir as regiões de S. Assim sendo, S' representa a segmentação automática a ser avaliada e S uma segmentação manual. É importante ressaltar que o *covering* não apresenta a propriedade de simetria, portanto $\mathcal{C}(S' \rightarrow S) \neq \mathcal{C}(S \rightarrow S')$.

$$\mathcal{C}(S' \rightarrow S) = \frac{1}{N} \sum_{R \in S} |R| \max_{R' \in S'} \{\mathcal{O}(R, R')\} \tag{6.11}$$

Nessa equação, N é o número total de *pixels* da imagem, $|R|$ é o número de *pixels* da região R e a função $\mathcal{O}(R, R')$, dada pela equação 6.12, corresponde ao cálculo da sobreposição

(*overlap*) entre as regiões R e R', proposta por Ge, Wang e Liu (2006). Os resultados do *covering* pertencem ao intervalo $[0,1]$, de modo que, quanto maior o resultado, melhor a qualidade da segmentação e segmentações iguais apresentam $\mathcal{C} = 1$.

$$\mathcal{O}(R,R') = \frac{|R \cap R'|}{|R \cup R'|} = \frac{|R \cap R'|}{|R| + |R'| - |R \cap R'|} \tag{6.12}$$

Como o *ground-truth* de uma imagem pode conter mais de uma segmentação manual, Arbelaez et al. (2009) propôs que fosse calculada a média do *covering* entre a segmentação automática (S') e cada segmentação manual, pertencente ao seu *ground-truth* (G). A equação 6.13, onde $|G|$ é o número de segmentações manuais de uma imagem, computa essa média e é utilizada para calcular a qualidade da segmentação encontrada para uma imagem.

$$A_c = \frac{1}{|G|} \sum_{S \in G} \mathcal{C}(S' \rightarrow S) \tag{6.13}$$

Portanto, a qualidade de um algoritmo de segmentação automático pode ser quantificado pela equação 6.14, que calcula a média da qualidade da segmentação para um conjunto de imagens com seu respectivo *ground-truth*.

$$A_M = \frac{1}{N} \sum_{i=1}^{N} A_{ci} \tag{6.14}$$

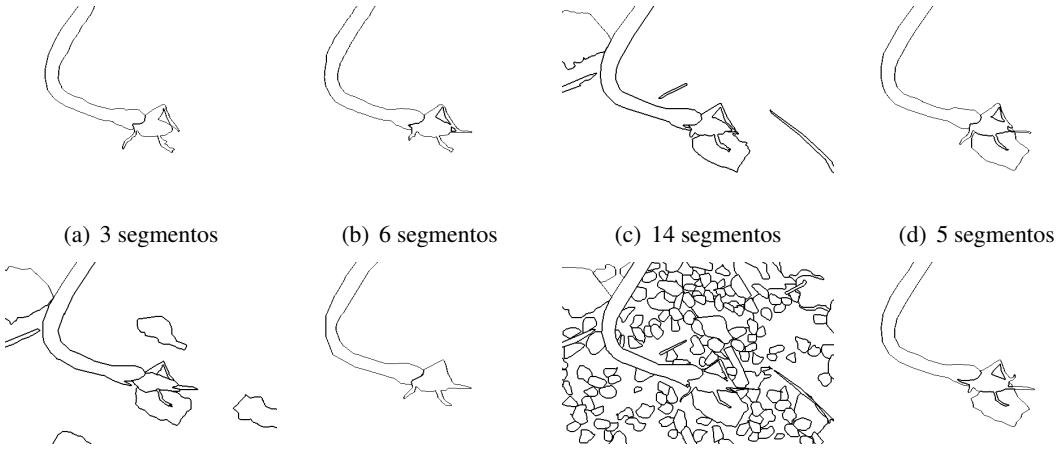
Nesta equação, N é o número de imagens utilizada para calcular a qualidade do método de segmentação e A_{ci} é a medida de qualidade da segmentação automática encontrada para uma imagem e seu respectivo *ground-truth*.

Para avaliar a metodologia proposta, foi utilizada a base de imagens BSDS300 pertencente ao banco de dados *Berkeley Image Dataset* (MARTIN et al., 2001), disponibilizadas pelo grupo de Visão Computacional da Universidade de Berkeley da Califórnia.

A BSDS300 é constituída por 300 imagens naturais de dimensões 481×321 no formato JPG, com os respectivos *ground-truth*, obtidos pela segmentação manual de, no mínimo, 5 indivíduos distintos para cada imagem. Essa base de dados está dividida em um conjunto de 200 imagens para treinamento e as outras 100 destinadas a testes.

Figura 6.3: Imagem 175043 do banco de imagens testes BSDS300

A figura 6.3 contém uma imagem de teste da base de imagens BSDS300 e a figura 6.4 contém o seu *ground-truth*, composto pela segmentação manual de 8 usuários distintos. É possível notar a subjetividade da segmentação ao observar a figura 6.4 e notar a grande variação do número de segmentos entre as diferentes segmentações manuais. A métrica de qualidade utilizada calcula a média do *covering* entre a segmentação automática e cada uma destas segmentações, por isso o resultado $Ac = 1$ só será encontrado quando todas as segmentações manuais forem iguais entre si e, simultaneamente, a segmentação automática for igual às manuais. Do mesmo modo, quanto mais distintas forem as segmentações manuais, mais baixo será o valor de Ac, e quanto mais próximas, maior será o seu valor. Portanto, o Ac é bastante dependente da similaridade entre as segmentações manuais contidas em seu *ground-truth*.



(a) 3 segmentos (b) 6 segmentos (c) 14 segmentos (d) 5 segmentos

(e) 15 segmentos (f) 4 segmentos (g) 208 segmentos (h) 6 segmentos

Figura 6.4: *Ground-truth* composto por 8 segmentações da imagem 175043 do banco de imagem BSDS300, com suas respectivas quantidades de segmentos.

Para efeito de medição de tempo, todos os experimentos realizados neste trabalho foram executadas em um computador Intel Core i7-5930K 3.50 GHz, RAM de 32GB e sistema operacional Windows 10 64 bits. Todos os algoritmos e experimentos foram implementados na linguagem de programação Python.

Capítulo 7

Resultados

Este capítulo contém a discussão dos experimentos realizados no método de segmentação proposto nesta monografia, que combina extração de *superpixels* com algoritmos de detecção de comunidades. Os experimentos realizados têm por objetivo avaliar a qualidade e o custo computacional do método proposto. Para atingir esse objetivo, os experimentos analisam a influência dos parâmetros envolvidos em busca da sua melhor combinação. E, por fim, realizam uma avaliação qualitativa e quantitativa dos resultados obtidos em comparação com uma técnica semelhante, denominada *Efficient Graph-Based Image Segmentation* (EGB) proposta por Felzenszwalb e Huttenlocher (2004).

Ao todo foram realizados quatro experimentos. O primeiro experimento se foca na análise qualitativa dos *superpixels* gerados. O segundo experimento analisa os parâmetros de maior influência na qualidade e tempo. O terceiro experimento analisa a influência dos parâmetros restantes para se definir a melhor combinação destes. O quarto experimento analisa a qualidade do método proposto em comparação ao método EGB (FELZENSZWALB; HUTTENLOCHER, 2004).

7.1 Análise dos *Superpixels*

Como mencionado no capítulo 6, optou-se pelo algoritmo SLIC para a extração de *superpixels*, devido aos bons resultados que este apresenta na literatura e à sua uniformidade, característica importante para a fase de criação da rede complexa. A implementação do algoritmo SLIC foi realizada em Python com o uso do módulo *segmentation* da biblioteca scikit-image. Este módulo contém a função *slic* que aplica o algoritmo SLIC retornando os *superpixels* calculados para a imagem de entrada, de acordo com os parâmetros de entrada

fornecidos.

Este experimento analisa de modo qualitativo a influência dos parâmetros do algoritmo SLIC na qualidade dos *superpixels* gerados. Os parâmetros da função *slic* utilizados nessa análise foram: número de segmentos, *compactness* ($cpact$), número máximo de iterações ($imax$) e modo. Como discorrido na seção 6.1, foi utilizado o tamanho do *superpixel* (S) em substituição ao número de segmentos, com a finalidade de evitar dependência do tamanho da imagem. O modo é um parâmetro *booleano*, onde verdadeiro significa modo SLIC0 e falso, modo SLIC.

O principal critério utilizado na análise qualitativa dos *superpixels* foi sua aderência às bordas e sua regularidade. O custo computacional de tempo de processamento, por sua vez, foi analisado quantitativamente e mensurado em segundos de processamento. A análise dos parâmetros tem por objetivo encontrar a melhor relação entre qualidade e custo computacional para o processo de extração de *superpixels*.

7.1.1 Modo: SLIC ou SLIC0

Primeiramente foi avaliado qual o melhor modo, SLIC ou SLIC0, para gerar uma rede complexa. Como ambos são dependentes do parâmetro *compactness*, a análise qualitativa da extração de *superpixels* para ambos modos foi realizada para valores diferentes de *compactness*. Essa extração foi realizada para 50 imagens de treinamento do banco de dados BSDS300 e para *compactness* variando de 10^{-8} a 10^{2} em potências de 10, pois sua escala é logarítmica. As 50 imagens foram escolhidas aleatoriamente e não foi utilizada o total de imagens de treinamento devido ao elevado tempo de processamento do experimento.

A análise qualitativa desta etapa do experimento mostrou que o modo SLIC0 apresentou resultados consideravelmente melhores que o SLIC, como pode ser visto na figura 7.1. Nesta figura, está representado o resultado qualitativo de apenas uma imagem da comparação entre o modo SLIC e SLIC0 para $S = 30$, $imax = 10$ e valores $cpact$ que apresentaram melhores resultados para o modo SLIC. A terceira linha da figura contém o melhor resultado encontrado para o modo SLIC, que ocorre em $cpact = 10$. É possível notar que o modo SLIC0 para o mesmo valor de *compactness* apresentou melhor qualidade na extração de *superpixels*.

O parâmetro *compactness* representa a compacidade, fazendo o balanço entre proximidade de cor e proximidade espacial. Desse modo, quanto menor o seu valor maior a aderência às bordas, porém menor a regularidade dos *superpixels*. Assim sendo, para valores muito pequenos a região de alcance no modo SLIC aumenta, consequentemente ele toma formas

geométricas bastante irregulares e não consegue encontrar as bordas dentro das 10 iterações e, como consequência, apresentou baixíssima qualidade para *compactness* menores que 10. O modo SLIC0, por sua vez, apresentou bons resultados qualitativos para uma faixa de valores de $cpact$, pelo fato deste ser adaptativo. Ele perdeu qualidade apenas para valores altos de $cpact$, acima de 10, causada pela perda de aderência às bordas.

Devido à grande qualidade do modo SLIC0 em comparação com o SLIC, o modo SLIC0 foi escolhido para os próximos experimentos. Na seção a seguir, o parâmetro *compactness* é analisado mais detalhadamente.

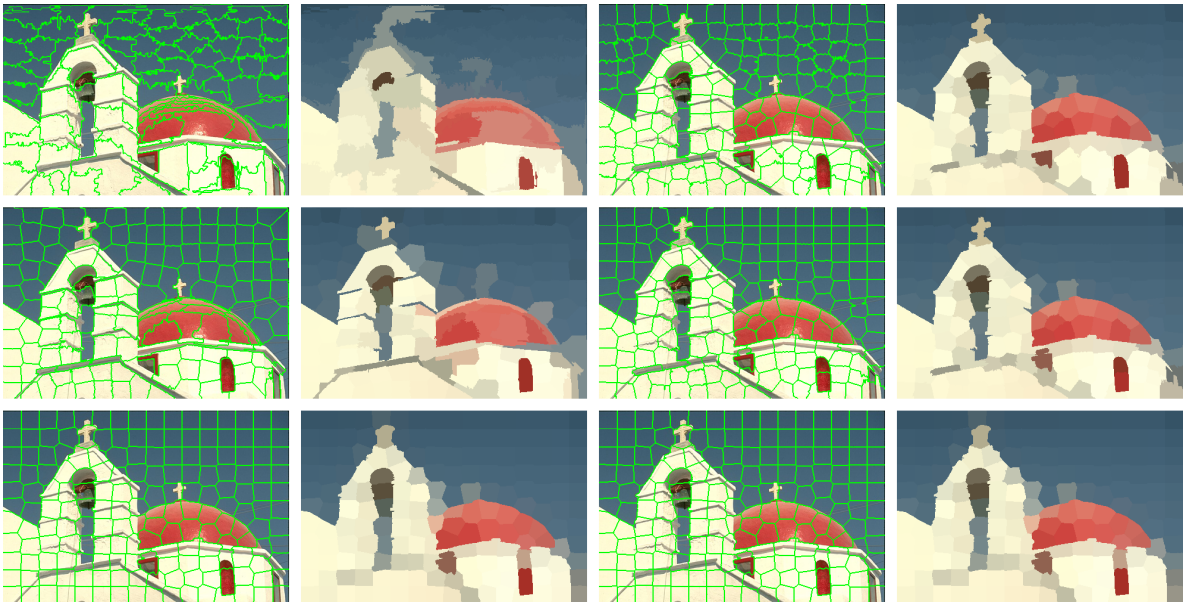


Figura 7.1: Cada linha contém o resultados da extração de *superpixels* do algoritmo SLIC com os mesmo valores de S, $imax$ e $cpact$. As imagens das primeira e terceira colunas contém as representações dos *superpixels*, enquanto que as segunda e quarta colunas mostram a imagem gerada com a média de cores de cada *superpixel*. A primeira e segunda coluna são segmentações do modo SLIC, enquanto que as terceira e quarta coluna são segmentações do modo SLIC0. Para todas as imagens foi fixado $S = 30$, $imax = 10$. Para a linha 1,2 e 3 tem-se, respectivamente, $cpact = 1, 10$ e 100.

7.1.2 Parâmetro: *compactness* ($cpact$)

O parâmetro *compactness* controla o peso dado à proximidade no espaço de cor e à proximidade espacial. Assim sendo, ela relaciona a regularidade da forma geométrica assumida pelos *superpixels* e sua aderência às bordas. Portanto, altos valores de *compactness* atribuem maior peso à proximidade espacial, resultando em *superpixels* com formatos mais quadrados.

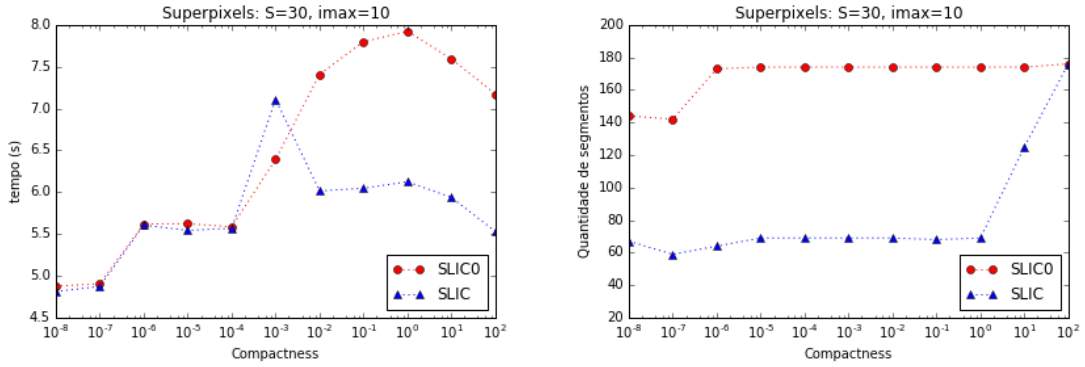
Com a finalidade de avaliar a influência que o parâmetro $cpact$ exerce na qualidade e no custo computacional da extração de *superpixels*, verificou-se a alteração no comportamento dos *superpixels* ao se variar este parâmetro, mantendo os outros constantes. O tamanho do *superpixels* foi fixado em $S = 30$, o número de iterações em $imax = 10$ e o *compactness*

variou de 10^{-8} a 10^{2} em potências de 10, pois sua escala é logarítmica. Os gráficos contidos na figura 7.2 contêm os resultados encontrados da média de 50 imagens de treinamento do banco de dados BSDS300 para os modos SLIC0 e SLIC. As 50 imagens foram escolhidas aleatoriamente e não foi utilizada o total de imagens de treinamento devido ao elevado tempo de processamento do experimento.

O gráfico da figura 7.2(a) contém a curva de $tempo \times compactness$ para ambos os modos. Observando-a é possível perceber que, de modo geral, o SLIC é mais rápido que o SLIC0, sendo esta diferença bastante expressiva para valores de $cpact$ maiores que 10^{-2}, porém para valores menores que 10^{-3} essa diferença é pouco expressiva.

O gráfico da figura 7.2(b) contém a curva de *quantidade de segmentos* $\times$ *compactness* para ambos os modos. Observando-a, é possível perceber que, no modo SLIC0, para valores acima de 10^{-6} o número de segmentos é aproximadamente constante e coerente com a grade inicial que, para $S = 30$ e imagens 481×321, é composta por 171 segmentos. Enquanto que para o modo SLIC, valores de *compactness* menores que 10^{0} apresentam variação de mais de 50% da grade inicial.

Como mostrada na figura 7.1, o modo SLIC0 apresentou melhor qualidade que o SLIC. Além disso, pelo fato do modo SLIC0 ser adaptativo, ele apresenta bons resultados para uma grande faixa de *cpact*, o que não ocorre com o modo SLIC.



(a) (b)

Figura 7.2: Gráficos da influência do parâmetro *compactness* no tempo e a quantidade de segmentos para os modos SLIC0 e SLIC.

A figura 7.3 contém os resultados qualitativos da variação do *compactness* para o modo SLIC0. Nela é possível notar que a extração de *superpixels* para valores de *cpact* na faixa de 10^{-5} a 10^{0} é pequena. Além disso, nota-se que, para *cpac* 10^{1} e 10^{2}, os *superpixels* têm forma notavelmente mais quadrada, sendo que em $cpact = 100$ muitas bordas já são

perdidas. Para valores de *cpact* menores que 10^{-7}, as bordas da cruz maior começam a ser perdidas, lembrando que elas foram muito bem detectadas para $cpact = 10^{-3}$, por exemplo. A diferença entre os valores 10^{-7} e 10^{-8} também foi percebida pelo gráfico da figura 7.2(b), pois há uma queda significativa no número de segmentos comparado ao da grade inicial.

Unindo os resultados das figuras 7.2 e 7.3, conclui-se que o valor de *compactness* com melhor custo benefício entre tempo e qualidade é $cpact = 10^{-4}$. Lembrando que esse valor é o melhor apenas para o modo SLIC0.

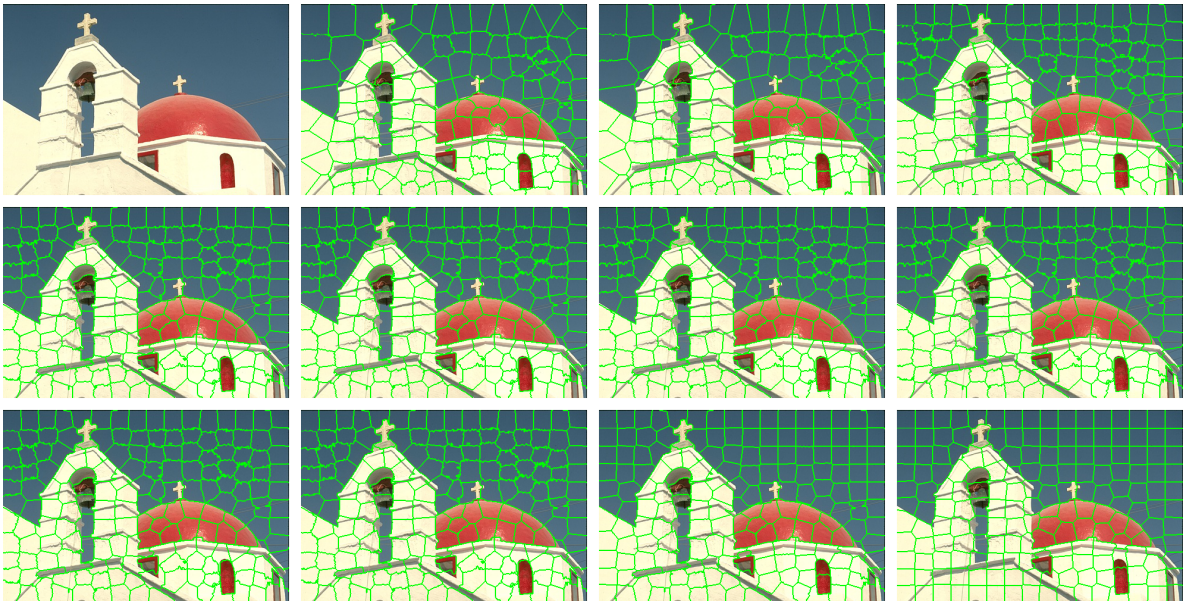


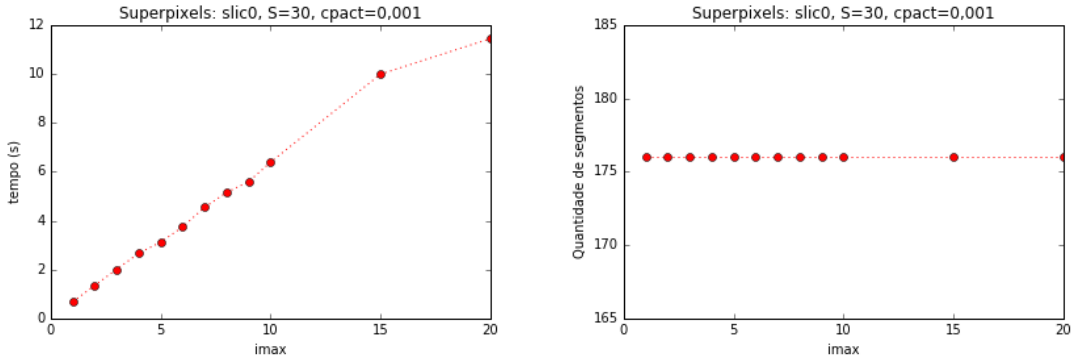
Figura 7.3: Resultado qualitativo do SLIC0 variando o *compactness*. A primeira imagem é a original e da segunda até a última imagem, os resultados qualitativos das extrações de *superpixels* no modo SLIC0 variando os valores de *cpact*, respectivamente, de 10^{-8} a 10^{2} em potências de 10.

7.1.3 Parâmetro: número máximo de iterações (*imax*)

Com a finalidade de avaliar a influência que o parâmetro número máximo de iterações, *imax*, exerce na qualidade e no custo computacional da extração de *superpixels*, verificou-se a alteração no comportamento dos *superpixels* ao se variar este parâmetro e mantendo os outros constantes. O tamanho dos *superpixels* foi fixado em $S = 30$, o *compactness* em $cpact = 0,001$ e o número de iterações foi variado na faixa de 1 a 50. Os gráficos na figura 7.4 contêm os resultados encontrados da média de 50 imagens de treinamento do banco de dados BSDS300 para o modo SLIC0. As 50 imagens foram escolhidas aleatoriamente e não foi utilizada o total de imagens de treinamento devido ao elevado tempo de processamento do experimento.

O gráfico da figura 7.4(a) contém a curva de $tempo \times imax$ para o modo SLIC0. Nela,

é possível perceber a relação aproximadamente linear entre o *imax* e tempo, onde o tempo aumenta com o crescimento de *imax*. O gráfico da figura 7.4(b) contém a curva de *quantidade de segmentos* $\times$ *imax* para o modo SLIC0, onde é possível perceber que o número de segmentos permanece praticamente constante a variações de *imax*.



(a) (b)

Figura 7.4: Gráficos da influência do parâmetro número máximo de iterações no tempo e na quantidade de segmentos para o modo SLIC0.

Na figura 7.5 há o resultado qualitativo para uma imagem da extração de *superpixels* no modo SLIC0 variando a quantidade máxima de iterações. A primeira imagem desta figura representa a iteração 1, portanto é a grade inicial. Para *imax* a partir de 5 obtém-se bons resultados qualitativos.

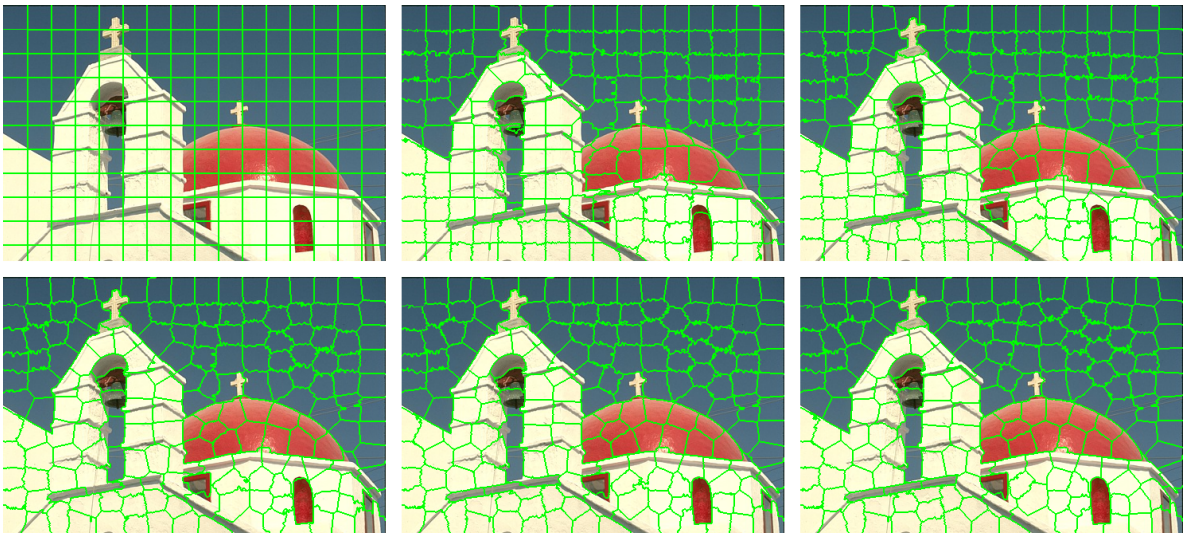


Figura 7.5: Resultado qualitativo para a extração de *superpixels* no modo SLIC0 variando a quantidade máxima de iterações, respectivamente, em $1, 2, 5, 10, 15, 20$.

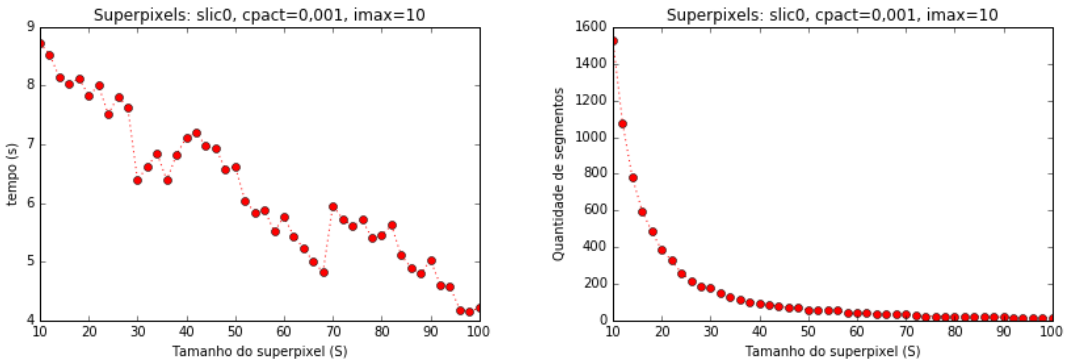
Os resultados do artigo Achanta et al. (2012) apontam que 10 iterações é o suficiente para se alcançar as bordas das imagens. Desse modo, é possível concluir que a faixa de *imax* entre

5 e 10 apresenta bons resultados de qualidade e custo computacional, sendo que, quanto maior o *imax*, maior o tempo, porém menor a possibilidade de perda de bordas. Portanto, para evitar perda de qualidade com tempo razoável de processamento adotou-se $imax = 6$.

7.1.4 Parâmetro: tamanho do *superpixel* (S)

O parâmetro tamanho do *superpixel* é de extrema importância para as etapas de geração do grafo e detecção de comunidades do método segmentação proposto. A relação entre eles se deve ao fato de que o tamanho do *superpixel* limita o tamanho dos objetos que ele consegue detectar. Assim sendo, quanto menor o tamanho dos *superpixels* mais objetos eles são capazes de distinguir, porém maior é o número de nós da rede e, por conseguinte, o custo computacional total do método.

Superpixels com lado de tamanho menor que 10 são inviáveis para a técnica proposta, pois fornecem mais de 1500 segmentos para imagens 321×481, de modo que a rede gerada torna-se muito grande, inviabilizando o custo computacional dos algoritmos de detecção de comunidades, principalmente considerando a quantidade de memória RAM utilizada. Para avaliar a influência de S na qualidade e custo computacional, gerou-se os gráficos da figura 7.6, que contêm, respectivamente, os dados de tempo e quantidade de segmentos para o modo SLIC0 variando-se o tamanho dos *superpixels* de 10 a 100 com passo 2.



(a) (b)

Figura 7.6: Gráficos da influência do parâmetro tamanho do *superpixel* no tempo e na quantidade de segmentos para o modo SLIC0.

Analisando o gráfico da figura 7.6(b), percebe-se que a quantidade de segmentos decresce exponencialmente com o aumento do tamanho do *superpixel*. Para $S = 16$ tem-se em torno de 600 segmentos – a partir deste valor, tem-se um número de nós razoável para a rede. Analisando o gráfico da figura 7.6(a), percebe-se que o tempo decresce com o aumento do

tamanho dos *superpixels*, de modo que, quanto maior S, melhor o desempenho em termos de custo computacional, entretanto, menor a qualidade dos *superpixels*. As variações que ocorrem na curva de *tempo* $\times$ *tamanho dos superpixels* se devem à dependência do algoritmo SLIC com a geometria espacial da imagem, ou seja, a localização inicial dos *superpixels* altera o seu tempo de convergência e a sua qualidade.

A figura 7.7 contém os resultados qualitativos da variação do tamanho do *superpixel* para duas imagens. As primeira linha dessa imagem contém as imagens originais, as demais são a representação das bordas dos *superpixels* com sua respectiva intensidade média de cor. Nela, é possível observar a perda de qualidade com o aumento de S, pois as bordas e objetos da imagem são cada vez menos detectados. A perda de objetos se refletem em erros na intensidade média dos *superpixels*. Como exemplo, na imagem da igreja da penúltima linha ($S = 80$) da figura 7.7, a perda da janela vermelha produz um *superpixel* com intensidade média de cor rosa, devido a mistura da cor vermelha da janela e branca da parede da igreja.

Conclui-se das figuras 7.7 e 7.6 que a faixa de tamanho de *superpixel* que melhor equilibra a qualidade e o custo computacional é de 20 a 50, de modo que, para imagens de 321×481, a melhor quantidade de *superpixels* está entre 400 e 600.

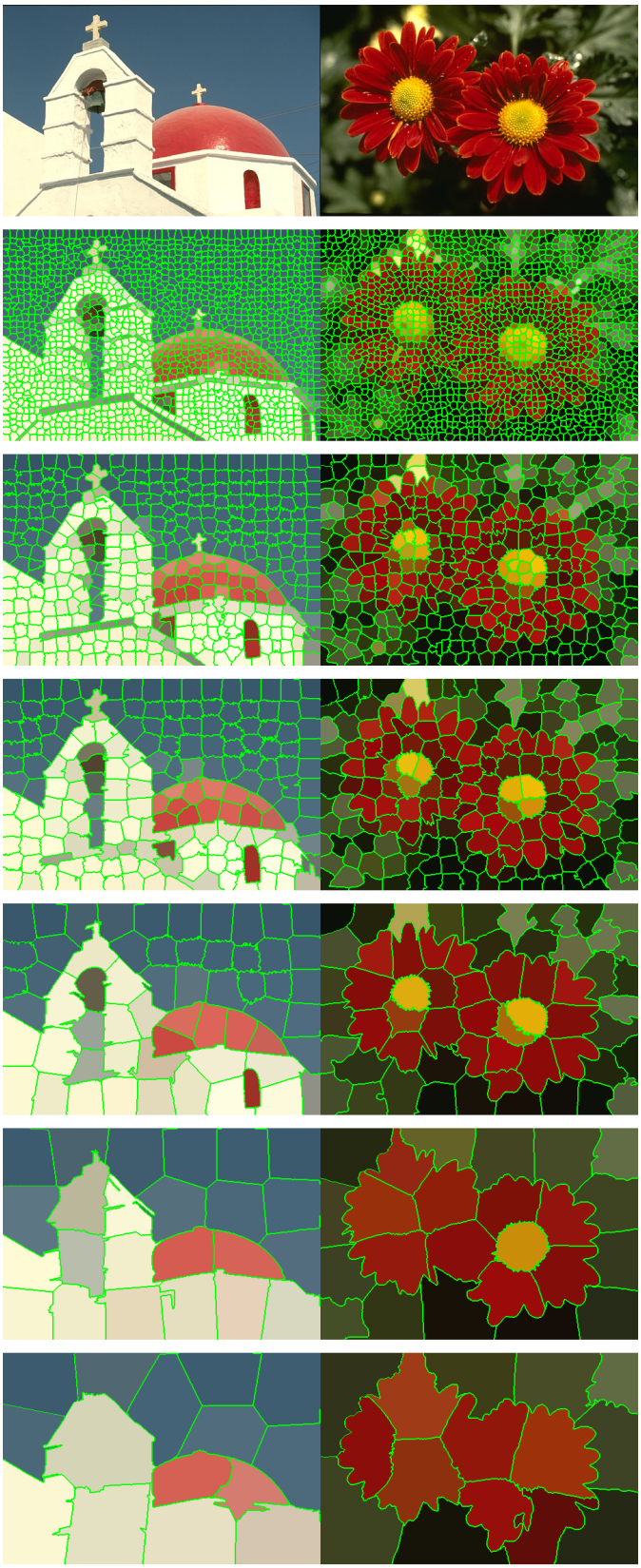


Figura 7.7: Resultado qualitativo para a extração de *superpixels* no modo SLIC0 variando o tamanho do *superpixel*, respectivamente, em $10, 20, 30, 50, 80, 100$. A primeira linha contém as imagens originais e as outras linhas imagens com a fronteira dos *superpixels* com intensidade média da cor do respectivo *superpixel*.

7.1.5 Análise e considerações

O algoritmo SLIC realiza a extração de *superpixels* utilizando a distância de intensidade de cor entre os *pixels*. Desse modo, ele apresenta bons resultados para objetos com cores distintas, porém o mesmo não se aplica a imagens com textura. Na figura 7.8 há o exemplo da extração de *superpixels* para uma imagem com textura. É possível perceber nesta imagem que as bordas da onça não foram detectadas. Devido à textura apresentada em sua pelagem, o algoritmo não foi capaz de diferenciá-lo da paisagem. Ainda é possível notar que parte das bordas encontradas por alguns *superpixels* foram as pintas que compõem a textura da pelagem da onça.

A figura 7.9 contém a representação da intensidade média dos *superpixels*, ainda para imagem da onça da figura 7.8, com distintos tamanhos de *superpixels*. Nela, é possível perceber que, mesmo para tamanhos menores, a qualidade da extração dos *superpixels* é baixa, o que prejudica as etapas posteriores do algoritmo.

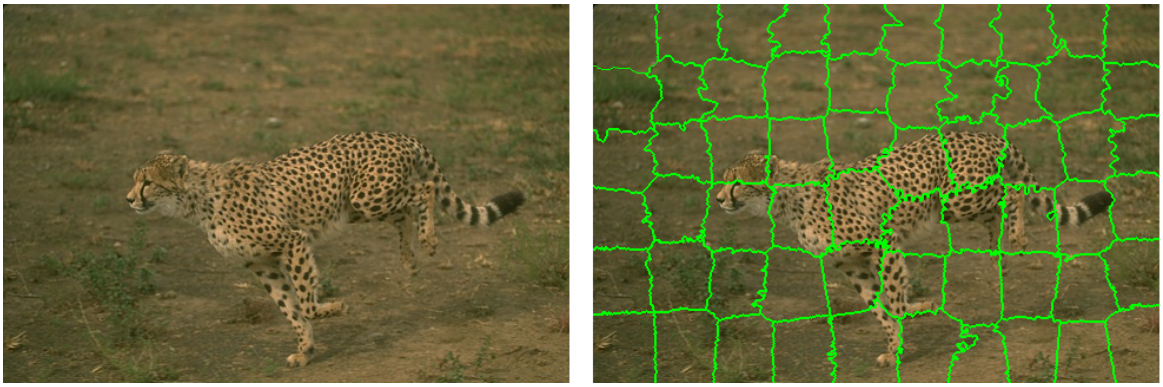


Figura 7.8: Resultado da extração de *superpixels* com algoritmo SLIC para uma imagem com textura. A primeira imagem é a original e a segunda contém os *superpixels* sobrepostos à imagem original para $S = 30$.

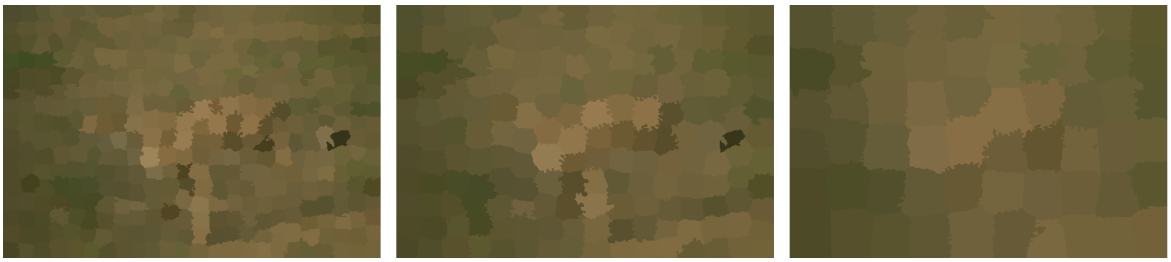


Figura 7.9: Resultado da extração de *superpixels* representado pela intensidade média do respectivo *superpixel*. Para tamanhos de *superpixels*, respectivamente, $20, 30$ e 50.

É importante ressaltar que o algoritmo SLIC é dependente da posição espacial dos objetos da imagem. A figura 7.10 ilustra essa dependência. Nesta figura, a primeira e a terceira imagem contêm os mesmos objetos, porém na segunda o objeto com formato circular é deslo-

cado. É possível notar que essa mudança de localização foi suficiente para que os *superpixels* conseguissem localizar parte da borda deste objeto, que passou despercebido anteriormente.

Portanto, a geometria da imagem, mesmo que apenas de um objeto, pode promover mudanças no resultado dos *superpixels*, tanto em qualidade e quanto em custo computacional.

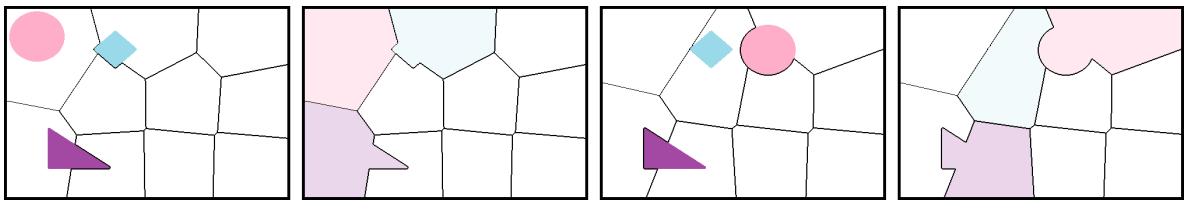


Figura 7.10: Representação da extração de *superpixels* para duas imagens distintas com mesmos parâmetros. A primeira e terceira imagem contém as bordas dos *superpixels* sobrepostos a imagem original. A segunda e quarta a borda dos *superpixels* preenchida que sua intensidade média de cor, respectivamente, para a primeira e segunda imagem.

Em resumo, com os resultados deste experimento pode-se concluir que: o modo SLIC0 tem maior aderência às bordas, com melhor resultado para *compactness* $0,0001$; o crescimento do tamanho dos *superpixels* reduz o tempo de processamento, porém diminui a qualidade; e que a redução do número máximo de iterações resulta em considerável redução de tempo de processamento, porém diminui-se a aderência às bordas, sendo o ponto de ótimo entre 5 e 10.

Após a extração de *superpixels* ser realizada, para cada conjunto de *pixels* constituinte de um *superpixel* é calculada a intensidade média do vetor tridimensional de cores e o seu centroide. Esse são os dados de saída da etapa de extração de características dos *superpixels* e são a entrada para a etapa de geração da rede complexa.

7.2 Análise dos principais parâmetros para geração da rede

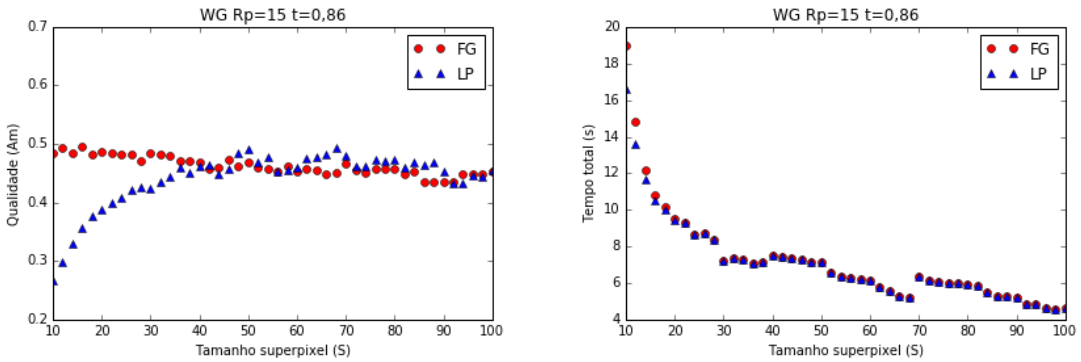
Como mencionado no capítulo 6, foram utilizadas quatro funções de peso para gerar as arestas da rede complexa, em conjunto com os parâmetros raio e *threshold*. Depois de gerada a rede, utilizou-se dois algoritmos de detecção de comunidade para encontrar as comunidades de *superpixels*: *Fast Gredy* e *Label Propagation*. Ao resultado da detecção de comunidades, pode-se aplicar um algoritmo para separar regiões não conectadas da segmentação.

Nas seções 7.2.1, 7.2.2 e 7.2.3, foram avaliadas as influências dos parâmetros raio, *threshold* e tamanho do *superpixel* para cada função peso e algoritmo de detecção. Para este experimento, aplicou-se ao resultado da detecção de comunidades um algoritmo de separação de regiões não conectadas.

Este experimento foi realizada para 50 imagens de treinamento do banco de imagens BSDS300. As 50 imagens foram escolhidas aleatoriamente e não foi utilizada o total de imagens de treinamento devido ao elevado tempo de processamento do experimento. O critério de avaliação foi quantitativo, utilizando-se da medida de tempo de processamento mensurada em segundos e da métrica de qualidade proposta em Arbelaez et al. (2009), detalhada na seção 6.5.

7.2.1 Função peso Gaussiana (WG)

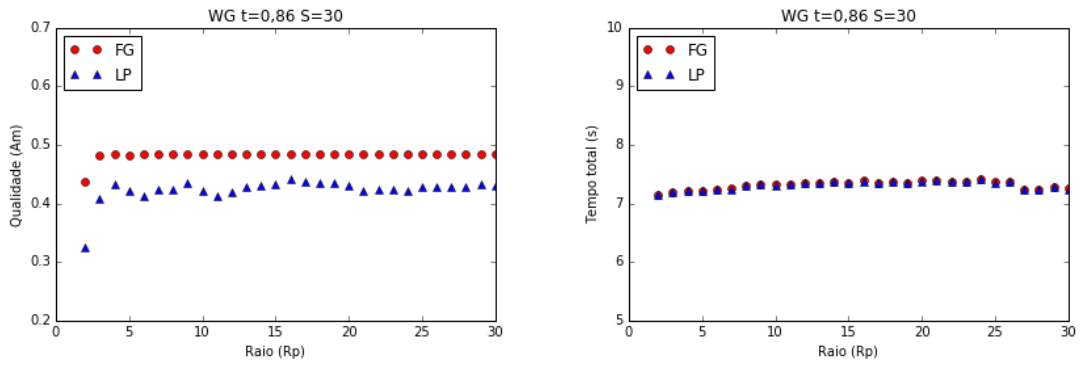
A figura 7.11 contém os resultados da influência do parâmetro tamanho do *superpixel* na função peso WG para ambos algoritmos de detecção de comunidade. Observando o gráfico em 7.11(b), é possível perceber que o decrescimento do tempo é exponencial para o tamanho do *superpixel*. Além disso, o tempo sofre pouca alteração entre o FG e LP, sendo bastante próximos, apresentando diferença considerável somente para $S < 15$. Esse resultado pode ser explicado pelo fato de que para $S > 15$ o tempo da extração de *superpixels* é muito maior que o tempo da detecção de comunidades, de modo que no tempo total apenas a primeira etapa é relevante. Entretanto, para $S < 15$, o número de nós da rede é muito grande, com isso, o tempo para a geração do grafo e detecção de comunidades cresce exponencialmente com a diminuição do tamanho do *superpixel*, se tornando bastante relevante para o tempo total do método de segmentação. Para o método FG, esse crescimento é ainda mais expressivo. Observando o gráfico em 7.11(a), é possível perceber que, para o FG, a qualidade da segmentação decresce com o aumento do *superpixel*, enquanto que, para o LP, a melhor qualidade se encontra entre $45 < S < 75$, não sendo bem definido qual o melhor S.



(a) (b)

Figura 7.11: Gráficos da influência do parâmetro tamanho do *superpixel* no tempo e na qualidade para função peso WG, $Rp = 15$, $t = 0,86$ e detecção de comunidade FG e LP.

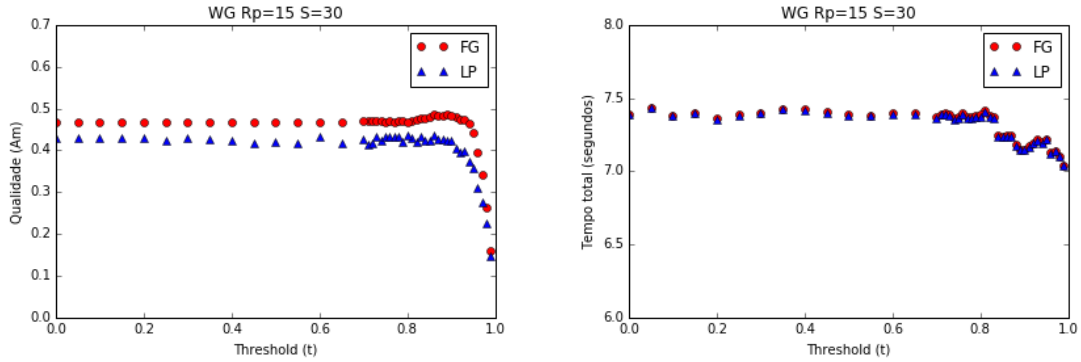
A figura 7.12 contém os resultados da influência do parâmetro raio na função de peso WG para ambos algoritmos de detecção de comunidade. Observando o gráfico em 7.12(b) é possível perceber que o tempo apresenta um pequeno crescimento com o aumento do raio, sendo bastante próximo para FG e LP. Essa pequena variação se deve ao aumento do número de arestas da rede, porém é pouco expressiva para $S = 30$, onde se predomina o tempo da extração de *superpixels*. Observando o gráfico em 7.12(a) é possível perceber que tanto para o FG quanto para o LP a variação de raio, para $R > 5$, foi pouco significativa para qualidade de segmentação. A explicação para este resultado se deve à função WG ser exponencial para a distância entre centroides dos *superpixels*. Para $R < 5$ poucas arestas são formadas, resultando em imagens super-segmentas, de baixa qualidade. Ainda é possível notar que o FG apresentou melhores resultados que o LP para todos os valores de raio avaliados, com os outros parâmetros fixos – função de peso WG, $S = 30$ e $t = 0,86$.



(a) (b)

Figura 7.12: Gráficos da influência do parâmetro raio no tempo e na qualidade para função peso WG, $S = 30$, $t = 0,86$ e detecção de comunidade FG e LP.

A figura 7.13 contém os resultados da influência do parâmetro *threshold* na função de peso WG para ambos algoritmos de detecção de comunidade. Observando o gráfico em 7.13(b), é possível perceber que o tempo decresce com o aumento de t com maior expressividade para $t \geq 0,84$, sendo bastante próximo para FG e LP. Essa variação se deve à diminuição de arestas da rede, que reduz o tempo de geração do grafo. Observando o gráfico em 7.13(a), é possível perceber que, tanto para o FG quanto para o LP, a qualidade tem pico para $0,75 < t < 0,95$. O pico de qualidade de FG ocorreu em $t = 0,89$ e o de LP em $t = 0,80$.

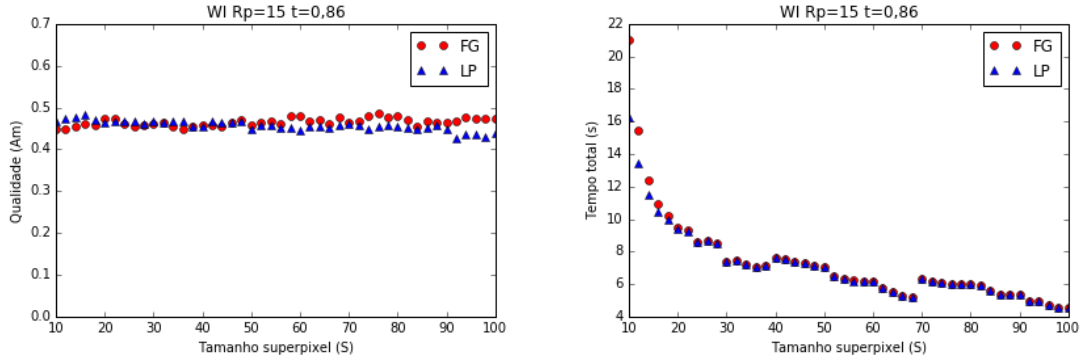


(a) (b)

Figura 7.13: Gráficos da influência do parâmetro *threshold* no tempo e na qualidade, para função peso WG, $S = 30$, $Rp = 15$ e detecção de comunidade FG e LP.

7.2.2 Função peso de intensidade (WI)

A figura 7.14 contém os resultados da influência do parâmetro tamanho do *superpixel* na função peso WI para ambos algoritmos de detecção de comunidade. Observando o gráfico em 7.14(b) é possível perceber que o decrescimento do tempo é exponencial para o tamanho do *superpixel*, apresentando alteração relevante entre o FG e LP apenas para $S < 15$. Este resultado é similar ao da figura 7.11(b), e suas causas idênticas. Observando o gráfico em 7.14(a), é possível perceber que, para LP, a qualidade da segmentação decresce com o aumento do *superpixel*, enquanto que, para o FG, esta curva sofreu muitas oscilações, sendo a região com melhor qualidade $55 < S < 85$.

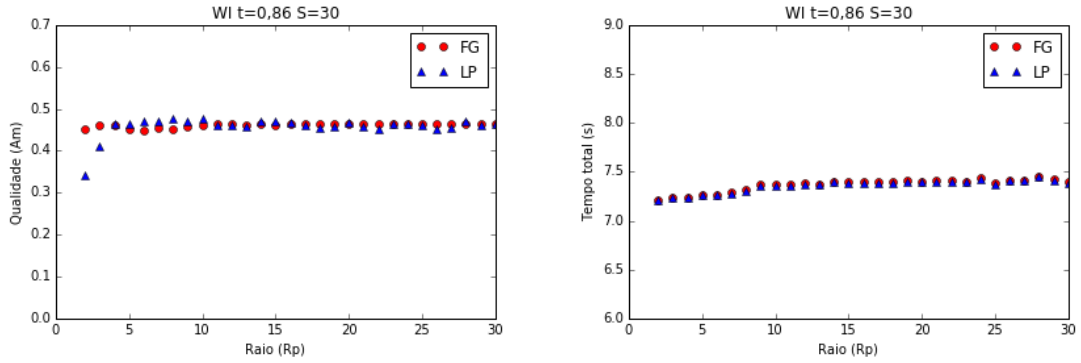


(a) (b)

Figura 7.14: Gráficos da influência do parâmetro tamanho do *superpixel* no tempo e na qualidade para função peso WI, $Rp = 15$, $t = 0,86$ e detecção de comunidade FG e LP.

A figura 7.15 contém os resultados da influência do parâmetro raio na função peso WI para ambos algoritmos de detecção de comunidade. Observando o gráfico em 7.15(b), é

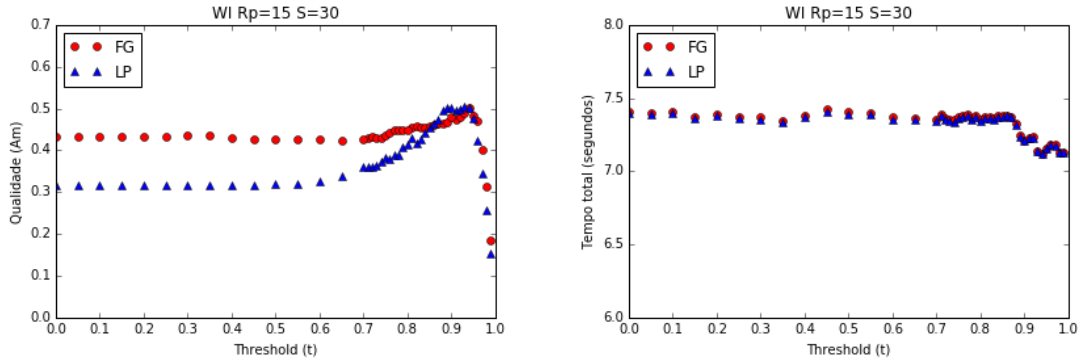
possível perceber que o tempo apresenta um pequeno crescimento para aumento do raio, sendo bastante próximo para FG e LP. Essa pequena variação se deve ao aumento do número de arestas da rede. Observando o gráfico em 7.15(a) é possível perceber que, tanto para o FG, quanto para o LP, a variação de raio, para $R > 5$, foi pouco significativa para qualidade de segmentação. A explicação para este fato se deve ao algoritmo de separação de regiões não conectadas. Para $R < 5$ poucas arestas são formadas, resultando em imagens super segmentas, principalmente para o LP, devido à dificuldade de propagação.



(a) (b)

Figura 7.15: Gráficos da influência do parâmetro raio no tempo e na qualidade para função peso WI, $S = 30$, $t = 0,865$ e detecção de comunidade FG e LP.

A figura 7.16 contém os resultados da influência do parâmetro *threshold* na função peso WI para ambos algoritmos de detecção de comunidade. Observando o gráfico em 7.16(b), é possível perceber que o tempo decresce com o aumento de t com maior expressividade para $t \geq 0,84$, sendo bastante próximo para FG e LP. Essa variação se deve à diminuição de arestas da rede, que reduz o tempo de geração do grafo. Observando o gráfico em 7.16(a) é possível perceber que, tanto para o FG, quanto para o LP, a qualidade tem pico para $0,8 < t < 0,95$. O pico de qualidade de FG ocorreu em $t = 0,94$ e o de LP em $t = 0,9$.

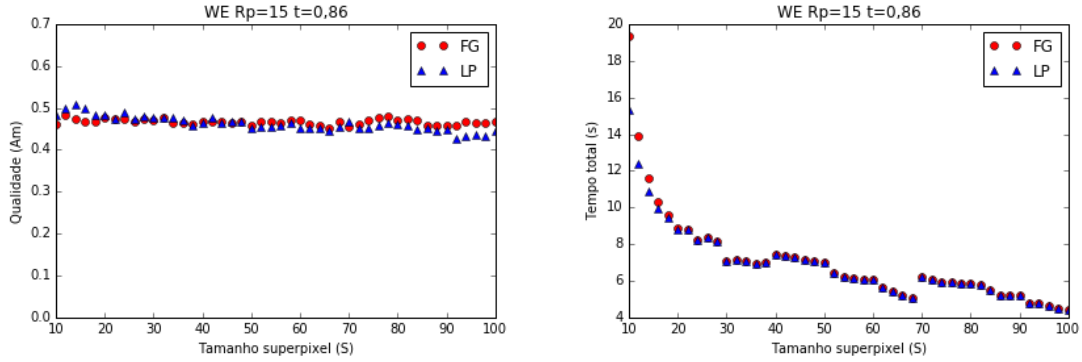


(a) (b)

Figura 7.16: Gráficos da influência do parâmetro *threshold* no tempo e na qualidade para função peso WI, $S = 30$, $Rp = 15$ e detecção de comunidade FG e LP.

7.2.3 Função peso exponencial (WE)

A figura 7.17 contém os resultados da influência do parâmetro tamanho do *superpixel* na função de peso WE para ambos algoritmos de detecção de comunidade. Observando o gráfico em 7.17(b), é possível perceber que o decrescimento do tempo é exponencial para o tamanho do *superpixel*, apresentando alteração relevante entre o FG e LP apenas para $S < 15$. Este resultado é similar ao das figuras 7.11(b) e 7.14(b), e suas causas idênticas. Observando o gráfico em 7.17(a) é possível perceber que, para LP, a qualidade da segmentação decresce com o aumento do *superpixel*, enquanto que, para o FG, esta curva se mantém aproximadamente constante.

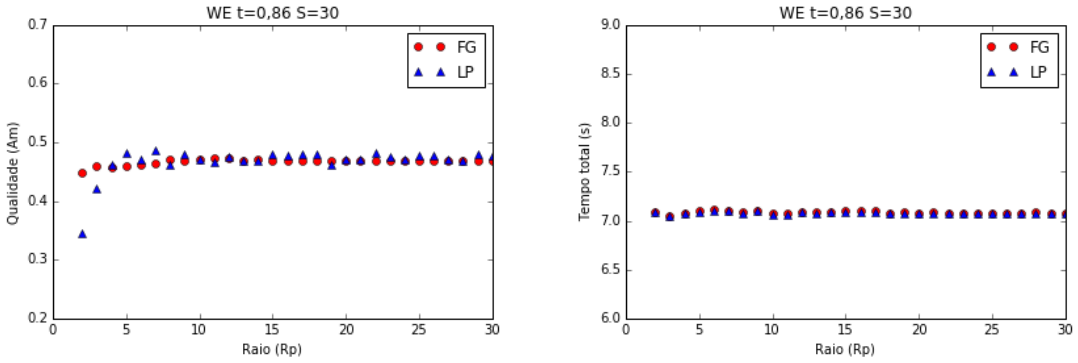


(a) (b)

Figura 7.17: Gráficos da influência do parâmetro tamanho do *superpixel* no tempo e na qualidade para função peso WE, $Rp = 15$, $t = 0,86$ e detecção de comunidade FG e LP.

A figura 7.18 contém os resultados da influência do parâmetro raio na função peso WE para ambos algoritmos de detecção de comunidade. Observando o gráfico em 7.18(b), é

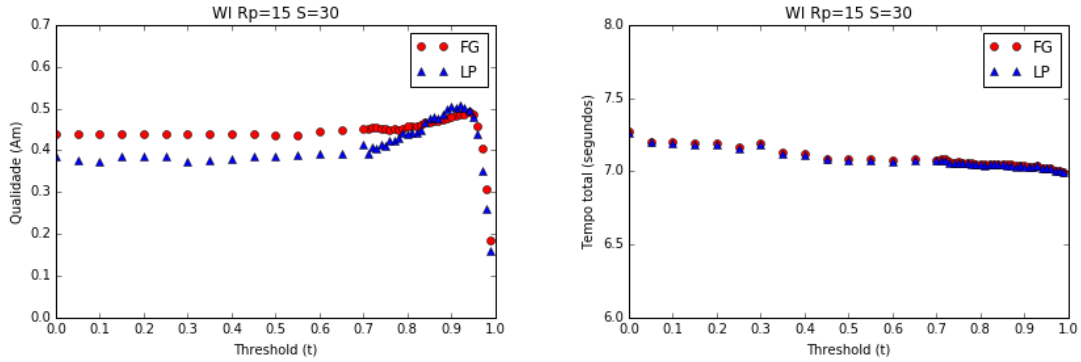
possível perceber que o tempo apresenta um pequeno crescimento para aumento do raio, sendo bastante próximo para FG e LP. Essa pequena variação se deve ao aumento do número de arestas da rede. Observando o gráfico em 7.18(a) é possível perceber que, tanto para o FG, quanto para o LP, a variação de raio, para $R > 5$, foi pouco significativa para qualidade de segmentação. A explicação para este fato se deve ao algoritmo de separação de regiões não conectadas. Para $R < 5$ poucas arestas são formadas, resultando em imagens super-segmentas para o LP, pois os *labels* têm poucas arestas para se propagar.



(a) (b)

Figura 7.18: Gráficos da influência do parâmetro raio no tempo e na qualidade para função peso WE, $S = 30$, $t = 0,86$ e detecção de comunidade FG e LP.

A figura 7.19 contém os resultados da influência do parâmetro *threshold* na função peso WE para ambos algoritmos de detecção de comunidade. Observando o gráfico em 7.19(b), é possível perceber que o tempo decresce com o aumento de t com maior expressividade para $t \geq 0,84$, sendo bastante próximo para FG e LP. Essa variação se deve à diminuição de arestas da rede, que reduz o tempo de geração do grafo. Observando o gráfico em 7.19(a), é possível perceber que, tanto para FG, quanto para LP, a qualidade tem pico para $0,8 < t < 0,95$. O pico de qualidade de FG ocorreu em $t = 0,94$ e o de LP em $t = 0,92$.



(a) (b)

Figura 7.19: Gráficos da influência do parâmetro *threshold* no tempo e na qualidade para função peso WE, $S = 30$, $Rp = 15$ e detecção de comunidade FG e LP.

7.2.4 Análise e considerações

Dos resultados deste experimento, é possível concluir que o comportamento do tempo total do método de segmentação proposto é praticamente independente da função de peso. O parâmetro mais influente no tempo é o tamanho do *superpixel*, pois se reflete no número de nós da rede, enquanto que o parâmetro mais importante para a qualidade é o *threshold*, que alcança maiores resultados para t em torno de $0,9$. O parâmetro raio, quando maior que 5, mostrou-se pouco influente no tempo e na qualidade para todas as funções peso. Além disso, foi possível observar que o LP não apresentou bons resultados para função peso WG, variações no σ_d podem melhorar este resultado.

7.3 Análise dos parâmetros secundários

Este experimento analisa a influência dos parâmetros restantes para encontrar os melhores parâmetros para o FG e LP. Nele são utilizados os resultados dos experimentos das seções 7.1 e 7.2. Da análise dos resultados destes experimentos, é possível concluir que, para o FG, o melhor resultado é dado para modo SLIC0, $cpact = 0,0001$, $imax = 6$, $S = 16$, $Rp = 8$, $t = 0,89$ e função peso WG. Já para o LP, o melhor resultado ocorre em modo SLIC0, $cpact = 0,0001$, $imax = 6$, $S = 16$, $Rp = 8$, $t = 0,92$ e função peso WE.

Os parâmetros avaliados neste experimento são o modelo de cor para extração da característica dos *superpixels* (RGB ou CIELAB), função de distância espacial (*Euclidean*, Manhattan ou Chebyshev), aplicação do método para separação de regiões não conexas (*on* ou *off*).

Este experimento foi realizado para as 200 imagens de treinamento do banco de dados BSDS300. O critério de avaliação foi quantitativo, utilizando-se o tempo de processamento mensurado em segundos e a métrica de qualidade proposta em Arbelaez et al. (2009), detalhada na seção 6.5.

7.3.1 Melhores resultados para FG

Nesta seção são avaliados para o algoritmo de detecção FG os parâmetros: modelo de cor, distância espacial e separação de regiões não conectadas. Para cada um deste parâmetros, é obtido um histograma do resultado individual para cada imagem e uma tabela com o tempo total e qualidade, que é a média dos resultados do histograma. Os parâmetros que não entram nesta análise foram fixos em: FG, modo SLIC0, $cpact = 0,0001$, $imax = 6$, $S = 16$, $Rp = 8$, $t = 0,89$ e função peso WG.

Os resultados da análise do parâmetro modelo de cor se encontram na tabela 7.1 e na figura 7.20, onde *Am* é a medida de qualidade do algoritmo, dada pela equação 6.14. Os resultados da tabela e do histograma indicam que modelo de cor RGB apresenta melhor qualidade e custo computacional quando comparado ao CIELAB.

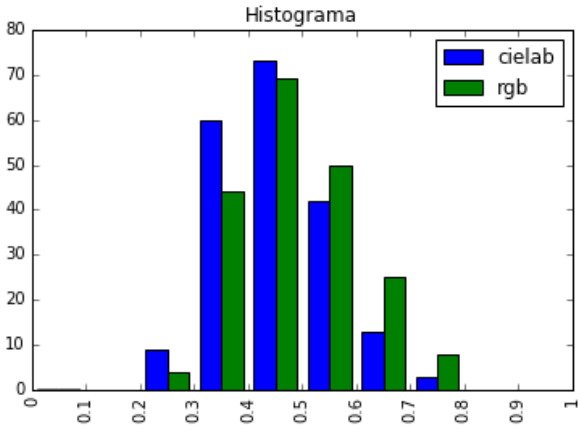


Figura 7.20: Histograma do parâmetro modelo de cor em FG

Tabela 7.1: Modelo de cor em FG

	Am	Tempo total(s)
CIELAB	0,452	5,50
RGB	0,486	5,37

Os resultados da análise do parâmetro distância espacial se encontram na tabela 7.2 e na figura 7.21. Os resultados da tabela e do histograma indicam que, em termos de qualidade, a diferença entre os resultado é pequena, sendo o pior caso para a Manhattan. A diferença entre Chebyshev e Euclidean é pouco significativa em termos de qualidade, porém em custo computacional o Chebyshev apresenta melhores resultados.

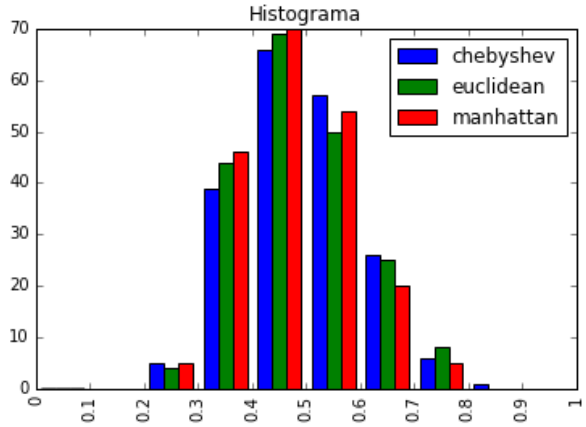


Figura 7.21: Histograma do parâmetro distância espacial em FG

Tabela 7.2: Distância espacial em FG

	Am	Tempo total(s)
Chebyshev	0,488	4,90
Euclidean	0,486	5,36
Manhattan	0,473	4,99

Os resultados da análise do parâmetro separação de regiões não conectadas se encontram na tabela 7.3 e na figura 7.22. Os resultados da tabela e do histograma indicam que, em termos de qualidade, o modo *on* (com separação de regiões não conectadas) apresentou melhores resultados. Entretanto, em termos de custo computacional, o modo *off* (sem separação regiões não conectadas) mostrou-se melhor. A diferença em qualidade é de $1,45\%$ e em tempo de $1,27\%$.

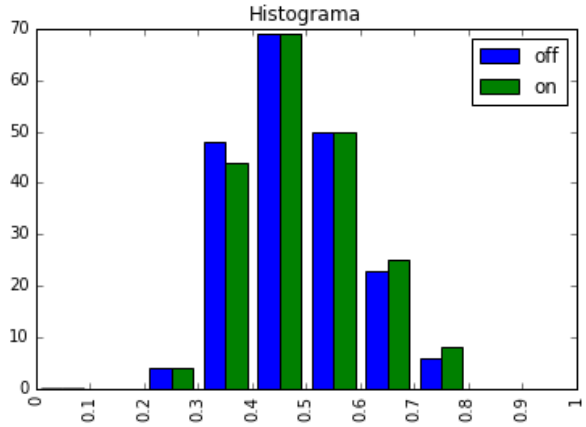


Figura 7.22: Histograma para parâmetro separação de regiões não conectadas em FG

Tabela 7.3: Separação de regiões não conectadas em FG

	Am	Tempo total(s)
Off	0,479	5,45
On	0,486	5,52

7.3.2 Melhores resultados para LP

Nesta seção são avaliados para o algoritmo de detecção LP os parâmetros: modelo de cor, distância espacial e separação de regiões não conectadas. Para cada um deste parâmetros, é

obtido um histograma do resultado individual para cada imagem e uma tabela com o tempo total e qualidade, que é a média dos resultados do histograma. Os parâmetros que não entram nesta análise foram fixos em: LP, modo SLIC0, $cpact = 0,0001$, $imax = 6$, $S = 16$, $Rp = 8$, $t = 0,92$ e função peso WE.

Os resultados da análise do parâmetro modelo de cor se encontram na tabela 7.4 e na figura 7.23. Os resultados da tabela e do histograma indicam que modelo de cor RGB apresenta melhor qualidade e custo computacional quando comparado ao CIELAB.

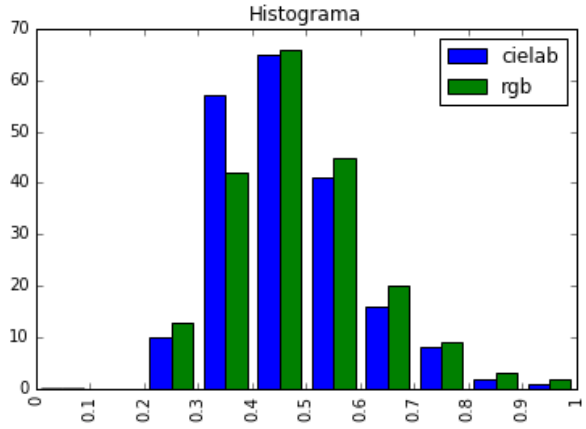


Figura 7.23: Histograma do parâmetro modelo de cor em LP

Tabela 7.4: Modelo de cor em LP

	Am	Tempo total(s)
CIELAB	0,465	5,31
RGB	0,482	5,24

Os resultados da análise do parâmetro distância espacial se encontram na tabela 7.5 e na figura 7.24. Os resultados da tabela e do histograma indicam que, em termos de qualidade, a diferença entre os resultado é pequena, sendo o pior caso para Manhattan. A diferença entre Chebyshev e Euclidean é pouco significativa em termos de qualidade, porém, em custo computacional, o Chebyshev apresenta melhores resultados.

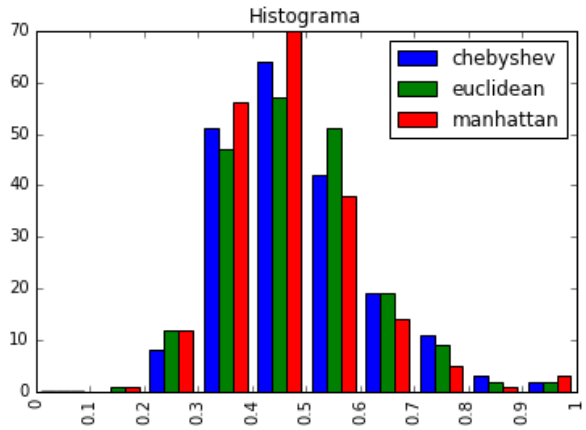


Figura 7.24: Histograma do parâmetro distância espacial em LP

Tabela 7.5: Distância espacial em LP

	Am	Tempo total(s)
Chebyshev	0,481	4,79
Euclidean	0,483	5,20
Manhattan	0,456	4,88

Os resultados da análise do parâmetro separação de regiões não conectadas se encontram na tabela 7.6 e na figura 7.25. Os resultados da tabela e do histograma indicam que, em termos de qualidade, o modo *on* (com separação de regiões não conectadas) apresentou melhores resultados. Entretanto, em termos de custo computacional, o modo *off* (sem separação regiões não conectadas) mostrou-se melhor. A diferença em qualidade é de $2,76\%$ e em tempo de $0,74\%$.

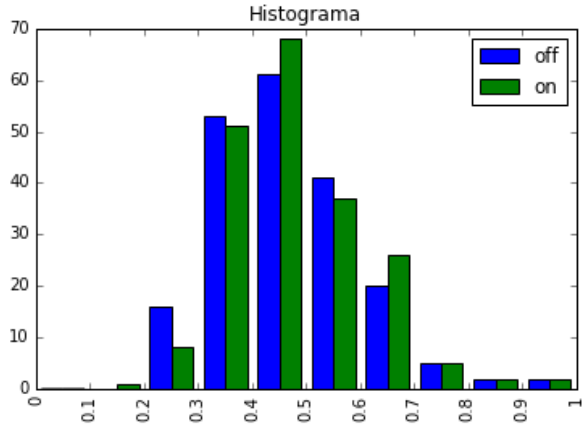


Figura 7.25: Histograma para parâmetro separação de regiões não conectadas em LP

Tabela 7.6: Separação de regiões não conectadas em LP

	Am	Tempo total(s)
Off	0,465	5,39
On	0,478	5,35

7.3.3 Análise e considerações

Dos resultados deste experimento, é possível concluir que, dos parâmetros analisados, os melhores resultados, tanto para o FG, quanto para o LP, ocorreram para: modelo de cor RGB, distância espacial Chebyshev e separação de regiões não conectadas *on*.

Destes parâmetros, o que mostrou maior relevância foi o modelo de cor, cuja diferença em qualidade chegou a $7,25\%$ para o FG e a $3,6\%$ para o LP. Além disso, a distância espacial Manhattan mostrou resultados ruins de qualidade para LP, cuja diferença chegou a $1,93\%$.

7.4 Comparação dos resultados

O experimento apresentado nesta seção realiza a comparação entre o método de segmentação de imagens proposto neste trabalho e o método EGB proposto por Felzenszwalb e Huttenlocher (2004). Para esta comparação, foram utilizadas as 100 imagens teste do banco de imagens BSDS300. Como o método proposto é baseado em extração de *superpixels* e

detecção de comunidades, utilizou-se duas versões diferentes deste: uma que utilizada FG e a outra LP.

Para o EGB, foram utilizados os parâmetros definidos pelos autores como sendo a melhor combinação. Para o método proposto, os parâmetros utilizados foram escolhidos com base nos resultados dos experimentos explicitados nas seções anteriores deste capítulo, de modo a se manter o compromisso entre acurácia e tempo de processamento. Para a segmentação que utiliza o algoritmo de detecção de comunidades FG, os parâmetros utilizados foram: modo SLIC0, $cpact = 0,0001$, $imax = 6$, $S = 16$, $Rp = 8$, $t = 0,89$, função peso WG, função distância espacial Chebyshev, modelo de cor RGB e separação de regiões não conectadas *on*. Já para a segmentação com o uso do algoritmo de detecção de comunidades LP, utilizou-se: modo SLIC0, $cpact = 0,0001$, $imax = 6$, $S = 16$, $Rp = 8$, $t = 0,92$, função peso WE, função distância espacial Chebyshev, modelo de cor RGB e separação de regiões não conectadas *on*.

Tabela 7.7: Resultados da comparação quantitativa entre o método de segmentação EGB e os métodos de segmentação propostos que utilizam os algoritmos de detecção FG e LP.

	Am	Tempo total(s)	Nº segmentos
FG	0,474	4,64	34,7
LP	0,462	4,48	36,6
EGB	0,465	0,60	1640,2

A tabela 7.7 contém os resultados quantitativos de qualidade, tempo e número de segmentos encontrados para as técnicas citadas. A figura 7.26 contém o histograma dos resultados de qualidade obtidos para cada imagem e a figura 7.27 contém o resultado qualitativo da segmentação encontrada por essas técnicas, com sua respectiva qualidade quantitativa e imagem original. A análise destas figuras e tabela mostram que ambos os métodos propostos obtiveram qualidade semelhante à do EGB, entretanto com tempo de processamento muito maior. Em contrapartida, a média do número de segmentos encontrado por eles é expressivamente menor que a média do EGB, já que este último gera imagens super-segmentadas. Outra medida de comparação é o resultado qualitativo de aderência de bordas. O EGB obteve melhor aderência às bordas que os métodos propostos para objetos pequenos; para objetos grandes, o resultado é similar.

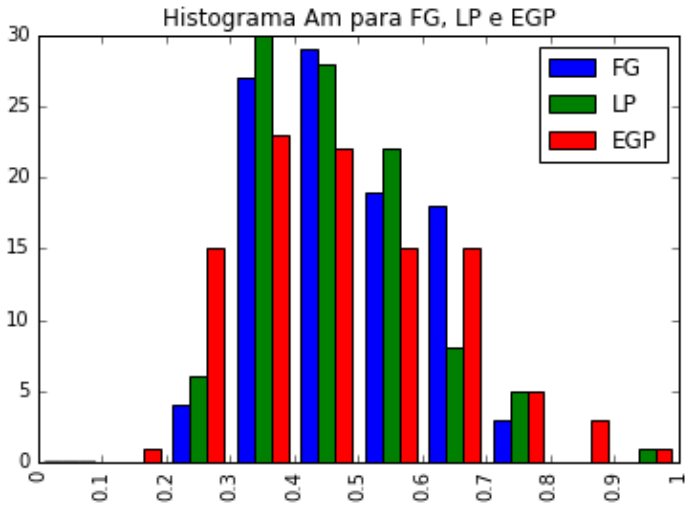


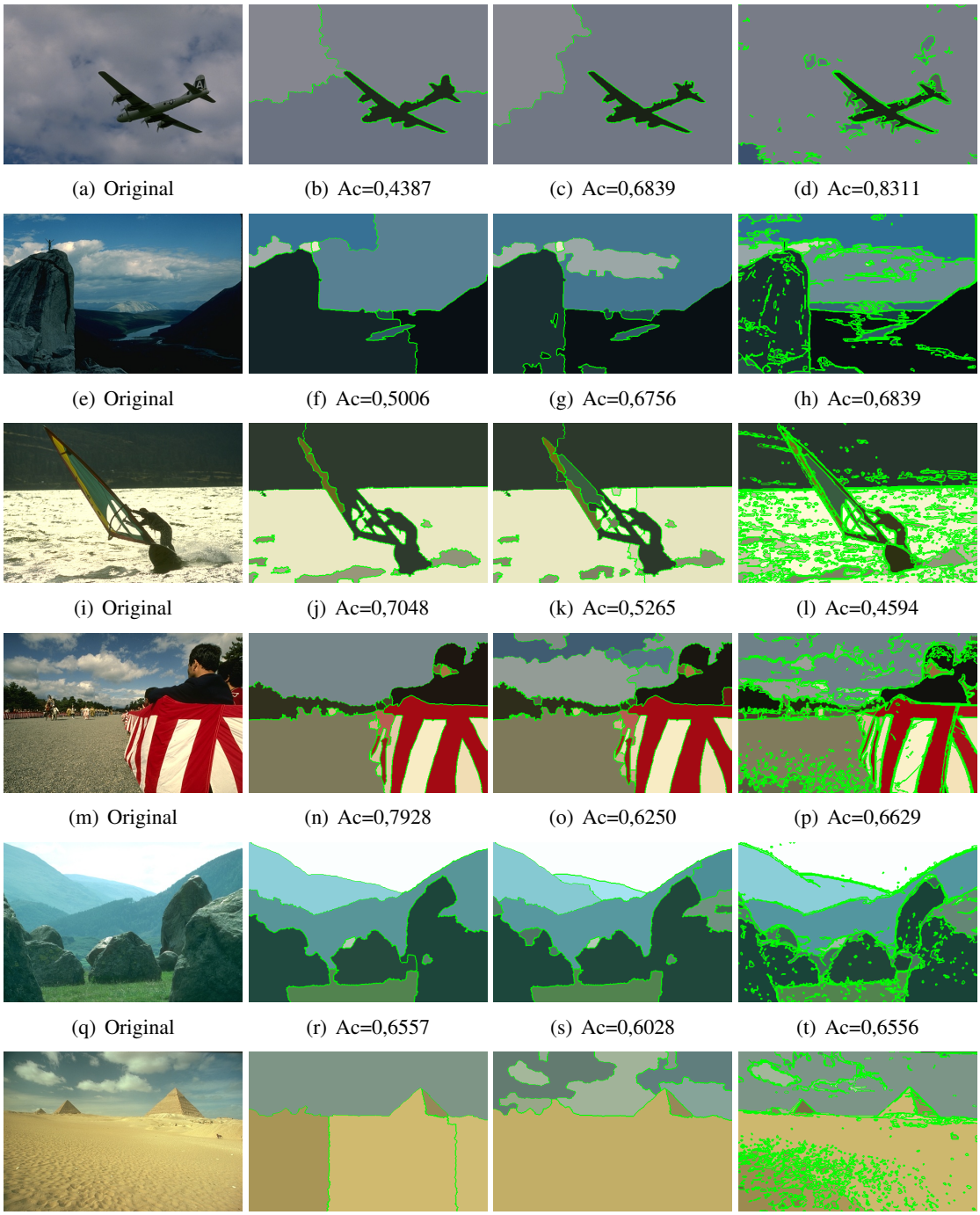
Figura 7.26: Histograma comparativo do resultado quantitativo de qualidade entre o método de segmentação EGB e os métodos de segmentação propostos que utilizam os algoritmos de detecção FG e LP.

Desse modo, é possível concluir que os métodos propostos obtiveram bons resultados de qualidade, quantidade de segmentos e aderência às bordas, porém tempo de processamento muito alto. O resultado de qualidade do método proposto com uso do algoritmo de detecção comunidades FG foi $1,9\%$ maior que o EGB e com LP $0,6\%$ menor. Devido à sua quantidade de segmentos ser muito menor que o do EGB, cerca de $2,2\%$ do encontrado no EGB, para alguns pós-processamentos os métodos propostos são mais viáveis. Para o cálculo de qualidade, por exemplo, as imagens segmentadas pelo EGB demoram entre 2 e 3 minutos para o cálculo de qualidade Ac por imagem, enquanto que os métodos propostos demoram menos de 2 segundos.

Tabela 7.8: Resultados do tempo de processamento, em segundos, para os métodos de segmentação propostos, respectivamente, para as etapas geração de *superpixels*, extração de suas características, geração da rede e detecção de comunidades.

	Etapa 1	Etapa 2	Etapa 3	Etapa 4	Total
FG	3,590	0,368	0,602	0,076	4,64
LP	3,590	0,341	0,535	0,013	4,48

A tabela 7.8 contém o tempo para cada etapa dos métodos propostos. Nela, é possível perceber que $77,4\%$ do tempo é para a extração de *superpixels*. Melhorias deste valor são encontradas reduzindo-se o parâmetro *imax* ou aumentando o parâmetro S, porém, em ambos os casos, perde-se aderência às bordas. A tabela 7.9 exemplifica as alterações causadas por estes parâmetros, mantendo os outros constantes. Nela, é perceptível que o menor tempo



(a) Original (b) Ac=0,4387 (c) Ac=0,6839 (d) Ac=0,8311

(e) Original (f) Ac=0,5006 (g) Ac=0,6756 (h) Ac=0,6839

(i) Original (j) Ac=0,7048 (k) Ac=0,5265 (l) Ac=0,4594

(m) Original (n) Ac=0,7928 (o) Ac=0,6250 (p) Ac=0,6629

(q) Original (r) Ac=0,6557 (s) Ac=0,6028 (t) Ac=0,6556

(u) Original (v) Ac=0,5027 (w) Ac=0,7074 (x) Ac=0,6129

Figura 7.27: Resultado qualitativo e quantitativo da segmentação para algumas das 100 imagens testes do banco BSDS300 obtidos pelos métodos de segmentação propostos e do método EGB, com a respectiva imagem original.

ocorre para $S = 30$ e $imax = 2$, porém este tempo ainda é muito maior que o do EGB. Ainda nesta tabela, é possível perceber que a alteração da medida de qualidade *Am* é pequena para alterações destes parâmetros, porém a perda de aderência às bordas para $imax < 6$ é significativa. É importante ressaltar que, na métrica quantitativa de qualidade utilizada, proposta por Arbelaez et al. (2009), perdas de bordas são pouco detectadas, pois nela é comparada a quantidade de *pixels* e, em geral, perdas pouco expressivas de bordas apresentam pequenas quantidades de *pixels* quando comparada ao total da imagem. Outra alternativa para se reduzir o tempo de processamento total é utilizar um algoritmo de extração de *superpixels* com menor tempo de processamento.

Tabela 7.9: Resultado comparativo de qualidade e tempo, em segundos, para outros valores de *imax* e *S*.

	Am FG	Tempo FG	Am LP	Tempo LP
S=16 e imax=6	0,474	4,637	0,462	4,479
S=16 e imax=2	0,467	2,525	0,460	2,422
S=30 e imax=6	0,462	4,177	0,471	4,120
S=30 e imax=2	0,464	1,661	0,442	1,675

Capítulo 8

Conclusão

Neste trabalho foi proposto um método de segmentação de imagens que combina extração de *superpixels* com algoritmos de detecção de comunidades, pertencentes à teoria das redes complexas. Este método é composto pelas seguintes etapas: cálculos dos *superpixels*, extração de características dos *superpixels*, geração de uma rede complexa com as características dos *superpixels* e, por fim, detecção das comunidades existentes. Diversas técnicas para a execução destas etapas foram exploradas ao longo dos experimentos para se definir a melhor combinação destas para o método final. Todos os experimentos realizados foram implementados na linguagem de programação Python, em um mesmo computador e em imagens do banco BSDS300.

Para o cálculo dos *superpixels* foi utilizada o algoritmo SLIC, cujos parâmetros foram analisados em busca da melhor combinação. Dessa análise, obteve-se as seguintes conclusões: o modo SLIC0 tem maior aderência as bordas; o tempo de processamento é inversamente proporcional ao tamanho dos segmentos; redução do número máximo de iterações resulta em considerável redução de tempo de processamento, porém diminui-se a aderência às bordas, bons resultados foram encontrados para 6 iterações; o SLIC tem pouca aderência às bordas de objetos com texturas; objetos com tamanho menor que *superpixels* dificilmente são detectados; e o SLIC é dependente da disposição geométrica dos objetos que compõem a imagem. As características extraídas dos *superpixels* foram os centroides e média do vetor de cores, devido ao fato do algoritmo SLIC encontrar os *superpixels* com base na diferença da intensidade de cor.

Para geração da rede complexa foram analisadas três funções de peso (WG, WE e WI), três funções para cálculo da distância espacial (Chebyshev, Euclidean e Manhattan), dois modelos de cores (RGB e CIELAB), raio e *threshold*. Os melhores resultados foram encontrados

para função de distância espacial Chebyshev, modelo de cor RGB, *threshold* pertencente ao intervalo de $0,89$ a $0,92$ e raio maior que 5. Destes parâmetros, o *threshold* foi mais influente na qualidade. A variação da qualidade devido à função peso mostrou dependência em relação ao algoritmo de detecção de comunidades.

Dois algoritmos de detecção de comunidades foram utilizados (FG e LP) e para cada um deles foi avaliado qual a rede complexa obteve maior desempenho. Como resultado, o FG apresentou melhor qualidade para função de peso WG, enquanto que o LP apresentou para o WE. Ambos os algoritmos de detecção de comunidades apresentaram desempenhos semelhantes com o uso de sua rede ótima. À segmentação encontrada pela detecção de comunidades, avaliou-se a aplicação de um algoritmo para separar regiões não conectadas, cujo resultado apresentou melhor qualidade com esse pós-processamento.

Finalmente, o método proposto apresentou qualidade semelhante ao de EGB, com menor número médio de segmentos, cerca de $2,2\%$ do encontrado no EGB, porém maior tempo de processamento. O resultado de qualidade do método proposto com uso do algoritmo de detecção comunidades FG foi $1,9\%$ maior que o EGB e com LP $0,6\%$ menor.

8.1 Contribuições

As principais contribuições deste trabalho são:

- Proposição de um método de segmentação que combina algoritmos de *superpixels* e detecção de comunidades. De modo a possibilitar a aplicação de algoritmos de detecção de comunidades em imagens de alta resolução, devido a redução de vértices da rede complexa.
- Estudo da influência dos parâmetros do algoritmo de extração de *superpixels* SLIC.
- Estudo da influência dos parâmetros envolvidos na abordagem de segmentação proposta, com uma nova função de peso.

8.2 Limitações

A segmentação de imagens é um processo que envolve muita subjetividade e, quando realizada manualmente, é bastante influenciada por conhecimentos empíricos acerca de objetos relevantes em uma figura. Uma das limitações do método proposto é o fato de não incluir

técnica capaz de representar esse conhecimento empírico.

Outra limitação do método proposto é fato do seu algoritmo de *superpixel* e descritores de características serem baseados somente em informações de cor, de modo que objetos com textura dificilmente são detectados, fazendo com que imagens com textura apresentem baixa qualidade de segmentação.

Por fim, outra limitação deste método é a dependência dos parâmetros, principalmente o *threshold*, pois este parâmetro é muito significativo para a qualidade da segmentação e dependente da variação de cores da imagem.

8.3 Trabalhos Futuros

Em trabalhos futuros pretende-se investigar um método de convergência de *superpixels* com menor tempo de processamento e que utilize uma combinação entre cor e textura. Pretende-se também extrair informações de textura dos *superpixels* e propor novas funções de peso para que a etapa de geração da rede considere informações de textura e cor. Outra proposta para os trabalhos futuros é estudar e avaliar outros métodos de pré-segmentação em substituição ao de *superpixels*.

Por fim, em trabalhos futuros, pretende-se formular outras funções de peso e técnicas para o cálculo automático e adaptativo dos parâmetros *threshold* e raio, em busca de melhor qualidade e menor custo computacional.

Referências Bibliográficas

ABOUD NETA, S. R.; DUTRA, L. V.; ERTHAL, G. J. Limiarização automática em histogramas multimodais. 7^{th} *Brazilian Conference on Dynamics, Control and Aplications*, May 2008.

ACHANTA, R. et al. Slic superpixels. *Technical Report 149300 École Polytechnique Fédéral de Lausssanne*, 2010.

ACHANTA, R. et al. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 34, n. 11, p. 2274–2282, 2012.

AKBAR, S. et al. Tumor localization in tissue microarrays using rotation invariant superpixel pyramids. In: IEEE. *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*. [S.l.], 2015. p. 1292–1295.

ALBERT, R.; BARABÁSI, A.-L. Statistical mechanics of complex networks. *Reviews of modern physics*, APS, v. 74, n. 1, p. 47, 2002.

ALMEIDA, L. J. *Detecção de comunidades em redes complexas utilizando estratégia multinível*. Tese (Doutorado) — Universidade de São Paulo, 2009.

ARBELAEZ, P. et al. *From contours to regions: An empirical evaluation*. Miami, 2009. 2294–2301 p.

BACKHAUS, W. G. et al. Physiological and psychophysical simulations of color vision in humans and animals. *Color Vision-Perspectives from Different Disciplines, eds. W. Backhaus, R. Kliegl, JS Werner*, p. 45–77, 1998.

BALLARD, D. H.; BROWN, C. M. Computer vision, 1982. *Prenice-Hall, Englewood Cliffs*, 1982.

BARABÁSI, A.-L. Linked: How everything else is connected to everything else and what it means for business, science, and everyday life. Plume Printing, Penguin Group New York, 2003.

BARABÁSI, A.-L. Scale-free networks: a decade and beyond. *science*, American Association for the Advancement of Science, v. 325, n. 5939, p. 412–413, 2009.

BARABÁSI, A.-L.; ALBERT, R. Emergence of scaling in random networks. *science*, American Association for the Advancement of Science, v. 286, n. 5439, p. 509–512, 1999.

BELIZARIO, I. V. *Avaliação de algoritmos de agrupamento em grafos para segmentação de imagens*. Tese (Doutorado) — Universidade de São Paulo, 2012.

BERGH, M. Van den et al. Seeds: Superpixels extracted via energy-driven sampling. In: SPRINGER. *12th European conference on computer vision (ECCV)*. Florence, 2012. p. 13–26.

BERTHOLDO, F. A. R.; ARAÚJO, A. de A. *Técnicas de limiarizaçao para melhorar a qualidade visual de documentos históricos*. Dissertação de Mestrado em Ciência da Computação — Universidade Federal de Minas Gerais, 2007.

BEUREN, A. T.; PINHEIRO, R. J. G.; FACON, J. Abordagem morfológica de segmentação do melanoma. In: LEARNING AND NONLINEAR MODELS. *VII Workshop de Visão Computacional*. Curitiba, 2011. v. 7, p. 249–254.

BEZDEK, J. C.; HALL, L.; CLARKE, L. Review of mr image segmentation techniques using pattern recognition. *Medical Physics*, v. 20, n. 4, p. 1033–1048, 1992.

BINS, L. S. et al. Satellite imagery segmentation: a region growing approach. *Simpósio Brasileiro de Sensoriamento Remoto*, Imagem Multimidia, São Paulo. Proceedings, CD Salvador, Bahia, Brazil, v. 8, n. 1996, p. 677–680, 1996.

BOLLOBÁS, B. Modern graph theory, volume 184 of graduate texts in mathematics. Springer-Verlag, New York, 1998.

BONVENTI JUNIOR, W.; SMITH, W. S.; MOREIRA, P. A. P. Segmentação em cores de imagens aéreas por agrupamentos nebulosos. *Revista Hipótese*, v. 1, n. 2, p. 92–109, 2015.

BOTELHO, G. M. *Segmentação de imagens baseada em redes complexas e superpixels: uma aplicação ao censo de aves*. Tese (Doutorado) — ICMC, Universidade de São Paulo, 2014.

BRADSKI, G. *Dr. Dobb's Journal of Software Tools*, 2000.

BRANDES, U. et al. On modularity-np-completeness and beyond. Citeseer, 2006.

BRICE, C. R.; FENNEMA, C. L. Scene analysis using regions. *Artificial intelligence*, Elsevier, v. 1, n. 3, p. 205–226, 1970.

CIGLA, C.; ALATAN, A. Efficient graph-based image segmentation via speeded-up turbo pixels. In: IEEE. *17th IEEE International Conference on Image Processing*. Hong Kong, 2010. p. 3013–3016.

CLAUSET, A.; NEWMAN, M. E.; MOORE, C. Finding community structure in very large networks. *Physical review E*, APS, v. 70, n. 6, p. 066111, 2004.

COLEMAN, G. B.; ANDREWS, H. C. Image segmentation by clustering. *Proceedings of the IEEE*, IEEE, v. 67, n. 5, p. 773–785, 1979.

COMANICIU, D.; MEER, P. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, IEEE, v. 24, n. 5, p. 603–619, 2002.

COSTA, L. d. F. et al. Analyzing and modeling real-world phenomena with complex networks: a survey of applications. *Advances in Physics*, Taylor & Francis, v. 60, n. 3, p. 329–412, 2011.

COSTA, L. d. F. et al. Characterization of complex networks: A survey of measurements. *Advances in physics*, Taylor & Francis, v. 56, n. 1, p. 167–242, 2007.

COSTA, L. d. F. D.; CESAR JR, R. M. Shape analysis and classification: theory and practice. CRC Press, Inc., 2000.

CSARDI, G.; NEPUSZ, T. The igraph software package for complex network research. *InterJournal*, Complex Systems, p. 1695, 2006. Disponível em: <http://igraph.org>.

CUADROS, O. *Segmentação de imagens de alta dimensão por meio de algoritmos de detecção de comunidades e super pixels*. Dissertação de Mestrado em Ciências de Computação e Matemática Computacional — ICMC, Universidade de São Paulo, 2013.

DLUGOSZ, F. L. et al. Uso da segmentação por crescimento de regiões em imagem ikonos na discriminação de tipologias da floresta ombrófila mista. *Simpósio Brasileiro de Sensoriamento Remoto (SBSR)*, v. 12, p. 1493–1500, 2005.

DOROGOVTSEV, S. N.; MENDES, J. F. Evolution of networks. *Advances in physics*, Taylor & Francis, v. 51, n. 4, p. 1079–1187, 2002.

ERDÖS, P.; RÉNYI, A. On random graphs, i. *Publicationes Mathematicae (Debrecen)*, v. 6, p. 290–297, 1959.

ERDÖS, P.; RÉNYI, A. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci*, Citeseer, v. 5, p. 17–61, 1960.

ERDÖS, P.; RÉNYI, A. On the strength of connectedness of a random graph. *Acta Mathematica Hungarica*, Akadémiai Kiadó, co-published with Springer Science+ Business Media BV, Formerly Kluwer Academic Publishers BV, v. 12, n. 1-2, p. 261–267, 1961.

EULER, L. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, v. 8, p. 128–140, 1741.

FAN, J. et al. Automatic image segmentation by integrating color-edge extraction and seeded region growing. *IEEE transactions on image processing*, IEEE, v. 10, n. 10, p. 1454–1466, 2001.

FELZENSZWALB, P.; HUTTENLOCHER, D. Efficient graph-based image segmentation. *International Journal of Computer Vision*, Springer, v. 59, n. 2, p. 167–181, 2004.

FIDUCCIA, C. M.; MATTHEYSES, R. M. A linear-time heuristic for improving network partitions. In: IEEE. *Design Automation, 1982. 19th Conference on*. Piscataway, 1982. p. 175–181.

FORTUNATO, S. Community detection in graphs. *Physics reports*, Elsevier, v. 486, n. 3, p. 75–174, 2010.

FREEMAN, L. C. A set of measures of centrality based on betweenness. *Sociometry*, JSTOR, p. 35–41, 1977.

FULKERSON, B. et al. Class segmentation and object localization with superpixel neighborhoods. v. 9, p. 670–677, 2009.

FURUIE, S.; MOURA, L.; UDUPA, J. Classificação de imagens médicas 3d baseado em vetor de atributos. *Revista Brasileira de Engenharia - Caderno de Engenharia Biomédica*, v. 12, n. 3, p. 75–86, 1996.

GE, F.; WANG, S.; LIU, T. Image-segmentation evaluation from the perspective of salient object extraction. In: IEEE. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. New York, 2006. v. 1, p. 1146–1153.

GONZALEZ, R. C.; WOODS, R. E. Processamento de imagens digitais. Edgard Blucher, 2000.

HARALICK, R.; KELLY, G. Pattern recognition with measurement space and spatial clustering for multiple images. *Proceedings of the IEEE*, IEEE, v. 57, n. 4, p. 654–665, 1969.

HARALICK, R. M.; SHAPIRO, L. G. Image segmentation techniques. *Computer vision, graphics, and image processing*, Elsevier, v. 29, n. 1, p. 100–132, 1985.

HOIEM, D.; EFROS, A. A.; HEBERT, M. Automatic photo pop-up. *ACM transactions on graphics (TOG)*, ACM, v. 24, n. 3, p. 577–584, 2005.

JAIN, A. K. Fundamentals of digital image processing. Prentice-Hall, Inc., 1989.

KARGER, D. R. Minimum cuts in near-linear time. *Journal of the ACM (JACM)*, ACM, v. 47, n. 1, p. 46–76, 2000.

LANCICHINETTI, A.; FORTUNATO, S. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, APS, v. 80, n. 1, p. 016–118, 2009.

LEVINSHTEIN, A.; DICKINSON, S. J.; SMINCHISESCU, C. Multiscale symmetric part detection and grouping. In: IEEE. *ICCV*. Kyoto, 2009. p. 2162–2169.

LEVINSHTEIN, A. et al. Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 31, n. 12, p. 2290–2297, 2009.

LIANG, Z.; MACFALL, J. R.; HARRINGTON, D. P. Parameter estimation and tissue segmentation from multispectral mr images. *IEEE transactions on Medical Imaging*, IEEE, v. 13, n. 3, p. 441–449, 1994.

LOPES, F. M. *Um modelo perceptivo de limiarização de imagens digitais*. Dissertação de Mestrado em Informática — Universidade Federal do Paraná, 2003.

MARTIN, D. et al. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: ICPR. *Proc. 8th Int'l Conf. Computer Vision*. Nanjing, 2001. v. 2, p. 416–423.

MILGRAM, S. The small world problem. *Psychology today*, New York, v. 2, n. 1, p. 60–67, 1967.

NEUBERT, P.; PROTZEL, P. Superpixel benchmark and comparison. *Chemnitz University of Technology, Department of Electrical Engineering and Information Technology*, p. 1–12, 2012.

NEWMAN, M. E. The structure and function of complex networks. *SIAM review*, SIAM, v. 45, n. 2, p. 167–256, 2003.

NEWMAN, M. E.; GIRVAN, M. Finding and evaluating community structure in networks. *Physical review E*, APS, v. 69, n. 2, p. 026–113, 2004.

NEXUS, M. B. Small worlds and the groundbreaking science of networks. Norton Publishing, 2002.

OLIVEIRA, T. B. de; ZHAO, L. Complex network community detection based on swarm aggregation. In: IEEE. *2008 Fourth International Conference on Natural Computation*. Jinan, 2008. v. 7, p. 604–608.

OLIVEIRA, T. B. S. d. *Clusterizaçao de dados utilizando técnicas de redes complexas e computaçao bioinspirada*. Tese (Doutorado) — Universidade de São Paulo, 2008.

ORMAN, G. K.; LABATUT, V.; CHERIFI, H. Qualitative comparison of community detection algorithms. In: SPRINGER. *International conference on digital information and communication technology and its applications*. Konya, 2011. p. 265–279.

OTSU, N. A threshold selection method from gray-level histograms. *Automatica*, v. 11, n. 285-296, p. 23–27, 1975.

PARKER, J. R. Algorithms for image processing and computer vision. John Wiley & Sons, 2010.

PHAM, D. L.; XU, C.; PRINCE, J. L. Current methods in medical image segmentation. *Annual review of biomedical engineering*, v. 2, n. 1, p. 315–337, 2000.

PORTO, S. M. *Metodologia para a evolução de comunidades em redes complexas dinâmicas*. Dissertação de Mestrado em Computação Aplicada — INPE, 2014.

PRATT, W. Digital image processing. John Wiley & Sons, 1991.

RAGHAVAN, U. N.; ALBERT, R.; KUMARA, S. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, APS, v. 76, n. 3, p. 036106, 2007.

REN, X.; MALIK, J. Learning a classification model for segmentation. In: IEEE. Santiago, 2003. v. 01, p. 10–17.

ROCHA, L. E. C. d. *Redes acopladas: estrutura e dinâmica*. Tese (Doutorado) — Universidade de São Paulo, 2007.

RODRIGUES, F. A. Caracterização, classificação e análise de redes complexas. *Instituto de Fısica de Sao Carlos, Universidade de Sao Paulo, Sao Carlos*, 2007.

RUBINOV, M.; SPORNS, O. Complex network measures of brain connectivity: uses and interpretations. *Neuroimage*, Elsevier, v. 52, n. 3, p. 1059–1069, 2010.

SAKAI, T.; IMIYA, A. Fast spectral clustering with random projection and sampling. In: SPRINGER. *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. New York, 2009. p. 372–384.

SARATH, D. S. *Medição Automática do Efeito de Herbivoria em Folhas de Soja Utilizando Técnicas de Segmentação e Aprendizagem Supervisionada*. Tese (Doutorado), 2014.

SCHANDA, J. Current cie work to achieve physiologically-correct color metrics. *Color vision: Perspectives from different disciplines*, Citeseer, p. 307–318, 1998.

SEIXAS, F. L. et al. Avaliação dos métodos para a segmentação automática dos tecidos do encéfalo em ressonância magnética. *SPOLM 2008*, 2008.

SHI, J.; MALIK, J. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, IEEE, v. 22, n. 8, p. 888–905, 2000.

STROGATZ, S. H. Exploring complex networks. *Nature*, Nature Publishing Group, v. 410, n. 6825, p. 268–276, 2001.

TORRES, A. S. A. *Segmentação de imagens médicas visando a construção de modelos médicos*. Tese (Doutorado) — Instituto Politécnico de Bragança, Escola Superior de Tecnologia e Gestão, 2012.

TRUDEAU, R. J. Introduction to graph theory (corrected, enlarged republication. ed.). New York: Dover, 1993.

TUNG, F.; WONG, A.; CLAUSI, D. A. Enabling scalable spectral clustering for image segmentation. *Pattern Recognition*, Elsevier, v. 43, n. 12, p. 4069–4076, 2010.

UMBAUGH, S. E. *Digital image processing and analysis: Human and Computer Vision Applications with CVIPtools*. [S.l.]: CRC press, 2016.

VEDALDI, A.; SOATTO, S. Quick shift and kernel methods for mode seeking. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2008. p. 705–718.

WALT, S. van der et al. scikit-image: image processing in Python. *PeerJ*, v. 2, p. e453, 6 2014. ISSN 2167-8359. Disponível em: <http://dx.doi.org/10.7717/peerj.453>.

WANG, S. et al. Superpixel tracking. In: IEEE. *2011 International Conference on Computer Vision*. Barcelona, 2011. p. 1323–1330.

WATANABE, C. Y. V. *Métodos de apoio ao diagnóstico médico por imagens usando regras de associação e redes complexas*. Tese (Doutorado) — Universidade de São Paulo, 2013.

WATTS, D. J. Networks, dynamics, and the small-world phenomenon. *American Journal of sociology*, JSTOR, v. 105, n. 2, p. 493–527, 1999.

WATTS, D. J. *Small worlds: the dynamics of networks between order and randomness*. [S.l.]: Princeton university press, 1999.

WATTS, D. J.; STROGATZ, S. H. Collective dynamics of small-world networks. *Nature*, Nature Publishing Group, v. 393, n. 6684, p. 440–442, 1998.

YAN, D.; HUANG, L.; JORDAN, M. I. Fast approximate spectral clustering. In: ACM. *15th ACM SIGKDD international conference on Knowledge discovery and data mining*. Paris, 2009. p. 907–916.