
Segmentação de imagens de alta dimensão por
meio de algoritmos de detecção de comunidades e
super pixels

Oscar Alonso Cuadros Linares

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Segmentação de imagens de alta dimensão por meio de algoritmos de detecção de comunidades e super pixels

Oscar Alonso Cuadros Linares

Orientador: Prof. Dr. João do Espírito Santo Batista Neto

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

USP – São Carlos
Junho de 2013

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

C961s Cuadros Linares, Oscar Alonso
Segmentação de imagens de alta dimensão por meio
de algoritmos de detecção de comunidades e super
pixels / Oscar Alonso Cuadros Linares; orientador
João do Espírito Santo Batista Neto. -- São Carlos,
2013.
66 p.

Dissertação (Mestrado - Programa de Pós-Graduação em
Ciências de Computação e Matemática Computacional) --
Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, 2013.

1. Segmentação de Imagens. 2. Redes Complexas. 3.
Super Pixels. 4. Detecção de Comunidades. I. Batista
Neto, João do Espírito Santo, orient. II. Título.

À Deus e nossa Santa Mãe Maria

Agradecimentos

Ao meu orientador Prof. Dr. João Batista Neto, suas orientações e amizade foram de vital importância para o sucesso do meu mestrado.

Ao professor Francisco Rodrigues, seus aportes foram muito importantes para nosso trabalho.

Aos meus pais Oscar e Ruth, seu amor, apoio e fê em mim, são o pilar da minha vida.

À minha irmã Alejandra por ser a luz da nossa família, principalmente, na minha ausência.

À Aurea Soriano Vargas, minha noiva, o maior presente de Deus.

À Glenda Botelho, nossa amizade e as inúmeras horas de trabalho em equipe, foram a fórmula do sucesso deste mestrado .

À CPNq, pelo apoio financeiro.

Tanta coisa por fazer e cada um tem sua própria tarefa na façanha de nosso tempo.
Santíssima Mãe, interceda para que eu receba força e o encorajamento para cooperar
com a grande tarefa de mudar nosso mundo colocando meu grão de areia, que poderia
muito bem fazer a diferença.

Amém.

Resumo

Segmentação de imagens é ainda uma etapa desafiadora do processo de reconhecimento de padrões. Entre as abordagens de segmentação, muitas são baseadas em particionamento em grafos, as quais apresentam alguns inconvenientes, sendo um deles o tempo de processamento muito elevado. Com as recentes pesquisas na teoria de redes complexas, as técnicas de reconhecimento de padrões baseadas em grafos melhoraram consideravelmente. A identificação de grupos de vértices pode ser considerada um processo de detecção de comunidades de acordo com a teoria de redes complexas. Como o agrupamento de dados está relacionado com a segmentação de imagens, esta também pode ser abordada através de redes complexas. No entanto, a segmentação de imagens baseado em redes complexas apresenta uma limitação fundamental, que é o número excessivo de nós na rede. Neste trabalho é proposta uma abordagem de redes complexas para segmentação de imagens de grandes dimensões que é ao mesmo tempo precisa e rápida. Para alcançar este objetivo, é incorporado o conceito de Super Pixels, visando reduzir o número de nós da rede. Os experimentos mostraram que a abordagem proposta produz segmentações de boa qualidade em baixo tempo de processamento. Além disso uma das principais contribuições deste trabalho é a determinação dos melhores parâmetros, uma vez que torna o método bastante independente dos parâmetros, o que não fora alcançado antes em nenhuma pesquisa da área.

Palavras-chave: Segmentação de Imagens, Redes Complexas, Super Pixels, Detecção de Comunidades.

Abstract

Image segmentation is still a challenging stage of the pattern recognition process. Amongst the various segmentation approaches, some are based on graph partitioning, many of which show some drawbacks, such as the high processing times. Recent trends on complex network theory have contributed considerably to the development of graph-based pattern recognition techniques. The identification of group of vertices can be considered a community detection process according to complex network theory. Since data clustering is closely related to image segmentation, image segmentation tasks can also be tackled by complex networks. However, complex network-based image segmentation poses a very important limitation: the excessive number of nodes of the underlying network. In this work we propose a approach based on complex networks suitable for the segmentation of image with large dimensions that is accurate and yet fast. To accomplish that, we have incorporated the concept of Super Pixels aiming at reducing the number of the nodes in the network. The results have shown that the proposed approach delivered accurate image segmentation within low computational times. Another contribution worth mentioning is the determination of the best values for the parameters needed by the underlying graph-based segmentation and community detection algorithms, which enabled the proposed approach to become less dependent on the parameters. To the best of our knowledge, this is a new contribution to the field.

Key-words: Image Segmentation, Complex Networks, Super Pixels, Community Detection Algorithms.

Sumário

Resumo	vii
Abstract	ix
Sumário	xi
Lista de Figuras	xiii
Lista de Tabelas	xv
Lista de Siglas	xvii
1 Introdução	1
1.1 Motivação e Objetivo	2
1.2 Organização	4
2 Redes Complexas	7
2.1 Teoria de Grafos	8
2.2 Detecção de Comunidades	9
2.2.1 Modularidade	10
2.2.2 <i>Fast Greedy</i>	11
2.2.3 <i>Label Propagation</i>	13
2.2.4 Considerações Finais	15
3 Super Pixel	17
3.1 <i>Turbo Pixels</i>	18
3.2 <i>Simple Linear Iterative Clustering</i>	20
3.3 <i>Speeded-up Turbo Pixels</i>	22
3.3.1 <i>Quadtree Speeded-up Turbo Pixels</i>	23
3.4 Considerações Finais	24
4 Fundamentos Complementares	27
4.1 <i>Local Binary Pattern</i>	27
4.2 Métrica de similaridade para Segmentação de Imagens	29
5 Metodologia	33
5.1 Cálculo dos Super Pixels	33
5.2 Geração do Grafo	34
5.2.1 Intensidade Média	35

5.2.2	Modelo de Cor	35
5.2.3	Textura	36
5.3	Segmentação por meio de Redes Complexas	36
5.4	Avaliação dos Resultados	37
5.4.1	Avaliação da Qualidade	37
5.4.2	Avaliação do Tempo de Processamento	39
6	Resultados	41
6.1	Experimento 1: Imagem Sintética	42
6.2	Experimento 2: Flor de Lotus	44
6.3	Experimento 3: Textura	46
6.3.1	Imagens Sintéticas	46
6.3.2	Estrela do Mar	47
6.4	Experimento 4: <i>Quadtree Speeded-up Turbo Pixels</i>	48
6.5	Experimento 5: Avaliação Qualitativa	51
6.5.1	Parâmetro Raio	51
6.5.2	Parâmetro Limiar	52
6.5.3	Parâmetros do método <i>Speeded-up Turbo Pixels</i>	53
6.6	Experimento 6: <i>Graph Based Image Segmentation</i>	54
6.7	Considerações Finais	54
7	Conclusões	57
7.1	Limitações	58
7.2	Contribuições	58
7.3	Trabalhos Futuros	59
Referências		66

Lista de Figuras

2.1	Rede simples com três comunidades incluídas nos círculos pontilhados, as quais têm uma alta densidade de conexões internas e baixa densidade de conexões externas (Fortunato e Castellano, 2009).	8
2.2	Uma árvore hierárquica ou dendrograma ilustrando o tipo de saída gerada pelo algoritmo <i>Fast Greedy</i> . Os círculos na parte inferior da figura representam os vértices individuais da rede. Os vértices se unem para formar comunidades cada vez maiores, como indicado pelas linhas, até chegar ao topo, onde todos estão unidos em uma única comunidade. Um corte transversal da árvore em qualquer nível, tal como indicado pela linha tracejada, dará as comunidades nesse nível.	11
2.3	Nós são atualizados um a um da esquerda para a direita. Devido a uma alta densidade de arestas (mais alto possível, neste caso), todos os nós terminam com o mesmo rótulo.	13
2.4	Exemplo de uma rede bipartida em que os rótulos das duas partes são disjuntos. Neste caso, devido as escolhas feitas pelos nós no passo t , os rótulos dos nós oscilam entre a e b	14
3.1	Super pixels convexos e de tamanho quase uniforme.	18
3.2	Passos do algoritmo Turbo Pixels.	19
3.3	Duas imagens de tamanho (481×321)	20
3.4	O algoritmo repete iterativamente o processo de associar pixels com o centróide mais próximo até a convergência.	21
3.5	Duas imagens de tamanho (481×321)	22
3.6	Os pixels nas bordas (cor vermelha) são atualizados em cada iteração do algoritmo STP.	23
3.7	Duas imagens de tamanho (481×321)	24
3.8	Exemplo da árvore <i>Quadtree</i> após uma iteração.	24
3.9	Duas imagens de tamanho (800×600)	25
4.1	Exemplo do cálculo do código LBP para $P = 8, R = 1$	28
4.2	Os códigos binários da parte superior (a - e) ilustram padrões uniformes $U \leq 2$, e os códigos da parte de inferior (f - h) ilustram padrões não uniformes $U > 2$	29
4.3	Duas segmentações S e S' divididas em R e R' regiões respectivamente. .	31

5.1	Abordagem de redes complexas para segmentação de imagens combinado com super pixels.	34
5.2	Região (área amarela) circular de raio $R = 5$	35
5.3	As segmentações (a) e (b) mostram a subjetividade do processo.	38
5.4	Seleção da segmentação de referência, ilustrada na imagem (j) e ampliada na imagem (b)	39
6.1	Diferentes resoluções de super pixels. (a) Imagem original; (b) Convergência 10×10 super pixels; (c) Convergência 20×20 super pixels; (d) Convergência 50×50 super pixels; (e) Convergência 100×100 super pixels e (f) Convergência 150×150 super pixels.	42
6.2	Imagen Sintética: a) Imagen segmentada para todos os super pixels e b) Gráfico do tempo de processamento versus o tamanho dos super pixels.	43
6.3	Segmentação da Flor de Lotus. (a) Imagem original de tamanho 639×430 ; (b) Super pixels de 10×10 ; (c) Super pixels de 20×20 ; (e) Super pixels de 50×50 ; (e) Segmentação para os super pixels de 10×10 e 20×20 e (f) Segmentação para os super pixels de 50×50 . O componente do fundo verde, e a flor em rosa.	45
6.4	Segmentação de Flor de Lotus para super pixels de tamanho 10×10 , limiar = 0.91 e raio = 2. (a) Pétalas (b) Fundo, dois componentes e (c) Quarta comunidade, correspondente aos órgãos sexuais da Flor.	46
6.5	Segmentação de duas imagens por meio da técnica <i>Local Binary pattern</i>	47
6.6	Segmentação de uma imagem real por meio da técnica <i>Local Binary Pattern</i>	48
6.7	Experimento 1: Comparação do processo de geração de super pixels mediante a técnica <i>Quadtree Speeded-up Turbo Pixels</i> (b, c, d) e a técnica <i>Speeded-up Turbo Pixels</i> original (e, f, g). Além do resultado do algoritmo <i>Fast Greed</i> para ambas técnicas. (STP = Speeded-up Turbo Pixels) .	49
6.8	Experimento 2: Comparação do processo de geração de super pixels mediante a técnica <i>Quadtree Speeded-up Turbo Pixels</i> (b, c, d) e a técnica <i>Speeded-up Turbo Pixels</i> original (e, f, g). Além do resultado do algoritmo <i>Fast Greed</i> para ambas técnicas. (STP = Speeded-up Turbo Pixels) .	50
6.9	Histogramas gerados para as qualidades mais altas variando os valores do raio. O eixo x é a qualidade no intervalo de 0 a 1.	52
6.10	Histogramas gerados para os limiares usados nos resultados com a maior qualidade, variando os valores do raio. O eixo x são os valores dos limiares no intervalo de 0 a 40.	53
6.11	A primeira coluna mostra as imagens originais, a segunda coluna nossos resultados e a terceira coluna os resultados gerados pelo método <i>Graph Based Image Segmentation</i>	55
7.1	Duas colunas de imagens. (Imagen Original - Segmentação).	61
7.2	Três colunas de imagens. (Imagen Original - Segmentação).	62

Lista de Tabelas

6.1	Parâmetros usados na segmentação da imagem sintética, para distintos tamanhos de super pixels.	43
6.2	Tempo computacional em segundos usado na segmentação da imagem sintética, para distintos tamanhos de super pixels.	44
6.3	Parâmetros usados na segmentação da Flor de Lotus.	45
6.4	Tempo de processamento para a Flor de Lotus.	45
6.5	Tempos de processamento em segundos para os experimentos usando textura.	48
6.6	Tempos de processamento em segundos para comparar o método <i>Speeded-up Turbo Pixels</i> original e nossa proposta <i>Quadtree Speeded-up Turbo Pixels</i> . (SP = Super pixels, FG=Fast Greed).	50
6.7	Média e Desvio Padrão dos histogramas da Figura 6.10.	53

Lista de Siglas

LP	Label Propagation
FG	Fast Greedy
SP	Super Pixel
TP	Turbo Pixels
SLIC	Simple Linear Iterative Clustering
STP	Speeded-up Turbo Pixels
QSTP	Quadtree Speeded-up Turbo Pixels
LBP	Local Binary Pattern
MSSI	Métrica de similaridade para Segmentação de Imagens
GBIS	Graph Based Image Segmentation
SSMGD	Spectral Segmentation with Multiscale Graph Decomposition



CAPÍTULO

1

Introdução

A segmentação de imagens é a operação que permite a extração de subconjuntos de pixels com propriedades semelhantes a partir de uma imagem, e 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 ou as regiões de interesse de uma aplicação foram detectadas. A segmentação de imagens não triviais, como por exemplo cenas naturais, é uma das tarefas mais difíceis no processamento de imagens. Em muitas aplicações a precisão da segmentação determina o sucesso ou fracasso final dos procedimentos de análise computadorizada (Gonzalez e Woods, 2009).

Segmentar imagens tem sido um dos problemas mais populares em visão computacional nos últimos anos. Em imagens médicas, por exemplo, os procedimentos de segmentação de imagens podem ser utilizados para o diagnóstico, permitindo a localização de tumores e outras patologias (Noble e Boukerroui, 2006). Além disso, os métodos de segmentação de imagens podem ser aplicados a sistemas de controle de tráfego e de localização de objetos em imagens de satélites (Zhang et al., 2008). Diferentes algoritmos têm sido propostos para a segmentação de imagens, tais como os baseados em limiarização ou histogramas (Navon et al., 2005), métodos baseados em crescimento de regiões (Haralick e Shapiro, 1985) e métodos baseados em grafos, os quais têm-se tornado populares na última década (Shi e Malik (2000), Cour et al. (2005)).

Estes métodos geralmente criam um grafo G onde cada nó representa um pixel na imagem de entrada e os pesos das arestas são computadas de acordo com uma função de similaridade. Esta função estabelece certa relação entre cada par de pixels e é, geral-

mente, baseada na Distância Euclidiana, Manhattan, Gaussiana, entre outras. No entanto, a maioria dos métodos baseados em grafos apresentam alguns inconvenientes, tais como: tempo de processamento muito alto, consumo de memória excessivo, apenas conseguem segmentar imagens pequenas, além de não fornecer segmentações precisas e, em geral, definir as partições em c (constante fixa) grupos ao invés de um número arbitrário destes.

As técnicas de reconhecimento de padrões baseadas em grafos têm sido aperfeiçoadas com o desenvolvimento da teoria de redes complexas (de Oliveira e Zhao (2008), Rodrigues et al. (2011)). A identificação de grupos de vértices com grande concentração de arestas entre eles e a baixa concentração de arestas entre esses grupos, pode ser realizada pelos métodos de detecção de comunidades fornecidos pela teoria de redes complexas. Essas técnicas fornecem partições mais precisas que os métodos tradicionais baseados em grafos, métodos como o *Graph partitioning*, *Hierarchical clustering*, *Partitional clustering* e *Spectral clustering* (Fortunato, 2010). A detecção de comunidades procura encontrar grupos que possuem uma inerente ou externa noção específica de similaridade entre os nós no interior dos grupos, ou seja, grupos com nós densamente conectados entre eles. Além disso, o número de comunidades na rede e o seu tamanho não é conhecido de antemão. Eles são estabelecidos pelo algoritmo de detecção de comunidades (Raghavan et al., 2007).

Neste contexto, este trabalho trata de segmentação de imagens de alta dimensão por meio de Redes Complexas. Além disso, propõe uma pré-segmentação da imagem visando reduzir a cardinalidade da rede por meio do conceito de Super Pixels.

1.1 Motivação e Objetivo

Uma vez que o agrupamento de dados está estritamente relacionado com a segmentação de imagens, é possível considerar tais métodos para a identificação de objetos em uma imagem. Mais especificamente, uma imagem pode ser mapeada em um grafo e, assim, a detecção de comunidades pode ser considerada para identificar os objetos que correspondem às comunidades na rede.

Porém, a segmentação de imagens baseada em redes apresenta uma limitação fundamental. Cada imagem de tamanho $M \times M$ é mapeada em uma rede composta de M^2 nós, onde cada nó representa um pixel (Chalumeau et al., 2007). Desta forma, apenas pequenas imagens podem ser consideradas, uma vez que a maioria dos métodos de identificação de comunidades são computacionalmente caros, além da grande quantidade de memória principal usada para criar a rede.

Na literatura encontra-se alguns trabalhos relacionados à abordagem descrita para criar o grafo, onde cada pixel representa um nó na rede. Um destes trabalhos é o método

Graph Based Image Segmentation (GBIS) proposto por Felzenszwalb e Huttenlocher (2004). Este método segmenta a imagem de acordo com um critério para avaliar se existe ou não uma borda entre dois componentes em uma região. Este critério calcula a dissimilaridade, não negativa, entre dois componentes vizinhos, a qual é representada no grafo como o peso da aresta que conecta dois nós. Neste método, o grafo é criado conectando-se apenas os pixels vizinhos na imagem, deste modo, a quantidade de nós no grafo é igual à quantidade de pixels na imagem. Em cada passo do algoritmo é comparado o peso da aresta do nó atual com seus vizinhos. Se o peso é menor que um limiar k estes nós são unidos em um segmento. Após unir dois nós, é aplicado um processo que avalia a similaridade do segmento no qual foram inseridos. As funções de similaridade que usa este método são baseadas na intensidade da cor dos pixels. Por outro lado, este algoritmo realiza um pré-processamento e um pós-processamento para melhorar o resultado. O pré-processamento consiste em aplicar um filtro Gaussiano sobre a imagem, a fim de eliminar o ruído. O pós-processamento é aplicado para controlar o menor tamanho dos segmentos e evitar a presença de segmentos muito pequenos, o menor tamanho é um parâmetro min fornecido pelo usuário. Neste pós-processo, quando é detectada a presença de um segmento muito pequeno, este é unido a um segmento vizinho, por meio da função de dissimilaridade, mas forçando a fusão do segmento. No total, o algoritmo precisa de três parâmetros: σ para o filtro Gaussiano, o limiar k para controlar a similaridade dos pixels e o min para forçar o menor tamanho das regiões.

Um outro trabalho estudado é o *Spectral Segmentation with Multiscale Graph Decomposition* (SSMGD) proposto por Cour et al. (2005). Este método segmenta a imagem com base na abordagem *Normalized Cut Graph Partitioning Framework* apresentada por (Shi e Malik, 2000). O algoritmo cria um grafo conectando todos os pares de pixels (i, j) cuja distância entre eles seja menor a um raio G_r , fornecido pelo usuário. O peso das arestas é determinado por uma função $W(i, j)$ a qual calcula a probabilidade de um pixel i e um pixel j pertençam à mesma região na imagem. Este cálculo é obtido com base na intensidade da cor e a distância entre os pixels. Uma vez criado o grafo, é aplicado o algoritmo de corte em grafos, o qual divide o grafo recursivamente em dois grupos disjuntos de vértices, mediante a remoção das arestas que conectam as duas partes. O critério usado para dividir o grafo é calcular o grau de dissimilaridade *cut* entre os grupos, por meio da soma dos pesos das arestas a serem removidas. Uma ótima divisão do grafo é a que minimiza o valor *cut*. Este algoritmo apresenta a desvantagem de ser semi-supervisionado, porque requer um parâmetro que indique o número de regiões na imagem. Além disso, o tempo de processamento usado para segmentar uma imagem é muito alto (Cuadros et al., 2012).

Neste contexto, o objetivo deste trabalho é segmentar imagens de alta dimensão em

um tempo aceitável, usando algoritmos de detecção de comunidades em redes complexas, mediante a representação das imagens como grafos. Para superar a limitação de gerar redes com uma quantidade muito alta de nós, neste projeto, propomos uma pre-segmentação da imagem, chamada de Super Pixel. Segmentar previamente uma imagem por meio de super pixels (que é um agrupamento de vários pixels semelhantes) permite uma redução drástica no número de pixels de uma imagem. Portanto, o tempo de processamento para a segmentação da imagem é reduzido e imagens de alta dimensão podem ser processadas.

Outro objetivo do trabalho foi solucionar o problema dos parâmetros empregados na abordagem, uma vez que as técnicas até então exigiam muitos parâmetros, além de não fornecer os melhores valores ou critérios para estabelecer os. Como será mostrado no decorrer desta dissertação, conseguimos reduzi-los a um número aceitável e determinar seus melhores valores.

1.2 Organização

Esta dissertação está organizada, do seguinte modo:

- O Capítulo 2 introduz alguns conceitos importantes sobre redes complexas e apresenta duas técnicas para a detecção de comunidades em redes. Técnicas que permitem encontrar grupos (comunidades) que têm uma noção de similaridade entre seus nós.
- O Capítulo 3 apresenta técnicas recentes para a geração de super pixels a partir de imagens. Estas técnicas pré-segmentam uma imagem em grupos de pixels com propriedades similares, representando a imagem com um número reduzido de pixels representativos. Por consequência é reduzido o custo computacional das tarefas subsequentes sobre a imagem.
- No Capítulo 4 são apresentadas duas técnicas importantes para este trabalho, que por questões de contexto não caberiam nos Capítulos 2 ou 3. A primeira é uma técnica para medir textura chamada *Local Binary Pattern* (LBP) usada neste projeto para extrair a textura de cada super pixel no momento da criação do grafo. A segunda é um nova métrica para comparar segmentações. Esta nova métrica é usada para avaliar os resultados obtidos por meio da comparação com segmentações feitas por diversos usuários, chamadas neste trabalho, segmentações manuais. Merece destaque mencionar que esta nova métrica foi desenvolvida e proposta por nós neste trabalho.
- O Capítulo 5 apresenta a metodologia adotada neste trabalho.

- No capítulo 6 apresentam-se os experimentos realizados, ilustrando os principais aspectos da abordagem proposta.

CAPÍTULO

2

Redes Complexas

Uma rede é um grupo de itens chamados *vértices* ou *nós*, com conexões entre eles, chamadas *arestas* (Newman, 2003). Várias relações do mundo real podem ser representadas como uma rede (Reka e Barabási, 2002), por exemplo, uma célula pode ser descrita como uma rede de substâncias químicas ligadas por reações químicas. A Internet é uma rede de roteadores e computadores conectados por cabos ou dispositivos sem fios. Modismos e ideias espalhadas nas redes sociais, cujos nós são seres humanos e cujas arestas representam diferentes relações sociais é um outro exemplo que pode ser modelado como uma rede. As redes descritas nestes exemplos compartilham características como: possuir uma quantidade muito alta de nós, modelar relações do mundo real e representar complexos padrões de conexões. Porém, estas estruturas são mais dinâmicas, podem mudar no tempo e não são aleatórias. Neste tipo de redes, muda o enfoque de estudo e na literatura especializada são chamadas Redes Complexas.

Muitas redes complexas possuem uma característica interessante: a presença de grupos de nós muito conectados e, ao mesmo tempo, poucas conexões entre estes diferentes grupos. Tais grupos são chamados *comunidades* (de Oliveira et al., 2008), a Figura 2.1 mostra um grafo simples com três comunidades. Não há uma definição de comunidade universalmente aceita, porque a definição depende da aplicação específica que estamos trabalhando ou a aplicação que se tem em mente, mas é natural assumir que as comunidades são grupos de vértices parecidos (Fortunato, 2010). Neste trabalho assumimos a noção de que deve existir mais arestas entre os nós dentro de uma comunidade, que arestas entre os nós da comunidade e o restante do grafo.

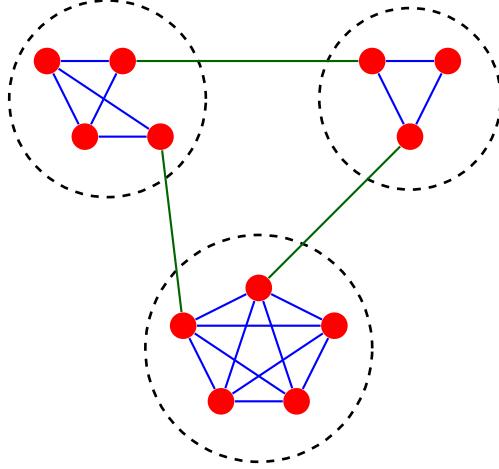


Figura 2.1: Rede simples com três comunidades incluídas nos círculos pontilhados, as quais têm uma alta densidade de conexões internas e baixa densidade de conexões externas (Fortunato e Castellano, 2009).

Usa-se o termo *rede* para referir-se informalmente aos objetos compostos por elementos e conexões entre esses elementos, tais como os exemplos descritos no primeiro parágrafo. O conceito matemático para modelar redes é dado pela teoria de grafos. A seguir é apresentada uma seção com os fundamentos básicos da teoria de grafos e uma seção de detecção de comunidades, além de duas técnicas importantes, *Fast Greedy* (Clauset et al., 2004) e *Label Propagation* (Raghavan et al., 2007).

2.1 Teoria de Grafos

Um *grafo* $G(V, E)$ é um objeto abstrato formado por um grupo V de vértices (nós) e um grupo E de arestas que unem (conectam) pares de vértices (Brandes e Erlebach, 2005). O grupo de vértices e o grupo de arestas de um grafo G é definido como $V(G)$ e $E(G)$, respectivamente. Se dois vértices estão conectados por uma aresta, eles são *adjacentes*, e podem ser chamados *vizinhos*. Os grafos podem ser *dirigidos* ou *não dirigidos*. Uma aresta não dirigida u , conectando dois vértices $v \in V$ é denotada por $\{u, v\}$. Nos grafos dirigidos, cada aresta dirigida tem uma *origem* e um *destino*. Uma aresta com origem $u \in V$ e destino $v \in V$ é representada por um par ordenado (u, v) . Para reduzir a notação, uma aresta $\{u, v\}$ ou (u, v) pode ser denotada por uv .

Muitas vezes é útil associar valores numéricos (pesos) com as arestas ou os vértices de um grafo $G = (V, E)$. Neste trabalho apenas são considerados os grafos com pesos nas arestas. Os pesos das arestas podem ser representados como uma função $w : E \rightarrow \mathbb{R}$, que atribui a cada aresta $e \in E$ um peso $w(e)$. Dependendo do contexto, os pesos das arestas podem descrever muitas propriedades, tais como o custo (tempo de viagem ou distância),

capacidade ou similaridade. Em geral, um grafo não ponderado $G = (V, E)$ é equivalente a um grafo ponderado com peso de arestas $w(e) = 1$.

O *grau* de um vértice v em um grafo não dirigido $G = (V, E)$, denotado por $d(v)$, é o número de arestas em E que têm v como vértice final. Um grafo não dirigido é chamado de *regular* se todos os vértices têm o mesmo grau e *r-regular* se o grau é igual a r .

Tanto para grafos dirigidos quanto para grafos não dirigidos, podemos permitir que um grupo de arestas E tenham a mesma aresta várias vezes. Se uma aresta aparece várias vezes em E , as cópias da aresta são chamadas *arestas paralelas*. Grafos com arestas paralelas são chamados de *multigrafos*. Um grafo é chamado de *simples*, se seus vértices estão apenas uma vez em E .

2.2 Detecção de Comunidades

Tipicamente as redes têm partes onde os nós estão mais fortemente conectados uns com os outros do que com o resto da rede. Esses conjuntos de nós, são geralmente chamados de grupos (*clusters*), comunidades, grupos coesos ou módulos (Palla et al., 2005). O objetivo dos algoritmos de detecção de comunidades é encontrar esses grupos de nós em uma rede.

A detecção de comunidades é um problema parecido ao problema de corte em grafos ou particionamento (Karger (2000), Kernighan e Lin (1970), Fiduccia e Mattheyses (1982)). O problema de particionamento em grafos é, em geral, definido como o problema de particionamento em c grupos de tamanho muito parecido ou igual, minimizando o número de arestas entre os grupos. O problema é *NP-hard*, embora existam muitas abordagens heurísticas como o algoritmo *Lin–Kernighan* (Kernighan e Lin, 1970), *Spectral Bisection* (Barnes, 1981), além de outros métodos baseados em algoritmos geométricos ou algoritmos multinível (Póthen, 1997).

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

Dada uma rede com n nós e m arestas $N(n, m)$, qualquer algoritmo de detecção de comunidades é capaz de encontrar subgrupos de nós. Sejam C_1, C_2, \dots, C_p as comunidades encontradas, essas comunidades satisfazem as seguintes propriedades (Fiduccia e Mattheyses, 1982).

1. $C_i \cap C_j = \emptyset$ para $i \neq j$

2. $\bigcup C_i$ abrange o conjunto de nós em N

O problema fundamental relacionado à detecção de comunidades é como definir a melhor divisão da rede em suas comunidades, já que em redes reais normalmente não há informações disponíveis sobre o número e o tamanho das comunidades existentes. Para resolver esse problema, Newman e Girvan (2004) propuseram uma medida, chamada *Modularidade*, que é detalhada a seguir.

2.2.1 Modularidade

A modularidade Q é uma medida originalmente criada para definir um critério de parada para o algoritmo *Newman and Girvan*. A modularidade representa uma das primeiras tentativas de alcançar uma compreensão do princípio do problema de agrupamento em grafos baseada no conceito de comunidade, para desenvolver um modelo geral a partir da expressão “força das comunidades e partições”. A modularidade rapidamente se tornou um elemento essencial de muitos métodos de agrupamento e é uma das funções de qualidade mais utilizadas e mais conhecidas (Fortunato, 2010). A modularidade é definida a seguir.

Considere uma divisão particular de uma rede em k comunidades. Defina também uma matriz simétrica E de tamanho $k \times k$, onde o elemento e_{ij} é a fração de todas as arestas na rede que unem os vértices da comunidade i com os vértices da comunidade j .

O traço dessa matriz $\text{Tr}(E) = \sum_i e_{ii}$ dá a fração de arestas na rede que conectam os vértices da mesma comunidade i , claramente uma boa divisão em comunidades deve ter um valor elevado deste traço.

Então pode ser definida a soma das linhas (ou colunas) $a_i = \sum_j e_{ij}$ que representa a fração de arestas que conectam os vértices da comunidade i com as outras comunidades j . Assim, definimos a medida da modularidade:

$$Q = \sum_i^k (e_{ii} - a_i^2) = \text{Tr } e - ||e|| \quad (2.1)$$

na qual, $||e|| = (\sum_{ij}^k e_{ij})^2$ que indica a soma dos elementos da matriz E . Teoricamente, esta quantidade mede a fração de arestas na rede que conectam vértices da mesma comunidade, menos o valor esperado da mesma quantidade na rede com as mesmas divisões de comunidades, mas com conexões aleatórias entre os vértices. Se o número de arestas em uma comunidade não é maior que o aleatório, então o valor da modularidade é $Q = 0$. Valores que se aproximam a $Q = 1$, o qual é máximo, indicam uma estrutura forte de comunidade. Na prática, valores dessas redes estão aproximadamente entre 0.3 e 0.7. Valores mais altos não são frequentes.

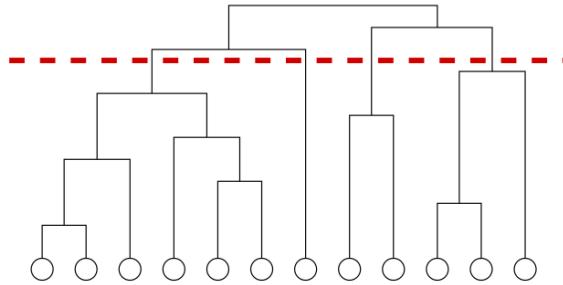


Figura 2.2: Uma árvore hierárquica ou dendrograma ilustrando o tipo de saída gerada pelo algoritmo *Fast Greedy*. Os círculos na parte inferior da figura representam os vértices individuais da rede. Os vértices se unem para formar comunidades cada vez maiores, como indicado pelas linhas, até chegar ao topo, onde todos estão unidos em uma única comunidade. Um corte transversal da árvore em qualquer nível, tal como indicado pela linha tracejada, dará as comunidades nesse nível.

Muitos algoritmos de detecção de comunidades foram desenvolvidos baseados na medida da modularidade, por exemplo, *Fast Greedy algorithm* (Clauset et al., 2004). No entanto, a otimização da modularidade tem um alto custo computacional. Por esse motivo, tem sido desenvolvidos outros algoritmos de baixo custo computacional que não fazem uso da modularidade. Um desses é o algoritmo *Label Propagation* (Raghavan et al., 2007). Detalhes dos algoritmos *Fast Greedy* e *Label Propagation* são apresentados a seguir. Estudos preliminares mostraram que estes algoritmos, frente a outros existentes na literatura, apresentam os melhores resultados para o agrupamento de dados na área de segmentação de imagens (Belizario, 2012).

2.2.2 *Fast Greedy*

O algoritmo *Fast Greedy* foi proposto por Clauset et al. (2004) e é uma otimização do algoritmo proposto por Newman (2004), que utiliza uma técnica gulosa. O algoritmo inicia com cada vértice da rede sendo o único membro de uma comunidade. Iterativamente, novos membros são adicionados. Esta amalgamação produz um maior incremento do valor da modularidade Q . Para uma rede de n vértices, após $n - 1$ uniões, o algoritmo terá apenas uma comunidade, e termina. O método pode ser representado como uma árvore, na qual as folhas são os vértices da rede original e os nós internos, as correspondentes uniões. A árvore é chamada de *dendrograma*, e representa uma decomposição hierárquica da rede em comunidades, como é mostrado na Figura 2.2.

A mais simples implementação dessa ideia é armazenar a matriz de adjacência do grafo como um vetor de inteiros e iterativamente combinar pares de linhas e colunas cada vez que as correspondentes comunidades são combinadas. Esta estratégia usa uma boa quantidade de tempo e espaço de memória. O algoritmo proposto por Clauset et al. (2004)

diminui o tempo de processamento, mediante o uso de estruturas de dados mais sofisticadas, conseguindo um tempo de execução de $O(md \log n)$, onde d é a profundidade do dendrograma, m as arestas e n os vértices. Muitas redes do mundo real são esparsas, de modo que $m \sim n$, além disso para redes com uma estrutura hierárquica $d \sim \log n$. Para essas redes o algoritmo se aproxima do tempo linear, $O(n \log^2 n) \approx O(n)$.

O funcionamento do algoritmo envolve encontrar as mudanças Δ na modularidade Q , o que resulta da fusão de cada par de comunidades e escolher a maior delas e realizar a fusão correspondente. Uma forma de desenvolver este processo é pensar na rede como um multigrafo, onde uma comunidade é representada por um vértice, pacotes de arestas conectam um vértice com outro e as arestas internas são representadas por auto-arestas (*self-edges*). A união de duas comunidades i e j na matriz de adjacência para o multigrafo, é feita substituindo as i^{th} e j^{th} linhas e colunas pela sua soma. No algoritmo de Newman (2004), essa operação é feita 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. Infelizmente, calcular ΔQ_{ij} e encontrar o par i, j com uma ΔQ_{ij} grande, consome muito tempo.

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, já que juntar duas comunidades sem arestas entre elas não produz incremento em Q . Basta apenas armazenar ΔQ_{ij} para os pares i, j que estão sendo juntados por uma ou mais arestas. Além disso o algoritmo usa eficientes estruturas de dados para armazenar valores grandes de ΔQ_{ij} . No total, o algoritmo mantém quatro estruturas de dados.

1. Uma matriz esparsa contendo ΔQ_{ij} para cada par i, j de comunidades com pelo menos uma aresta entre elas.
2. Uma árvore binária balanceada para armazenar cada linha da matriz esparsa.
3. Uma *max-heap* H que contém o maior valor de cada linha da matriz ΔQ_{ij} junto com os rótulos i, j das correspondentes comunidades.
4. Um vetor com os elementos a_i

Como descrito anteriormente, o algoritmo começa com cada vértice sendo o único membro de uma comunidade, na qual a fração de arestas e_{ij} que unem vértices na comunidade i com os vértices da comunidade j é $\frac{1}{2m}$, se i e j estão conectados e 0 caso contrário. E $a_i = \frac{k_i}{2m}$, onde k_i é o grau do vértice. Assim pode-se definir inicialmente:

$$\Delta Q_{ij} = \begin{cases} \frac{1}{2m} - \frac{k_i K_j}{(2m)^2} & \text{se } i, j \text{ estão conectados} \\ 0, & \text{caso contrário} \end{cases} \quad (2.2)$$

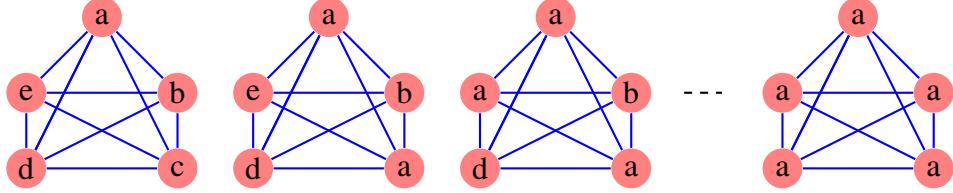


Figura 2.3: Nós são atualizados um a um da esquerda para a direita. Devido a uma alta densidade de arestas (mais alto possível, neste caso), todos os nós terminam com o mesmo rótulo.

Então, para cada comunidade i o algoritmo pode ser definido como segue (Algoritmo 1):

Algoritmo 1: Algoritmo para cada comunidade i

1. Calcular os valores iniciais de ΔQ_{ij} e a_i de acordo com a Equação 2.2, preenchendo a *max-heap* H com o maior elemento de cada linha da matriz ΔQ .
 2. Escolher o maior ΔQ_{ij} da *max-heap* H , juntar a correspondente comunidade, atualizar a matriz ΔQ , a *max-heap* H , o vetor a_i e incrementar Q mediante ΔQ_{ij} .
 3. Repetir o passo 2 até obter-se apenas uma comunidade.
-

2.2.3 Label Propagation

Raghavan et al. (2007) propuseram um algoritmo de detecção de comunidades com complexidade computacional $O(n + m)$, onde n é o número de nós e m é o número de arestas na rede. A principal ideia do algoritmo é a seguinte. Suponha que um nó x tem vizinhos x_1, x_2, \dots, x_k e cada vizinho tem um rótulo que indica a comunidade a que ele pertence. Então x determina sua comunidade baseado nos rótulos de seus vizinhos. O método assume que cada nó na rede escolhe a mesma comunidade à qual pertencem o maior número de seus vizinhos. O algoritmo rotula todos os nós com um rótulo diferente para cada um e deixa que os rótulos se propaguem através da rede. Quando os rótulos se propagam, grupos densamente conectados chegam rapidamente a ter um único rótulo (Figura 2.3). Muitos grupos são criados em toda a rede e eles continuam se expandindo enquanto seja possível. Ao final do processo de propagação, os nós com o mesmo rótulo são agrupados em uma comunidade.

O algoritmo realiza o processo iterativamente, sendo que em cada passo os nós atualizam seus rótulos com base nos rótulos de seus vizinhos. O processo de atualização pode ser síncrono ou assíncrono. Na atualização síncrona, o nó x na t -ésima iteração atualiza o rótulo baseado nos rótulos de seus vizinhos na iteração $t - 1$. Por isso,

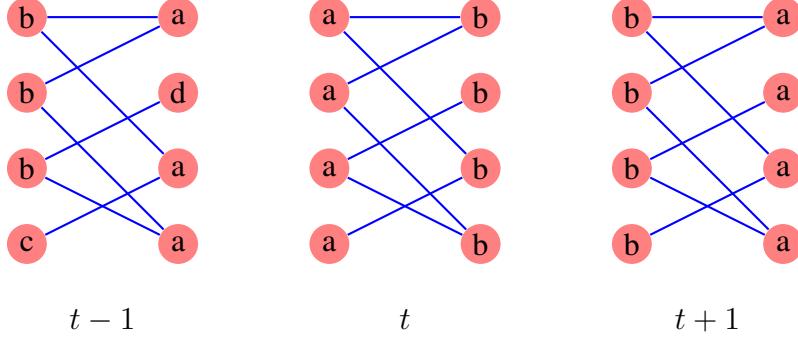


Figura 2.4: Exemplo de uma rede bipartida em que os rótulos das duas partes são disjuntos. Neste caso, devido as escolhas feitas pelos nós no passo t , os rótulos dos nós oscilam entre a e b .

$C_x(t) = f(C_{x_1}(t-1), \dots, C_{x_k}(t-1))$, onde $C_x(t)$ é o rótulo do nó x no tempo t . O problema porém, é que subgrafos na rede que são bipartidas ou quase bipartidas podem levar a oscilações de rótulos (ver Figura 2.4). Isso se apresenta em casos onde as comunidades têm forma de grafo estrela. Por isso, o algoritmo faz uso da atualização assíncrona onde $C_x(t) = f(C_{x_{i1}}(t), \dots, C_{x_{im}}(t), C_{x_{i(m+1)}}(t-1), \dots, C_{x_{ik}}(t-1))$ e x_{i1}, \dots, x_{im} são vizinhos de x que foram atualizados na atual iteração, enquanto $x_{i(m+1)}, \dots, x_{ik}$ são vizinhos que ainda não foram atualizados na iteração atual. A ordem em que todos os n nós na rede são atualizados em cada iteração é definida aleatoriamente. Note que enquanto temos n diferentes rótulos no início do algoritmo, o número de rótulos se reduz durante as iterações, gerando os rótulos das comunidades que compõem a rede.

Idealmente o processo iterativo deveria continuar até que todo nó na rede mude seu rótulo. No entanto, poderia haver nós da rede que possuísssem um número máximo igual de vizinhos em duas ou mais comunidades. Desde que o algoritmo quebra os laços aleatoriamente entre os possíveis candidatos, os rótulos dos nós podem mudar durante as iterações, mesmo que os rótulos de seus vizinhos permaneçam constantes. Por isso o processo iterativo é realizado até que cada nó da rede tenha o rótulo que tem maior ocorrência entre seus vizinhos. Se C_1, \dots, C_p são os rótulos que estão atualmente ativos na rede e $d_i^{C_j}$ é o número de nós vizinhos que o nó i tem com os nós de rótulos C_j , então o algoritmo finaliza quando todo nó i tem o rótulo C_m e $d_i^{C_m} \geq d_i^{C_j} \forall j$.

Ao final do processo iterativo os nós com o mesmo rótulo são agrupados em comunidades. O algoritmo pode ser descrito pelos seguintes passos.

1. Inicializar os rótulos para todos os nós da rede. Para um nó dado x , $C_x = x$.
2. Inicializar $t = 1$
3. Organizar os nós da rede em ordem aleatória e agrupar em X

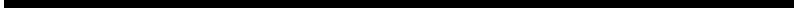
4. Para cada $x \in X$ computar $C_x(t) = f(C_{x_{i1}}(t), \dots, C_{x_{im}}(t), C_{x_{i(m+1)}}(t-1), \dots, C_{x_{ik}}(t-1))$. f aqui retorna o rótulo de maior frequência entre os vizinhos.
5. Se todos os nós têm o rótulo que o máximo número de seus vizinhos tem, então o algoritmo para. Caso contrário, fazer $t = t + 1$ e voltar ao passo (3).

2.2.4 Considerações Finais

O algoritmo *Fast Greedy* tem uma complexidade de ordem $O(md \log n)$ para uma rede com n vértices e m arestas onde d é a profundidade do dendrograma. Para as redes que são hierárquicas, que têm comunidades em muitas escalas e o dendrograma é mais ou menos equilibrado, temos que $d \sim \log n$. Além disso, se a rede é esparsa, $m \sim n$, então a complexidade computacional é $O(n \log^2 n)$. Isto é consideravelmente rápido.

Por outro lado, o algoritmo *Label Propagation* tem uma complexidade $O(n + m)$. Inicializar todos os nós com os seus rótulos requer um tempo $O(n)$. Cada iteração do algoritmo é de tempo linear no número de arestas $O(m)$. Para cada nó x , o método primeiramente agrupa os vizinhos segundo os seus rótulos $O(d_x)$. O processo é repetido para todos os nós e a média do tempo é $O(m)$ para cada iteração. Quando o algoritmo termina é possível que dois ou mais grupos de nós desconectados tenham o mesmo rótulo, o que conduz a que duas diferentes comunidades adotem o mesmo rótulo. Nesses casos, depois que o algoritmo terminar, é executado um simples método de busca em largura nas sub-redes de cada grupo para separar essas comunidades. Isso requer um tempo de $O(n + m)$.

No *Fast Greedy*, a partição que produz a modularidade Q máxima é eleita como a mais representativa da estrutura da comunidade na rede, mas outras partições com valores elevados de Q têm uma estrutura semelhante aquela obtida por meio de Q máxima. Assim, o *Label Propagation* encontra várias soluções, significativamente modulares, que têm uma certa quantidade de dissimilaridade (Raghavan et al., 2007), ou seja, o algoritmo *Label Propagation* pode encontrar comunidades com valores elevados de Q , mas não garante o máximo valor de Q , o que pode produzir resultados não desejados.



CAPÍTULO

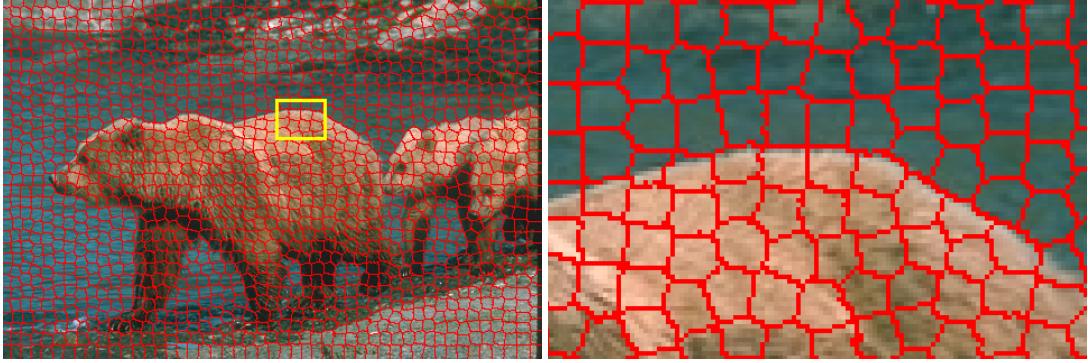
3

Super Pixel

A segmentação de imagens baseada em redes complexas tem uma limitação fundamental. Cada imagem de tamanho $M \times M$ é mapeada em uma rede composta por M^2 nós, onde cada pixel da imagem é mapeado como um nó na rede. Como será mostrado no decorrer deste capítulo, utilizar super pixels fornece uma eficiente e compacta representação da rede por causa da diminuição drástica do número de nós.

No processamento de imagens é usual empregar uma pré-segmentação para reduzir o número de pixels em uma imagem, e por consequência o custo computacional das tarefas subsequentes sobre esta. Super Pixels é uma técnica de segmentação de imagens baseada em regiões e tem o objetivo de representar as imagens com um número limitado de grupos de pixels (Ren e Malik, 2003). Para que a técnica seja útil, esta deve ser rápida, fácil de usar e produzir segmentações de boa qualidade (Achanta et al., 2010). Usualmente os super pixels são convexos e de tamanho quase uniforme, como mostra a Figura 3.1.

Muitos algoritmos têm sido desenvolvidos visando obter super pixels em uma imagem. Um deles é o algoritmo *Turbo Pixels* (Levinshtein et al., 2009) baseado em fluxos geométricos que adaptam a estrutura aos objetos da imagem mediante a dilatação de sementes. Por outro lado, de acordo com o estado da arte, dois métodos merecem destaque: *Simple linear iterative clustering (SLIC)* (Achanta et al., 2010) e *Speeded-up Turbo Pixels* (Cigla e Alatan, 2010). Estes métodos são baseados no algoritmo *k-means* e apresentam a divisão inicial da imagem em regiões retangulares, as quais intercambiam seus pixels até formar a estrutura dos super pixels. Detalhes desses algoritmos, além de uma modificação do algoritmo *Speeded-up Turbo Pixels* que aplica o modelo de cor CIELAB e a estrutura



(a) Visualização dos super pixels em uma imagem (b) Ampliação do quadro amarelo da Figura 3.1.a

Figura 3.1: Super pixels convexos e de tamanho quase uniforme.

de dados *Quadtree*, são apresentados a seguir.

3.1 Turbo Pixels

Esta técnica foi proposta por Levinshtein et al. (2009) e é baseada na dilatação de sementes mediante fluxos geométricos (Bakas, 2005). A ideia básica é criar e dilatar sementes cujas bordas contenham os super pixels. Inicialmente é dado um número constante K de super pixels. A seguir o algoritmo define K áreas circulares (sementes) em uma grade cuja distância entre elas é aproximadamente igual a $\sqrt{\frac{N}{K}}$, onde N é número de pixels na imagem. O raio inicial das sementes é de 1 pixel (Figura 3.2.a). Para a dilatação das sementes o algoritmo usa a seguinte discretização da Equação de evolução de curvas:

$$\Psi^{n+1} = \Psi^n - S_I S_B ||\nabla \Psi^n|| \Delta t \quad (3.1)$$

onde Ψ^n é a coordenada do pixel avaliado na iteração n . Cada aplicação da função 3.1 corresponde a um passo de tempo Δt na evolução das bordas (Figura 3.2.b). O termo principal da equação que controla a evolução é o produto das duas velocidades $S_I S_B$, as quais são os principais parâmetros do algoritmo. O primeiro termo S_I depende da estrutura local da imagem e da geometria dos super pixels em cada ponto de fronteira, e o segundo S_B depende da proximidade dos pontos das borda com os outros super pixels.

Para evitar que os super pixels se sobreponham, é criado o esqueleto homotópico (Ogniewicz e Kübler, 1995) da região ainda não atribuída da imagem, o que é, em geral, o processo de converter um objeto 2D em uma representação linear 1D, parecida a uma *stick figure* (Figura 3.2.c). É importante que o esqueleto possua a informação da forma da região circunscrita e preserve a conectividade. O algoritmo usa um termo binário de parada no esqueleto homotópico, no qual, 0 é o rótulo da região não atribuída $S_B(x, y) = 0$ se e somente se (x, y) está no esqueleto. Esta formulação permite que as bordas de cada

super pixel sejam guiadas até convergir completamente com os outros super pixels. Cada vez que as regiões entre as curvas envolventes mudam a cada iteração do algoritmo, o esqueleto também deve ser atualizado. Isto é realizado rotulando todos os pixels na região ainda não atribuída e aplicando o algoritmo de afinamento *preserving thinning algorithm* (Siddiqi et al., 2002) sobre eles para calcular o esqueleto. O algoritmo de afinamento remove os pixels de acordo com a sua distância com a borda da região com a restrição de que todos os pontos podem ser removidos se a topologia não é alterada.

O termo S_I combina os modelos de difusão e reação com um termo adicional para atrair o fluxo para as bordas:

$$S_I(x, y) = \underbrace{[1 - \alpha\kappa(x, y)]\phi(x, y)}_{\text{reação difusão}} - \underbrace{\beta[N(x, y) \cdot \nabla\phi(x, y)]}_{\text{termo adicional}} \quad (3.2)$$

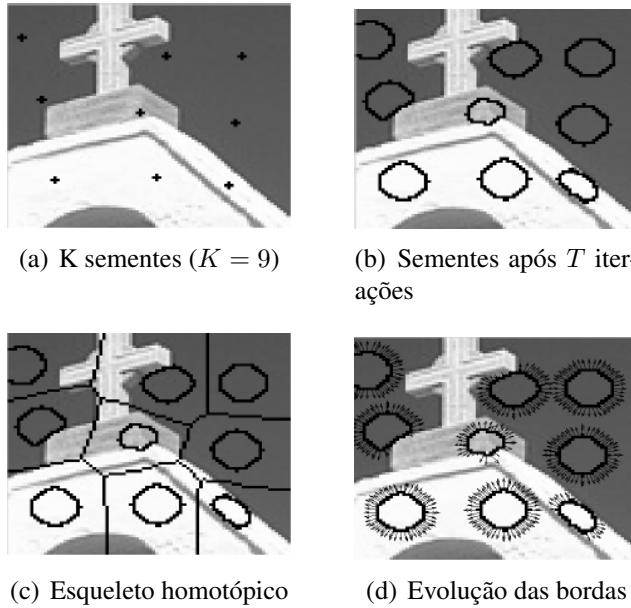


Figura 3.2: Passos do algoritmo Turbo Pixels.

O termo de reação - difusão garante que a evolução das bordas desacelere quando estejam próximas da região de maior gradiente na imagem (Figura 3.2.d). Este termo é controlado por três quantidades: 1) Uma função de afinidade local $\phi(x, y)$ (Levinshtein et al., 2009), calculada para todos os pixels na imagem, a qual é baixa perto das bordas e alta em outra região 2) A função de curvatura $\kappa(x, y)$ (Levinshtein et al., 2009) que expressa a curvatura da bordas em cada ponto (x, y) e suaviza as bordas envolventes e 3) um parâmetro de controle α para a contribuição do termo de curvatura. O termo adicional garante que as bordas sejam atraídas aos contornos da imagem, onde $N(x, y)$ é a normal no ponto (x, y) . Os parâmetros α e β controlam as contribuições relativas dos termos de reação - difusão e o adicional. Maiores valores de α evitam "vazamento" nas bordas, mas



(a) Imagem original

(b) Turbo Pixels

Figura 3.3: Duas imagens de tamanho (481×321).

também evitam bordas aguçadas, que são indesejadas. Maiores valores de β melhoram o critério de parada na evolução das sementes. Na teoria, os termos de velocidade S_I e S_B são definidos em todos os pontos do primeiro nível, mas na prática, o algoritmo calcula a velocidade só nas bordas dos super pixels em cada iteração.

O algoritmo termina quando as bordas deixam de evoluir. Teoricamente, as bordas podem evoluir indefinidamente apresentando uma sobre-diminuição das velocidades. Mas, o algoritmo finaliza quando o incremento relativo do total da área coberta pelos super pixels é menor que um limiar. Quando o algoritmo para, a evolução resultante é pós-processada para que as bordas dos super pixels sejam exatamente de um pixel de largura. Este processo é realizado em três passos: 1) Toda região extensa restante não atribuída é tratada como um super pixel; 2) Os super pixels muito pequenos são removidos e os seus pixels são marcados como não atribuídos e 3) Esses pixels não atribuídos são removidos, aplicando o algoritmo de afinamento (Siddiqi et al., 2002). O pós-processamento é realizado visando suavizar as bordas dos super pixels, como mostra a Figura 3.3.b.

3.2 Simple Linear Iterative Clustering

O algoritmo proposto por Achanta et al. (2010) gera super pixels mediante o agrupamento de pixels baseado na sua similaridade de cor e sua proximidade na imagem. Isso é feito no espaço lab_{xy} , no qual lab é a cor do pixel no espaço CIELAB, o que é considerado perceptualmente uniforme para distâncias pequenas de cor, e xy é a posição dos pixels.

O algoritmo tem como entrada um número K constante de super pixels, todos com o aproximadamente o mesmo tamanho (Figura 3.4.a). Para uma imagem com N pixels, o tamanho aproximado de cada super pixel é N/K pixels. Para super pixels aproximadamente de igual tamanho, haveria um centróide de cada super pixel em cada intervalo da grade $S = \sqrt{N/K}$.

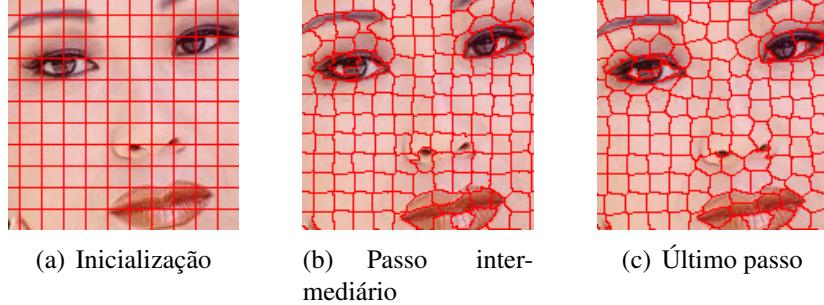


Figura 3.4: O algoritmo repete iterativamente o processo de associar pixels com o centróide mais próximo até a convergência.

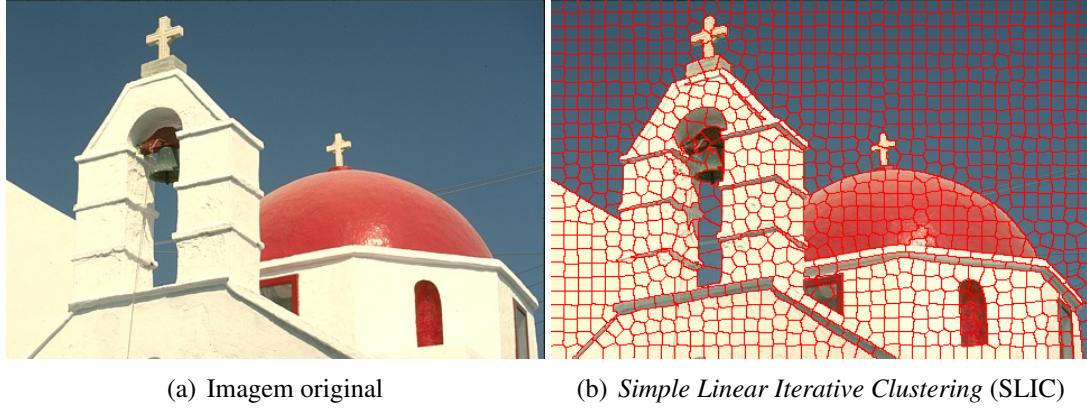
Inicialmente, são selecionados os K centróides dos super pixels $C_k = [l_k, a_k, b_k, x_k, y_k]$ com $k = [1, K]$ dos intervalos regulares da grade S . Dado que a extensão espacial de qualquer super pixel é de aproximadamente S^2 (a área aproximada de um super pixel), pode-se assumir que os pixels que estão associados com um centróide têm uma área $2S \times 2S$ em torno dele no plano xy . Esta é a área de busca para os pixels mais próximos a cada centróide. A distância Euclidiana no espaço de cor CIELAB é perceptivamente significativa para distâncias pequenas (m na equação 3.3). Se as distâncias espaciais dos pixels excedem o limite perceptual de cor, então eles começam a superar as semelhanças de cor (resultando em super pixels que não respeitam limites da região, apenas a proximidade no plano da imagem). Portanto, em vez de usar uma norma simples Euclidiana no espaço $5D$, é usada uma função D_S para medir a distância, definida a seguir:

$$\begin{aligned} d_{lab} &= \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2} \\ d_{xy} &= \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} \\ D_s &= d_{lab} + \frac{m}{S} d_{dx} \end{aligned} \quad (3.3)$$

onde D_s é a soma da distância lab e xy normalizada pelos intervalos S . O parâmetro m em D_s permite o controle da convexidade dos super pixels. Quanto maior o valor de m , será mais enfatizada a proximidade espacial. Este valor pode estar no intervalo de $[1 - 20]$.

O algoritmo inicia testando os K centróides regularmente distribuídos, movendo-os à correspondente posição mais baixa da gradiente em uma vizinhança de 3×3 . Isto é realizado para evitar colocá-los em uma borda e para reduzir as chances de escolher um pixel ruidoso. Os gradientes da imagem são computados como mostra a Equação a seguir:

$$G(x, y) = \|I(x+1, y) - I(x-1, y)\|^2 + \|I(x, y+1) - I(x, y-1)\|^2 \quad (3.4)$$

Figura 3.5: Duas imagens de tamanho (481×321) .

onde, $I(x, y)$ é o vetor *lab* correspondente ao pixel na posição (x, y) , e $\|\cdot\|$ é a norma L_2 . Isto leva em conta a cor e a intensidade. Cada pixel na imagem é associado ao centróide mais próximo, cuja área de busca alcance ao pixel. Depois que todos os pixels estejam associados ao centróide mais próximo, um novo centróide é calculado como a média do vetor lab_{xy} de todos os pixels pertencentes ao super pixel. O algoritmo repete iterativamente o processo de associar pixels com o centróide mais próximo até a convergência como mostra a Figura 3.5.b.

Ao finalizar o processo, alguns rótulos podem estar dispersos, ou seja, poucos pixels na vizinhança de um segmento têm o mesmo rótulo que ele, mas não estão conectados ao segmento. Embora seja raro, isso pode ocorrer, porque o algoritmo não reforça a conectividade explicitamente. No entanto, o método reforça a conectividade no último passo do algoritmo por meio da re-rotulação de segmentos disjuntos com o rótulo do maior segmento vizinho. Esse passo tem complexidade linear $O(N)$ e gasta menos do 10% do tempo total requerido pela segmentação da imagem.

3.3 Speeded-up Turbo Pixels

Cigla e Alatan (2010) propuseram uma técnica para gerar super pixels baseada no algoritmo de agrupamento *k – means*, capaz de gerar super pixels uniformes com baixo custo computacional chamada *Speeded-up Turbo Pixels* (STP). Inicialmente, a imagem é dividida em regiões quadradas de acordo com um tamanho desejado do lado do quadrado. Por isso, inicialmente, cada super pixel tem forma retangular com centróides equidistantes como mostra a Figura 3.6.a. Este passo é considerado como a inicialização da extração dos super pixels. No passo seguinte, os pixels sobre as bordas dos segmentos são atribuídos aos novos segmentos mediante a minimização da função de custo a seguir:

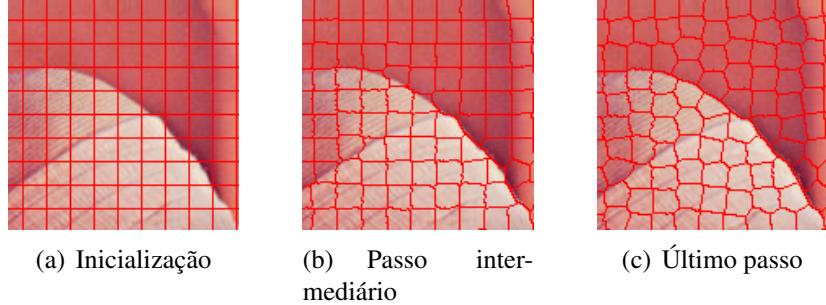


Figura 3.6: Os pixels nas bordas (cor vermelha) são atualizados em cada iteração do algoritmo STP.

$$C_{x,y}(i) = \lambda_1 \cdot |I(x, y) - I_i| + \lambda_2 \cdot |(x - C_x^i)^2 + (y - C_y^i)^2| \quad (3.5)$$

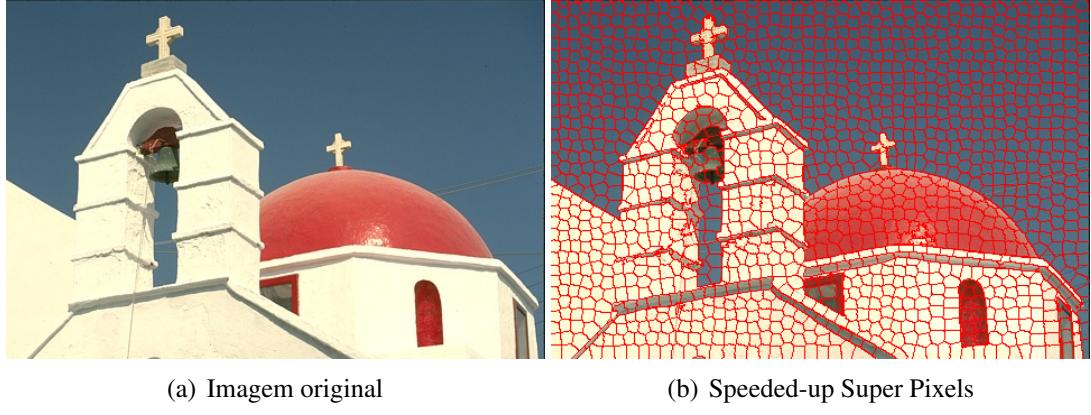
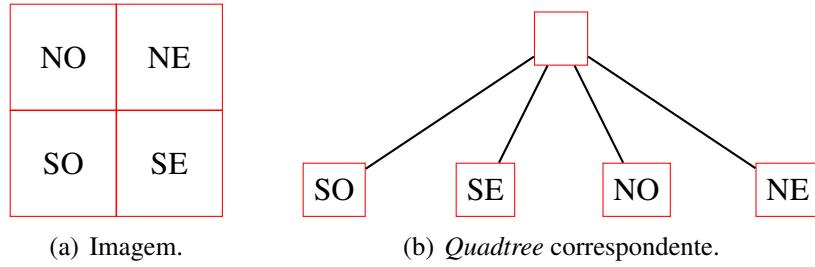
onde I_i indica a intensidade media do I -ésimo segmento, e (x, y) e a coordenada do pixel testado entre os diferentes segmentos. C_x^i e C_y^i são a coordenada do centróide do I -ésimo segmento. O parâmetro λ_1 é o peso da similaridade de intensidade e as limitações da conectividade. O primeiro termo da Equação 3.5 garante a similaridade da cor dos pixels que serão agrupados, enquanto que o segundo termo garante a convexidade dos super pixels evitando que pixels muito distantes sejam agrupados. A convexidade dos super pixels pode ser mudada alterando-se o parâmetro λ_2 .

Durante o processo da geração dos super pixels, apenas os pixels nas bordas dos segmentos são testados, garantindo a conectividade dos super pixels, o seja, que não tenham “buracos” (pixels que não pertencem a nenhum super pixel) na grade dos super pixels, como mostra a Figura 3.7.b. Tal abordagem, fornece um algoritmo rápido, já que um número limitado de pixels é testado entre os segmentos vizinhos (4 conetado). Uma vez que todas as bordas são testadas, as intensidades médias e os centroides são atualizados com os novos pixels agrupados ou removidos. O número de iterações do algoritmo é limitado mediante um parâmetro it .

3.3.1 Quadtree Speeded-up Turbo Pixels

O algoritmo STP original inicia a convergência dos super pixels a partir de uma pré-segmentação da imagem em segmentos de forma e tamanho regular como mostra a Figura 3.6.a. Mas é possível otimizar a pré-segmentação inicial, tentando aproveitar as regiões da imagem com propriedades similares, como por exemplo a cor. Neste projeto propomos uma pré-segmentação adaptativa, baseada na estrutura de dados *Quadtree* proposta por Finkel e Bentley (1974).

O *Quadtree* é uma estrutura de dados espacial usada para segmentar espaços de duas dimensões, como as imagens. O processo de segmentação é feito mediante a divisão re-

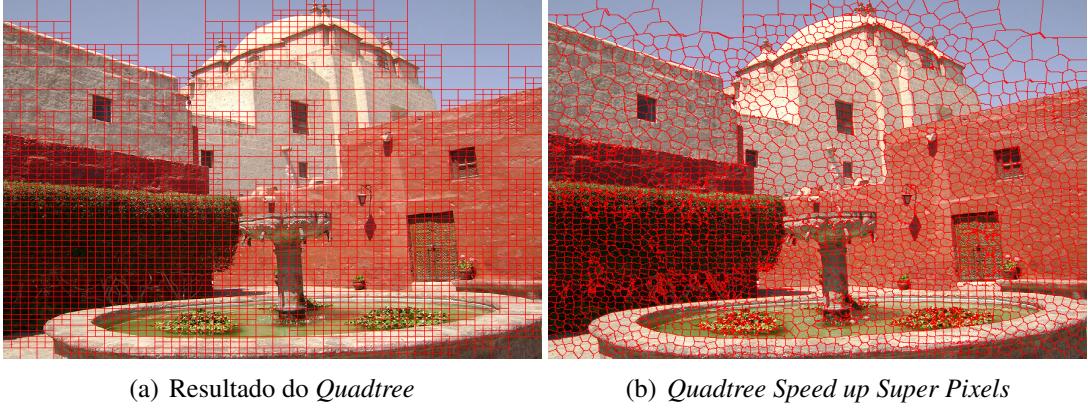
Figura 3.7: Duas imagens de tamanho (481×321) .Figura 3.8: Exemplo da árvore *Quadtree* após uma iteração.

cursiva do espaço em quatro quadrantes ou regiões. A imagem é armazenada em uma árvore cujos quatro nós representam os quadrantes: noroeste (NO), nordeste (NE), sudoeste (SO) e sudeste (SE), a Figura 3.8.a mostra uma imagem dividida em 4 regiões, após uma iteração, com seu *Quadtree* correspondente na Figura 3.8.b.

O resultado do *Quadtree*, como ilustra a Figura 3.9.a, é uma segmentação da imagem em segmentos retangulares com tamanhos adaptados às propriedades da imagem, onde as regiões de maior gradiente apresentam segmentos menores. O processo iterativo é controlado por dois parâmetros: l que é o menor tamanho permitido do lado dos segmentos, e t que é um limiar para controlar o nível de diferença das cores. A partir da pré-segmentação obtida com o *Quadtree* é possível continuar com o segundo passo do algoritmo STP, ou seja, aplicar a Equação 3.5 para gerar os super pixels. O resultado final é a imagem segmentada em super pixels de boa qualidade, mas com uma diminuição importante no número de segmentos, o que pode ser muito útil para os processos a serem aplicados sobre os super pixels gerados. O resultado do processo é mostrado na Figura 3.9.b.

3.4 Considerações Finais

A complexidade do algoritmo *Turbo Pixels* é aproximadamente linear no número total de pixels N da imagem. Em cada passo de tempo, todos os elementos da função Ψ são

Figura 3.9: Duas imagens de tamanho (800×600) .

atualizados (Equação 3.1). Cada atualização requer o cálculo das derivadas parciais de Ψ e a computação de S_IS_B . Assim, cada atualização implica em $O(N)$ operações. O cálculo do esqueleto homotópico é feito em $N \log N$. Por outro lado, a complexidade das técnicas SLIC e STP são um caso especial do algoritmo $k-means$ adaptado para a tarefa de gerar super pixels. A complexidade do algoritmo $k-means$ clássico é $O(NKI)$ onde N é o número de pontos (pixels na imagem), K é o número de *clusters* ou segmentos, e I é o número de iterações necessárias para convergir. O método SLIC possui uma complexidade de $O(N)$, porque precisa apenas calcular as distâncias de qualquer ponto com não mais de oito centróides e o número de iterações é constante, 10. O método *Speeded-up Turbo Pixels* precisa apenas fazer os cálculos de similaridade com os pixels nas bordas M dos super pixels e com não mais de quatro centróides. As iterações podem ser alteradas visando segmentações mais precisas e o número ideal de iterações é diretamente proporcional ao tamanho dos super pixels iniciais. Por exemplo, para super pixels de tamanho inicial de 10×10 pixels, o número de iterações, geralmente, é no máximo 10. Então, a complexidade do algoritmo *Speeded-up Turbo Pixels* é $O(MI)$ e a versão Quadtree é dada pela complexidade da árvore $O(p + q)$ para uma imagem de tamanho $2^p \times 2^q$ mais a complexidade do algoritmo *Speeded-up Turbo Pixels*. No entanto, na prática, o algoritmo STP, para as duas versões, gera super pixels em menor tempo de processamento que as outras duas técnicas, sendo que o algoritmo *Turbo Pixels* é o mais lento.

O três algoritmos apresentados precisam de parâmetros fornecidos pelo usuário. O algoritmo *Turbo Pixels* na Equação 3.2 necessita do valor α para evitar o “vazamento” das bordas e evitar formas aguçadas, e o valor β para controlar o critério de parada do algoritmo. O SLIC precisa do parâmetro m em 3.3 que permite o controle da convexidade dos super pixels. O algoritmo STP, na Equação 3.5, precisa de dois parâmetros, λ_1 para controlar a similaridade das intensidades dos super pixels e λ_2 , para controlar a convexidade deles, além do número de iterações it .

Obter ótimos resultados (super pixels bem ajustados aos objetos da imagem) é muito importante no processo de detecção de comunidades (segmentação da imagem). Pequenos erros na pré-segmentação (super pixels) podem gerar comunidades de baixa qualidade, isto é regiões da imagem misturadas com segmentos de outras. Para que os algoritmos consigam ajustar os super pixels diminuindo o erro é necessário ajustar os valores dos parâmetros. O problema é que os autores dos métodos não fornecem uma forma de escolher o melhor valor. Além disso uma pequena alteração leva a resultados diferentes sobre a mesma imagem, e valores ótimos usados em uma imagem nem sempre podem ser usados em outras, especialmente quando se muda a resolução e distribuição das cores na imagem.

Neste cenário, o método *Turbo Pixels* depende de muitas heurísticas para atingir seu objetivo, o que dificulta muito a sua implementação, além de ser o mais lento. Os métodos SLIC e STP usam a mesma função de convergência (*k-means*) para gerar super pixels, enquanto originalmente o SLIC usa o modelo de cor CIELAB, e o STP usa o nível de cinza. Como originalmente propostos, o algoritmo SLIC gera super pixels mais rápido do que o algoritmo STP, devido ao modelo de cor usado pelo SLIC. No entanto, o *Speeded-up Turbo Pixels* apresenta duas vantagens importantes. A primeira é que não depende de uma grade regular para iniciar o processo, o que permite desenvolver melhorias do algoritmo, como apresentado na Subseção 3.3.1. A segunda é que o intercâmbio de pixels é realizado apenas nas bordas dos segmentos. Assim, uma mudança no modelo de cor do algoritmo para o modelo CIELAB, gera os mesmos resultados que o SLIC, mas em menor tempo de processamento.



CAPÍTULO
4

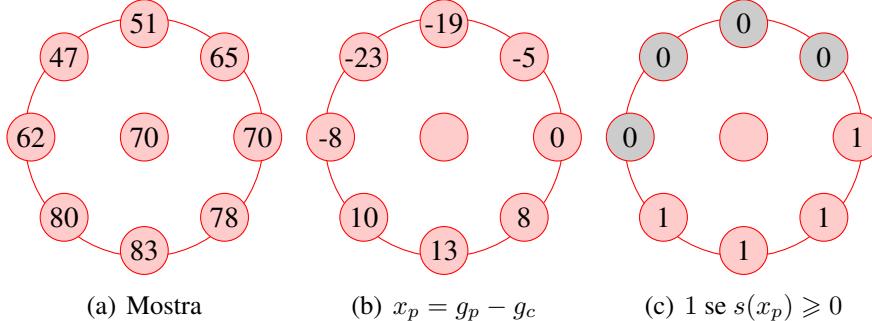
Fundamentos Complementares

Este capítulo apresenta dois fundamentos importantes empregados neste trabalho, que por razões conceituais, não caberiam nos capítulos de fundamentos de Redes Complexas (Capítulo 2) e Super Pixels (Capítulo 3) anteriores.

O primeiro, *Local Binary Pattern* (LBP), diz respeito a uma técnica de medida de textura a ser explorada no contexto de super pixel. O segundo fundamento, Métrica de similaridade para Segmentação de Imagens (MSSI), é uma contribuição original deste trabalho cujo objetivo é fornecer um mecanismo adequado para medir, quantitativamente, a qualidade da segmentação obtida.

4.1 *Local Binary Pattern*

O operador *Local Binary Pattern* é uma medida de textura local multirresolução invariante à rotação proposta por Ojala et al. (2002). A ideia básica é obter um código binário que descreve a textura local em uma vizinhança circular de raio R , mediante o valor do nível de cinza do centro e os P valores da vizinhança. Assim, o código LBP para um pixel é calculado aplicando a Equação definida a seguir:



$$(1 \times 1) + (1 \times 2) + (1 \times 4) + (1 \times 8) + (0 \times 16) + (0 \times 32) + (0 \times 64) = 15$$

(d) Multiplicar $s(x_p)$ por potências de dois e somar

Figura 4.1: Exemplo do cálculo do código LBP para $P = 8, R = 1$.

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p$$

$$s(x) = \begin{cases} 1, & \text{se } x \geq 0; \\ 0, & \text{caso contrário.} \end{cases} \quad (4.1)$$

onde g_c corresponde ao nível de cinza do pixel no centro da vizinhança local e $g_p(p = 0, \dots, P - 1)$ corresponde aos valores do nível de cinza dos pixels distribuídos no círculo de raio R , que forma uma vizinhança circularmente simétrica. O cálculo do código LBP para um pixel, pode ser melhor compreendido no exemplo da Figura 4.1.

Para remover o efeito da rotação, cada código LBP é rotacionado a uma posição de referência, tornando “padronizadas” todas as versões de um código binário. A transformação é definida na Equação a seguir:

$$LBP_{P,R}^{ri} = \min\{ROR(LBP_{P,R}, i) | i = 0, 1, \dots, P - 1\} \quad (4.2)$$

onde ri significa invariante à rotação. A função $ROR(x, i)$ desloca circularmente o número binário x , i vezes no sentido horário ($|i| < P$).

O processo de padronizar os códigos binários, mediante a rotação, deve ser feito apenas sobre os “códigos uniformes”. O conceito de códigos ou padrões uniformes foi introduzido por Mäenpää et al. (2000). Eles observaram que certos padrões possuíam as propriedades fundamentais da textura, as quais eram representadas na maioria desses padrões, as vezes superior a 90%. Esses padrões são chamados “uniformes” porque eles têm uma propriedade em comum: no máximo possuem dois cruzamentos, de zero para um ou de um para zero, no código binário circular, ou seja têm uniformidade $U \leq 2$. A

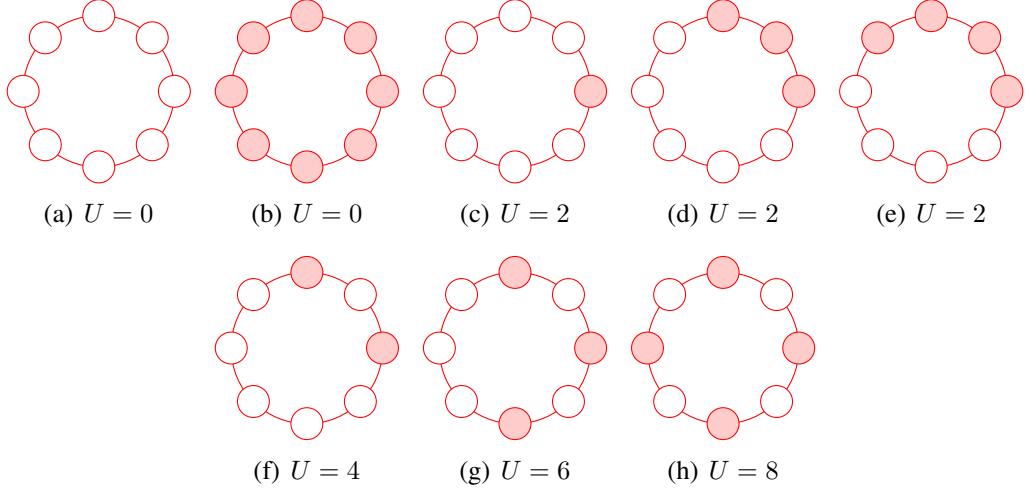


Figura 4.2: Os códigos binários da parte superior (a - e) ilustram padrões uniformes $U \leq 2$, e os códigos da parte de inferior (f - h) ilustram padrões não uniformes $U > 2$.

Figura 4.2 mostra cinco exemplos de padrões uniformes para $P = 8$, e três exemplos de padrões não uniformes para o mesmo valor de P

Assim, uma imagem é escaneada pixel a pixel calculando seu padrão LBP, o qual é acumulado em um histograma H de tamanho $P + 2$, o critério de acumulação de padrões LBP é detalhado a seguir:

$$index = \begin{cases} \log_2(LBP_{P,R}^{ri} + 1), & \text{se o LBP é uniforme;} \\ P + 1, & \text{caso contrário.} \end{cases} \quad (4.3)$$

$$H[index] = H[index] + 1$$

O histograma obtido de cada imagem é o vetor de características de textura da imagem e pode ser comparado com o vetor de características de uma outra imagem usando a distância Euclidiana, Manhattan, Chi-square, entre outras.

4.2 Métrica de similaridade para Segmentação de Imagens

Arbeláez Arbeláez et al. (2009) propuz uma métrica para medir a cobertura ou similaridade entre duas segmentações, a qual é baseada no somatório das interseções das regiões que compõem cada segmentação. O uso dessa métrica é sugerido pelo grupo de Visão Computacional da Universidade de Berkeley da Califórnia¹, que disponibiliza um conjunto de imagens livre, para pesquisa e educação. A métrica, mesmo que consiga

¹<http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/>

quantificar a similaridade entre duas segmentações, apresenta algumas limitações importantes para nosso trabalho. Em seguida serão apresentados os dois principais problemas da métrica do Arbeláez et al. descrita na Fórmula a seguir.

$$\begin{aligned} O(R, R') &= \frac{R \cap R'}{R \cup R'} \\ C(S, S') &= \frac{1}{N} \sum_{R \in S} R \cdot \max(O(R, R')) \end{aligned} \quad (4.4)$$

onde $C(S, S')$ é a cobertura (resultado) entre as segmentações S e S' . A função \max é o maior valor da cobertura (sobreposição) $O(R, R')$ entre uma região R e R' .

O primeiro problema da métrica é que esta não satisfaz a propriedade da “simetria”, ou seja, $C(S, S') \neq C(S', S)$, o que representa um grande inconveniente, principalmente para nosso método de avaliação dos resultados, o qual será explicado no Capítulo 5. Além disso, a métrica apresenta outro problema no momento de comparar duas segmentações iguais, $d(S, S') = 0.5$ onde $S = S'$. Nesse caso, o resultado da técnica é 0.5 quando, normalmente, o esperado é 1 ou 0.

As limitações encontradas na métrica sugerida pelo grupo de Berkeley, inspiraram o desenvolvimento de uma nova métrica para comparar a similaridade entre duas segmentações que não apresente os mesmos problemas, além de apresentar maior precisão nos resultados. A nova métrica proposta neste trabalho, denominada Métrica de Similaridade para Segmentação de Imagens (MSSI), é detalhada a seguir.

A ideia básica da métrica é medir a similaridade entre duas segmentações, calculando a maior interseção das regiões que as compõem. Dadas duas segmentações S e S' compostas por R e R' regiões. Criar uma matriz M de tamanho $R \times R'$ onde:

$$M_{ij} = R_i \cap R'_j \quad (4.5)$$

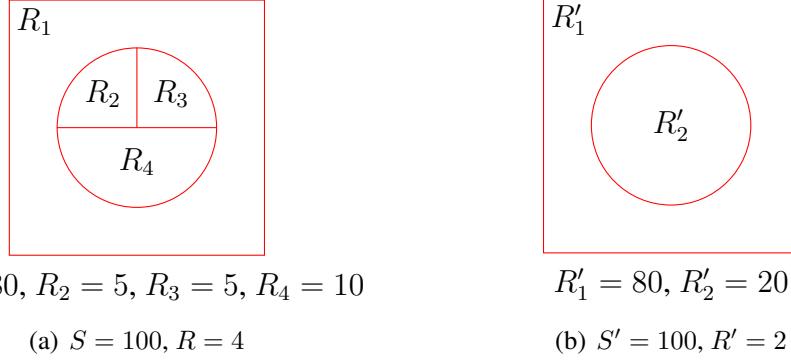
Assim, a interseção (similaridade) I entre duas segmentações S e S' é definida na Fórmula 4.6 a seguir:

$$I(S, S') = \frac{1}{N} \sum_{k=1}^{\min(R, R')} \max_k(M_{rc}) \quad (4.6)$$

onde N é o número de pixels na imagem e $\max_k(M_{rc})$ são os maiores elementos tanto por linha quanto por coluna da matriz M .

A métrica definida satisfaz as seguintes propriedades importantes:

1. $0 \leq I(S, S') \leq 1$

Figura 4.3: Duas segmentações S e S' divididas em R e R' regiões respectivamente.

$$2. I(S, S') = 1, \text{ se } S = S'$$

$$3. I(S, S') = I(S', S)$$

$$4. I(S, S') \geq I(S, S'') \wedge I(S, S'') \geq I(S', S'') \implies I(S, S') \geq I(S', S'')$$

A nova métrica pode ser melhor compreendida no exemplo a seguir. Dadas duas segmentações diferentes S e S' , ilustradas nas Figuras 4.3.a e 4.3.b, o valor da similaridade entre elas é obtido aplicando os seguintes passos:

a) Criar a matriz M com as intersecções entre todas as regiões de cada segmentação (Equação 4.5):

$$M = \begin{matrix} & \begin{matrix} R_1 & R_2 & R_3 & R_4 \end{matrix} \\ \begin{matrix} R'_1 \\ R'_2 \end{matrix} & \begin{bmatrix} 80 & 0 & 0 & 0 \\ 0 & 5 & 5 & 10 \end{bmatrix} \end{matrix} \quad (4.7)$$

b) Dado que o resultado da função $\min(R, R') = 2$. Encontrar os 2 maiores valores da matriz tanto por linha quanto por coluna:

$$\max(M_{rc}) = [80, 10] \quad (4.8)$$

C) Somar os maiores valores obtidos e dividir por N :

$$\sum_{k=1}^2 \max_k(M_{rc}) = 90$$

$$N = 100$$

$$I(S, S') = \frac{90}{100} = 0.9 \quad (4.9)$$

O resultado da métrica é 0.9 o que indica que as duas segmentações comparadas são

muito parecidas ou têm 90% de similaridade. Note-se que se utilizarmos a métrica proposta por Arbelaez, o resultado seria 0.43 o que indica que a similaridade entre as duas segmentações testadas seria de apenas 43%, o que não condiz com a realidade, já que esse resultado não representa a real similaridade entre as segmentações do exemplo.

CAPÍTULO

5

Metodologia

A metodologia adotada neste trabalho pode ser mais bem compreendida pelo diagrama na Figura 5.1. A partir de uma imagem, define-se uma grade com células que, após convergência, resultarão em super pixels, cujas bordas irregulares deverão estar bem ajustadas aos objetos na imagem. A partir dos super pixels, uma rede complexa é criada. O passo final consiste em aplicar os algoritmos de detecção de comunidades, que produzirão a imagem segmentada. Esses passos são detalhados a seguir.

5.1 Cálculo dos Super Pixels

Para obter uma representação eficiente e compacta para nossa estratégia de segmentação de imagens baseada em redes complexas, é realizada uma redução de nós no grafo. Duas versões de super pixels foram implementadas e testadas neste trabalho: a) a técnica *Speeded-up Turbo Pixels* (Subseção 3.3), como originalmente proposta; b) uma versão modificada da mesma técnica, com a adição da estrutura de dados *Quadtree* (Subseção 3.3.1). Em ambos casos é definida inicialmente uma grade de segmentos (grupos de pixels). Em seguida é aplicado o método de intercambio de pixels iterativamente, até a convergência. O resultado é a pré-segmentação da imagem em grupos de pixels similares. Desta forma, para as tarefas subsequentes, em vez de considerar cada nó como um pixel, é tomado um grupo destes chamado Super Pixel.

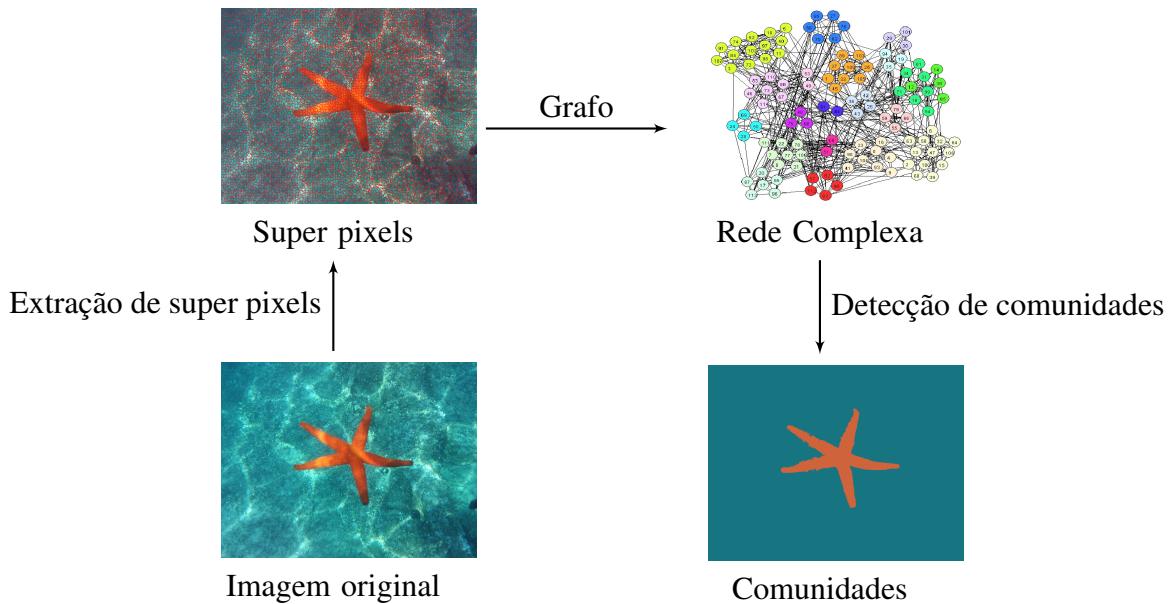


Figura 5.1: Abordagem de redes complexas para segmentação de imagens combinado com super pixels.

5.2 Geração do Grafo

A partir dos super pixels gerados, um grafo é criado. Usualmente, o que define a existência de uma aresta ligando dois vértices (nós) quaisquer em um grafo é a definição de um peso computado a partir de propriedades extraídas dos nós. É importante ressaltar que neste projeto os nós não são pixels individuais, são super pixels. Assim, para gerar o grafo, foram consideradas diversas propriedades dos super pixels, como a intensidade média, a media dos canais para o modelo de cor CIELAB e a textura. Em geral, cada super pixel é um nó no grafo, assim é aplicada uma função que permite calcular a diferença (peso) entre dois super pixels (nós). As conexões entre esses nós são estabelecidas somente se o peso for menor que um limiar t . O valor de t pode mudar de acordo com a similaridade dos pixels.

Testes preliminares mostraram que o valor de limiar t depende muito das características da imagem. Um limiar muito alto gera comunidades com regiões da imagem misturadas e um valor muito baixo pode sobre-segmentar a imagem. Destes problemas, o mais grave é gerar comunidades com regiões misturadas, o que acontece com valores do limiar muito altos. Alias, notamos que a sobre-segmentação é um “defeito” que acontece porque o valor baixo do limiar gera um grafo com muitos nós isolados (nós sem conexões), os quais, após aplicar os algoritmos de detecção de comunidades são considerados como uma comunidade muito pequena.

Diante disso, propomos montar o grafo usando um limiar adaptativo, ou seja, um limiar inicialmente baixo que incrementa seu valor quando o super pixel testado não tem nenhuma conexão. O valor do limiar é incrementado e o super pixel é testado novamente, este processo é realizado iterativamente até que o nó possua pelo menos uma conexão.

Além disso, as conexões são definidas dentro de uma região circular de raio R , que pode mudar de acordo com o tamanho da imagem, tamanho dos super pixels, a cor das regiões e a sua proximidade. O raio é usado para evitar conexões entre super pixels muito distantes. A Figura 5.2 mostra uma região amarela que compreende todos os super pixels para $R = 5$ a partir de um dado super pixel. A seguir são detalhadas as funções e métodos usados para a computação das arestas do grafo.

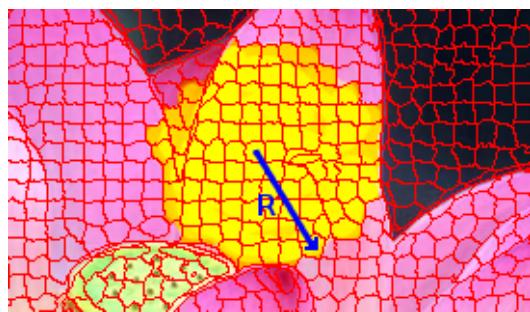


Figura 5.2: Região (área amarela) circular de raio $R = 5$.

5.2.1 Intensidade Média

A intensidade média dos super pixels é usada para calcular o peso W entre dois super pixels i e j . Se o peso for menor que o limiar t , então é criada uma aresta entre esses super pixels no grafo. A função de peso é definida a seguir:

$$W_{i,j} = |I_i - I_j| \leq t \quad (5.1)$$

onde I_i é a intensidade média do super pixel i .

5.2.2 Modelo de Cor

A criação de arestas entre dois super pixels i e j com base na cor é muito similar ao método baseado na intensidade média descrito anteriormente. É calculado o peso W entre dois super pixels e a conexão (aresta) entre eles é criada apenas para os pesos menores que o limiar t . A função de peso é definida a seguir:

$$W_{i,j} = \sqrt{\sum_k^n (C_{ki} - C_{kj})^2} \leq t \quad (5.2)$$

A função de peso W é a Distância Euclidiana, na qual C_{ki} é o canal k do modelo de cor do super pixel i .

A diferença da abordagem descrita para criar o grafo usando a intensidade média de cada super pixel. Neste caso o dado extraído a partir de um super pixel, não é apenas um valor entre 0 e 1. Assim, nesta abordagem, a propriedade extraída dos super pixels é um vetor de características. O tamanho do vetor é o número de canais do modelo de cor selecionado. Neste trabalho foram testados os modelos CIELAB e RGB que possuem 3 canais para representar a cor.

5.2.3 Textura

Dado que o super pixel é um grupo de pixels, é possível calcular diversas propriedades, como por exemplo a sua textura. No entanto, visto que o tamanho dos super pixels é, geralmente, pequeno (cerca de 100 pixels) é difícil aplicar os métodos tradicionais, como por exemplo os momentos estatísticos ou matrizes de co-ocorrência. Estes métodos, normalmente, requerem um número elevado de pixels para gerar bons descritores. Neste trabalho foi implementada a técnica *Local Binary Pattern* LBP (Seção 4.1) que produz bons descritores de textura para amostras pequenas.

Em nosso projeto o histograma LBP é obtido a partir de cada super pixel, ou seja, há um histograma LBP por cada super pixel. Assim, para a criação do grafo o histograma é considerado o vetor de características de cada super pixel e pode ser usada a Equação 5.2, descrita na abordagem baseada em cor.

5.3 Segmentação por meio de Redes Complexas

Uma vez construído o grafo, é possível aplicar os algoritmos de detecção de comunidades *Fast Greedy* e *Label Propagation* descritos nas Subseções 2.2.2 e 2.2.3, respectivamente. Esses algoritmos estão implementados na biblioteca Igraph¹ e foram incluídos em nossa implementação. O resultado desses algoritmos é a divisão dos nós dos grafo em diferentes comunidades. Cada comunidade encontrada representa um segmento na imagem.

Assim, o resultado do nosso processo de segmentação de imagens, por meio de redes complexas, é o resultado dos algoritmos de detecção de comunidades sobre os grafos montados a partir dos super pixels gerados das imagens a serem segmentadas.

¹<http://igraph.sourceforge.net/>

5.4 Avaliação dos Resultados

Diversas avaliações podem ser aplicadas sobre os resultados do nosso processo. Neste trabalho são avaliadas a qualidade da segmentação, o tempo de execução e a escalabilidade da técnica. A seguir são detalhados os critérios para as diferentes avaliações realizadas.

5.4.1 Avaliação da Qualidade

Após representar as comunidades encontradas nas imagens mediante os algoritmos de detecção de comunidades, é realizado o processo de avaliação dos resultados obtidos. Sendo a segmentação de imagens uma tarefa subjetiva e, dado que o ponto de vista dos usuários é muito variado, avaliar os resultados é também um processo complicado e subjetivo. Uma imagem pode possuir diferentes segmentações “corretas” ou de boa qualidade segundo o ponto de vista dos usuários. Ou uma mesma segmentação pode ser de boa ou má qualidade para as diferentes apreciações dos usuários.

Em função destas ponderações, foram avaliadas todas (300) as imagens disponibilizadas pelo grupo de Visão Computacional da Universidade de Berkeley da Califórnia para diferentes valores dos parâmetros usados em nossa abordagem, tanto para a pré-segmentação das imagens mediante super pixels, quanto para o processo da criação do grafo. Uma vez segmentadas todas as imagens, os resultados foram comparados com as segmentações manuais fornecidas pelo banco de imagens Berkeley. Por conta da imensa quantidade de valores obtidos pela nova métrica, visando facilitar a análise dos dados, foi criado um banco de dados para armazenar os resultados quantitativos para cada uma das 300 imagens segmentadas.

O critério usado para comparar nossos resultados com as segmentações manuais é baseado na métrica (MSSI) proposta neste trabalho (Secção 4.2), a qual compara as segmentações obtidas pelo método proposto com as segmentações manuais, calculando a intersecção de todas as regiões das segmentações manuais e as regiões dos nossos resultados. Por outro lado, o banco de imagens Berkeley disponibiliza várias e diferentes quantidades de segmentações manuais para uma mesma imagem, o que torna inviável a avaliação dos resultados, por causa da subjetividade de cada segmentação e pelo grande número de avaliações que teriam que ser realizadas. Porém, foi realizado um processo de seleção de apenas uma segmentação manual para cada imagem a ser comparada com nossos resultados. O critério da seleção das segmentações manuais é detalhado a seguir.

Uma imagem pode possuir diversas e diferentes segmentações manuais, como mostra o exemplo da Figura 5.3, onde uma imagem é acompanhada de duas segmentações feitas por diferentes usuários. Pode se ver a grande diferença entre as duas segmentações.

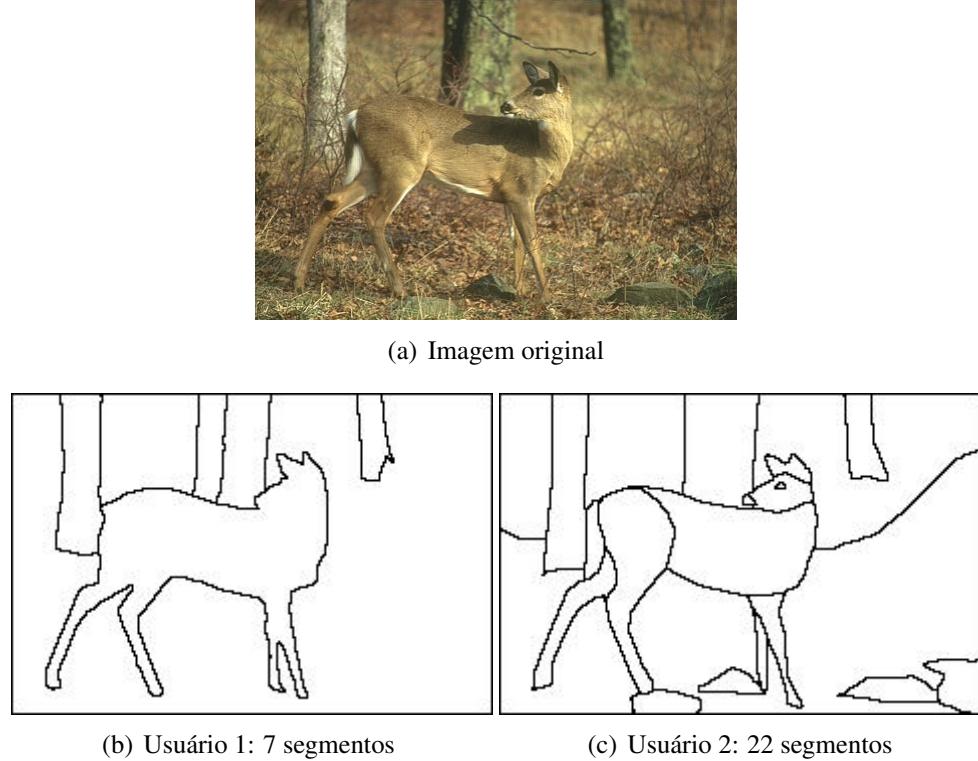


Figura 5.3: As segmentações (a) e (b) mostram a subjetividade do processo.

Para resolver o problema da subjetividade mostrado na Figura 5.3, nós realizamos a seleção de apenas uma segmentação para cada imagem. A imagem segmentada selecionada é aquela que, na média, se assemelha às demais segmentações manuais e neste trabalho será denominada imagem segmentada de “referência”.

Para realizar o processo de seleção da segmentação de referência, é empregada a métrica MSSI descrita na Secção 4.2. Para isso, são calculadas as interseções entre todas as segmentações, as quais são armazenadas em uma matriz M . Em seguida, são somadas todas as interseções de cada linha, e é escolhida a segmentação com a maior soma. O processo pode ser descrito na equação a seguir:

$$S = \max(\forall_{i=[1..n]}, \sum_{j=1}^n M_{ij}) \quad (5.3)$$

onde M é a matriz das interseções entre todas as segmentações manuais e S é a segmentação selecionada para ser testada com nossos resultados. Este processo foi aplicado sobre todas as segmentações manuais das 300 imagens do conjunto Berkeley. A Figura 5.4 mostra o resultado do processo de seleção da imagem de referência, além das outras segmentações manuais usadas nos processos.

Além da medição da qualidade dos resultados descrita anteriormente, os resultados foram comparados com as segmentações obtidas da técnica de segmentação de imagens

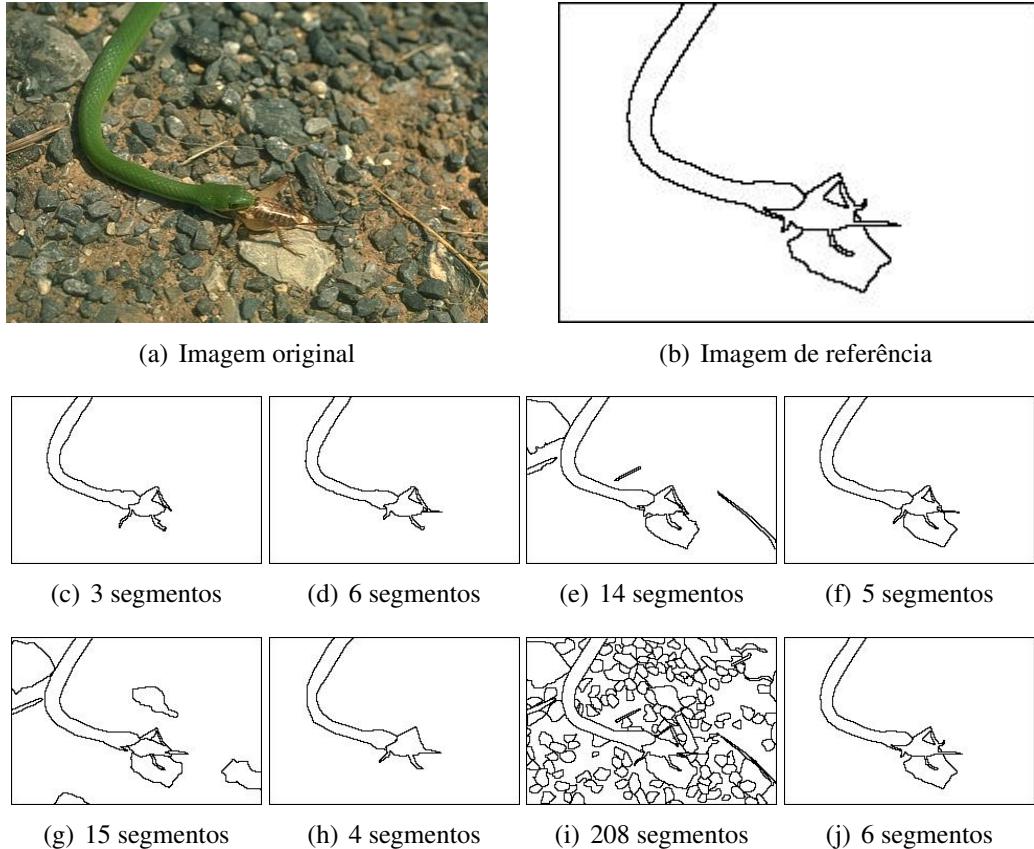


Figura 5.4: Seleção da segmentação de referência, ilustrada na imagem (j) e ampliada na imagem (b)

GBIS descrita na Seção 1.1. Foi calculada a qualidade e comparada com nossos resultados das segmentações geradas para as 300 imagens do Berkeley.

5.4.2 Avaliação do Tempo de Processamento

Todos os experimentos realizados neste projeto foram executados em um computador Intel Core i7-2600 (8M Cache, 3.40 GHz) e Memória de 4Gb com sistema operacional Ubuntu Linux 12.10. Para a medição do tempo foi empregada a função `gettimeofday`² da biblioteca `sys/time.h`. Os algoritmos para a geração tanto dos Super pixels, quanto do grafo, foram implementados na linguagem C++. Para os algoritmos de detecção de comunidades, foi empregado o pacote Igraph³ implementado na linguagem C.

²<http://linux.die.net/man/2/gettimeofday>

³<http://igraph.sourceforge.net/>

CAPÍTULO

6

Resultados

Este capítulo descreve 6 experimentos, cada qual ilustrando aspectos distintos da abordagem de segmentação de imagens por meio de redes complexas proposta neste trabalho. Para realizar os experimentos apresentados neste capítulo, foi implementada uma versão da técnica para gerar super pixels *Speeded-up Turbo Pixels* descrita na Seção 3.3, além da implementação de nossa proposta do método adaptativo baseado no *Quadtree* apresentado na Subseção 3.3.1. Esta técnica foi escolhida porque gera os mesmos resultados que o método SLIC em menor ou igual tempo de processamento, além de permitir iniciar a convergência dos super pixels usando grades com segmentos de tamanhos diferentes, sem a necessidade de fazer alterações na função de convergência nem nos parâmetros. Por outro lado, o algoritmo selecionado para a detecção de comunidades é o *Fast Greed* explicado na Seção 2.2.2. O algoritmo foi selecionado, porque seu tempo de processamento é muito parecido com o tempo usado pelo algoritmo *Label Propagation*, mas principalmente, porque este último apresenta sobre-segmentação das redes analisadas, ou seja, gera muitas comunidades para regiões similares nas imagens testadas.

Os experimentos apresentados nas seções a seguir visam avaliar os aspectos mais importantes da abordagem de segmentação de imagens proposta neste trabalho. Estes aspectos são: o compromisso das resoluções dos super pixels com o tempo e qualidade da segmentação, a segmentação de imagens mediante a extração de textura dos super pixels, avaliação do tempo e qualidade das segmentações pré-segmentadas por meio da técnica *Quadtree Speeded-up Turbo Pixels*, estudo do comportamento dos parâmetros envolvidos no processo e comparação da nossa abordagem com outro método de segmentação de

imagens baseado em grafos.

6.1 Experimento 1: Imagem Sintética

O primeiro experimento, ao invés de focar na precisão da segmentação da imagem, descreve o compromisso entre distintas resoluções de super pixels e o tempo computacional para os três passos do processo da segmentação de imagens, os quais são: convergência dos super pixels, geração do grafo e detecção de comunidades. Para isso, foi usada uma imagem sintética¹ de tamanho 700×700 com três objetos (elipse, estrela, retângulo) de diferente nível de cinza e fundo branco. A Figura 6.1 mostra a imagem original e a grade com a convergência dos super pixels para os tamanhos: 10, 20, 50, 100 e 150 pixels.

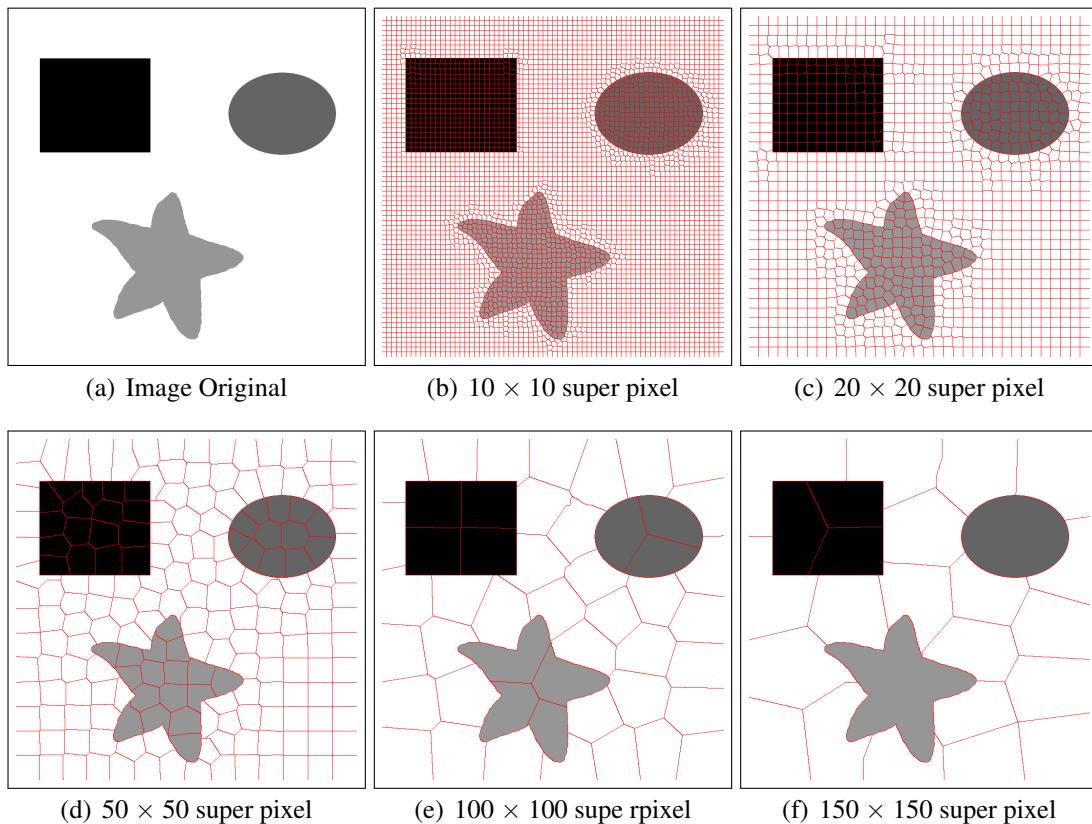


Figura 6.1: Diferentes resoluções de super pixels. (a) Imagem original; (b) Convergência 10×10 super pixels; (c) Convergência 20×20 super pixels; (d) Convergência 50×50 super pixels; (e) Convergência 100×100 super pixels e (f) Convergência 150×150 super pixels.

A imagem segmentada da Figura 6.1.a, para todas as resoluções de super pixels, é mostrada na Figura 6.2. O esperado 100% de precisão na segmentação deve-se a uma correta convergência da grade com as bordas dos objetos da imagem original (Figura

¹As imagens criadas manualmente, ou manipuladas, são chamadas neste trabalho *Sintéticas*

6.1.b - 6.1.f) e uma correta detecção das comunidades (fundo e três objetos) para todos os super pixels. Como descrito pela Equação 5.1, o peso das arestas para este experimento é calculado com o nível de cinza de cada super pixel.

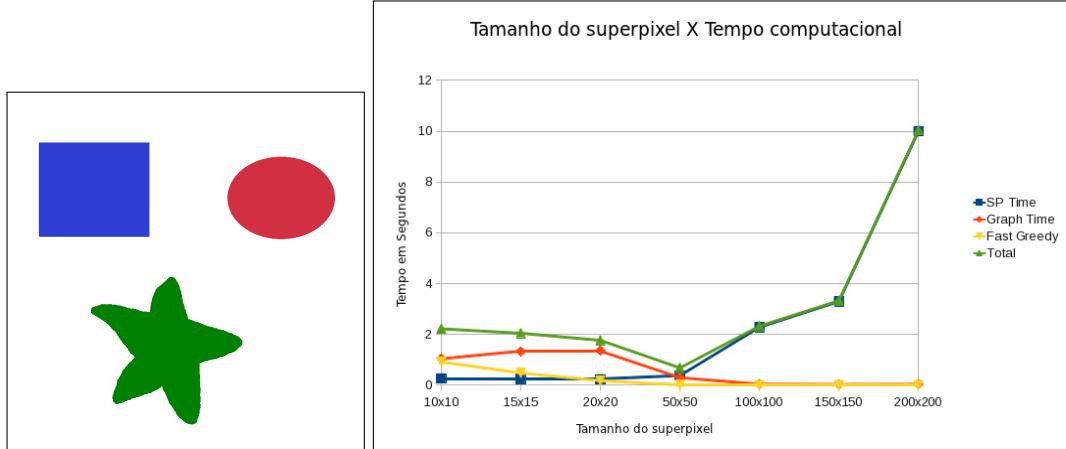


Figura 6.2: Imagem Sintética: a) Imagen segmentada para todos os super pixels e b) Gráfico do tempo de processamento versus o tamanho dos super pixels.

As Tabelas 6.1 e 6.2, e o gráfico da Figura 6.2.b fornecem uma ideia da influência da resolução dos super pixels no tempo computacional para todos os estados da segmentação de imagens proposta neste trabalho. Em geral, se o tamanho dos super pixel aumenta, também aumenta o tempo de processamento total. Embora poucos super pixels implique em menor tempo de processamento para o algoritmo de detecção de comunidades *Fast Greedy* (e a geração do grafo, se o raio permanece inalterado), um tempo extra é utilizado para a convergência dos super pixels com as bordas dos objetos da imagem atual. No entanto, o compromisso entre super pixels de tamanho 50×50 e o tempo de segmentação pode ser observado neste exemplo. Note que o menor tempo de processamento é obtido com este tamanho, superando a super pixels de tamanhos menores (10, 15 e 20).

Tamanho	Super Pixel					Grafo	
	Quantidade	Iterações	λ_1	λ_2	Limiar	Raio	
10x10	4900	7	1.00	0.005	0.89	6	
15x15	2209	7	1.00	0.001	0.7	7	
20x20	1225	7	1.00	0.001	0.7	7	
50x50	196	12	1.00	0.001	0.7	4	
100x100	49	70	3.00	0.00009	0.7	2	
150x150	25	100	5.50	0.00009	0.7	2	
200x200	16	150	9.00	0.00009	0.72	2	

Tabela 6.1: Parâmetros usados na segmentação da imagem sintética, para distintos tamanhos de super pixels.

Tamanho	Super pixels	Grafo	Fast Greed	Total
10×10	0.2631	1.0427	0,9139	2,22
15×15	0.2407	1.3229	0,4815	2,04
20×20	0.2288	1.3606	0,1774	1,77
50×50	0.3785	0.2962	0,0069	0,69
100×100	2.2807	0.0419	0,0001	2,32
150×150	3.2996	0.0283	$6,51 \times 10^{-5}$	3,33
200×200	10	0.0368	$6,91 \times 10^{-5}$	10,04

Tabela 6.2: Tempo computacional em segundos usado na segmentação da imagem sintética, para distintos tamanhos de super pixels.

6.2 Experimento 2: Flor de Lotus

Este experimento mostra como mudanças nos parâmetros podem afetar a segmentação de imagens reais. Estas mudanças no tamanho dos super pixels podem levar a uma incorreta segmentação, e também, como alterações nos parâmetros da geração do grafo produzem resultados corretos, revelando diferentes objetos na cena, para cada configuração. Como foi mencionado na Equação ??, o peso das arestas para este experimento é calculado com a media da cor (RGB) de cada super pixel e valor do limiar estático.

A Figura 6.3 mostra um exemplo onde as mudanças no tamanho dos super pixels podem levar a uma segmentação incorreta. Isto é de alguma forma o esperado para imagens reais com objetos mais complexos, com diferentes formas e áreas mais pequenas que o tamanho dos super pixels. As Figuras 6.3.a-f descrevem, respectivamente, a imagem original de 639×430 pixels, a grade dos super pixels de tamanho 10, 20 e 50, a imagem segmentada para os super pixels de 10×10 e 20×20 e a imagem segmentada para super pixels de 50×50 .

Embora dois componentes tenham sido identificados para os três casos (Flor e o fundo), uma segmentação correta é apenas obtida para super pixels de tamanhos 10×10 e 20×20 (6.3.e), enquanto que para super pixels de 50×50 partes da flor são agrupados como fundo e vice-versa (6.3.d). A Tabela 6.3 mostra os parâmetros empregados neste experimento e o tempo de processamento para todas as configurações é apresentado na Tabela 6.4. Note-se o pequeno tempo de processamento para todos os casos.

A Figura 6.4 ilustra a segmentação da Flor de Lótus com os parâmetros descritos na linha 4 da Tabela 6.3. Note que o tamanho dos super pixels manteve-se igual a 10, mas os parâmetros do grafo, limiar e raio são 0.96 e 2, respectivamente. Os experimentos foram executados com o algoritmo *Fast Greedy*, o qual retornou 52 comunidades diferentes. O número elevado de comunidades retornadas pelo algoritmos é por causa do uso do limiar estático na geração do grafo. Para maior clareza, as comunidades são exibidas separadamente, mas apenas aquelas com áreas que representam o 99.1589% do total da

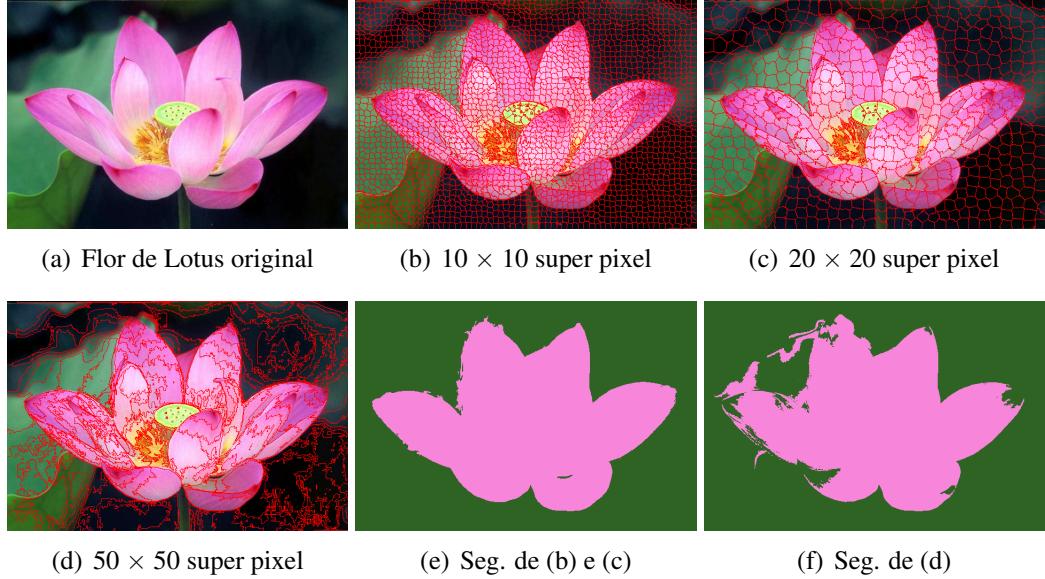


Figura 6.3: Segmentação da Flor de Lotus. (a) Imagem original de tamanho 639×430 ; (b) Super pixels de 10×10 ; (c) Super pixels de 20×20 ; (d) Super pixels de 50×50 ; (e) Segmentação para os super pixels de 10×10 e 20×20 e (f) Segmentação para os super pixels de 50×50 . O componente do fundo verde, e a flor em rosa.

Tamanho	Super Pixel				Grafo	
	Quantidade	Iterações	λ_1	λ_2	Limiar	Raio
10×10	10(2752)	6	1	0.0005	0.86	6
20×20	20(704)	15	2	0.0005	0.86	6
50×50	50(117)	30	1	0.000001	0.82	3
10×10	10(2752)	6	1	0.0005	0.91	2

Tabela 6.3: Parâmetros usados na segmentação da Flor de Lotus.

Tamanho	Super pixels	Grafo	Fast Greed	Total
10×10	0.16	0.94	0.42	1.52
20×20	0.34	0.61	0.02	0.97
50×50	0.74	0.14	0.01	0.89
10×10	0.27	0.11	0.039	0.42

Tabela 6.4: Tempo de processamento para a Flor de Lotus.

área na imagem. A Figura 6.4.a, Figura 6.4.b e a Figura 6.4.c descrevem, respectivamente, as pétalas, o fundo (contabilizadas as duas maiores comunidades encontradas) e os órgãos sexuais (os pistilos em amarelo e o cálice no meio). Este exemplo mostra que o método é capaz de incorporar, mediante a alteração dos parâmetros, a subjetividade implícita no processo de segmentação de imagens.

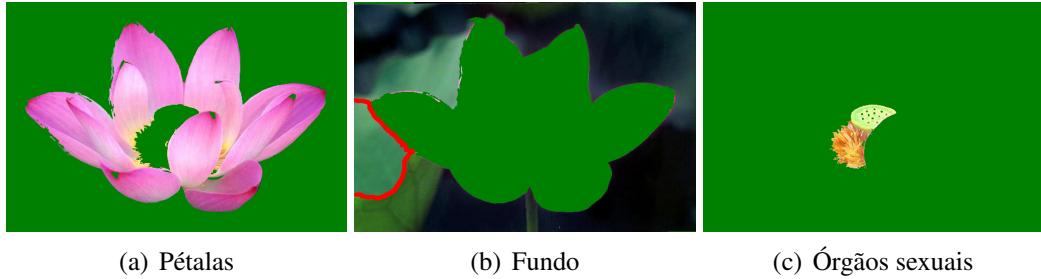


Figura 6.4: Segmentação de Flor de Lotus para super pixels de tamanho 10×10 , limiar = 0.91 e raio = 2. (a) Pétalas (b) Fundo, dois componentes e (c) Quarta comunidade, correspondente aos órgãos性uais da Flor.

6.3 Experimento 3: Textura

Os experimentos apresentados nesta seção mostram os resultados obtidos após aplicar o método em imagens com textura. O descritor de textura é o LBP apresentado na Seção 4.1. Como descrito na metodologia, os pesos das arestas são calculados aplicando a função de similaridade 5.2 sobre os histogramas LBP extraídos de cada super pixel.

6.3.1 Imagens Sintéticas

Neste experimento são testadas duas imagens ilustradas na Figura 6.5.a e 6.5.b: a primeira imagem mistura dois tipos de textura e a segunda mistura três tipos diferentes de textura. Este experimento mostra como o método LBP é capaz de extrair a textura de pequenas regiões da imagem (super pixels). As imagens possuem uma resolução de 640×640 e 1024×1024 , respectivamente. O tamanho usado para gerar os super pixels é de 15×15 e λ_1 é 1 e λ_2 é 0.09. Para montar o grafo o valor do raio foi 6 e o valor do limiar 0.1. Os experimentos foram executados com o algoritmo *Fast Greedy*.

Os experimentos mostraram que os super pixels extraídos das imagens com textura, mesmo que possibilitem pré-segmentar a imagem, não apresentam a qualidade esperada. Os métodos de geração de super pixels são baseados na cor das imagens, assim torna-se difícil aplicar estes métodos sobre texturas, já que são domínios diferentes. Os resultados dos super pixels são mostrados nas Figuras 6.5.b e 6.5.e. Para maior clareza foi aplicado zoom sobre uma região representativa destas.

O resultado do algoritmo de detecção de comunidades, ilustrado nas Figuras 6.5.c e 6.5.f, retornou 2 comunidades para a imagem da Figura 6.5.a e 12 comunidades para a imagem da Figura 6.5.d. Os testes mostraram que o método LBP consegue extrair corretamente a textura de pequenas regiões de 15×15 aproximadamente, neste caso. Além disso, mostraram que o método de segmentação de imagens, proposto neste trabalho, é capaz de segmentar imagens com vetores de características não apenas baseados em cor.

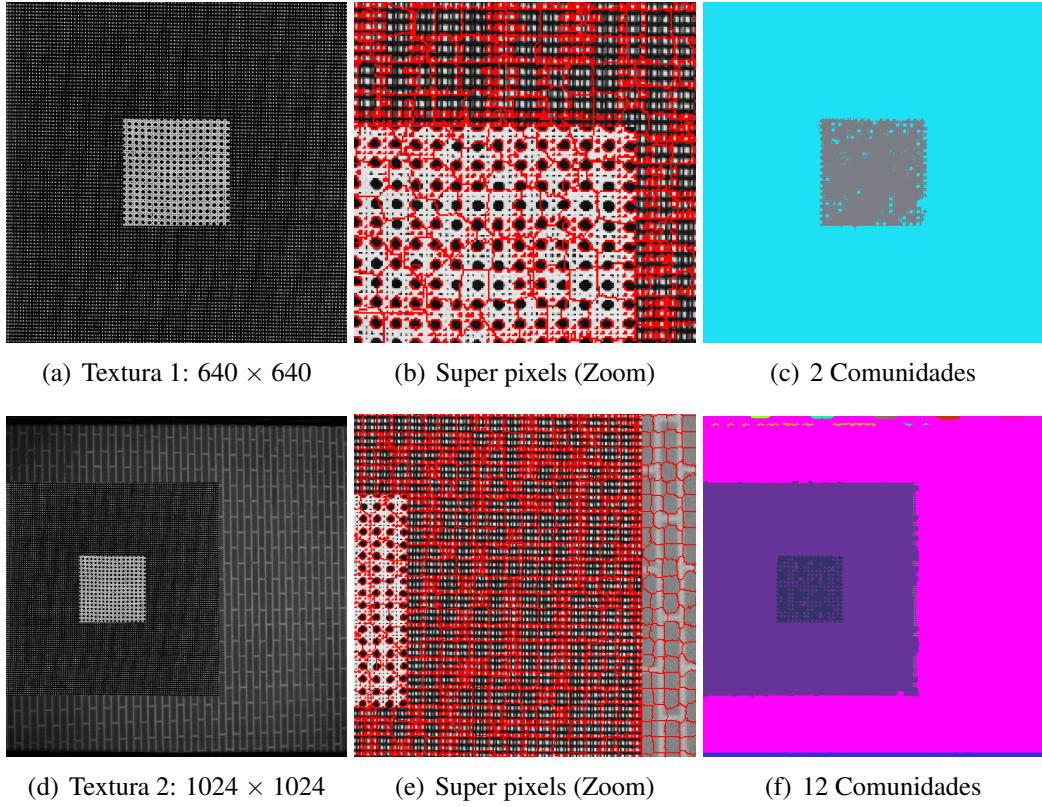


Figura 6.5: Segmentação de duas imagens por meio da técnica *Local Binary pattern*.

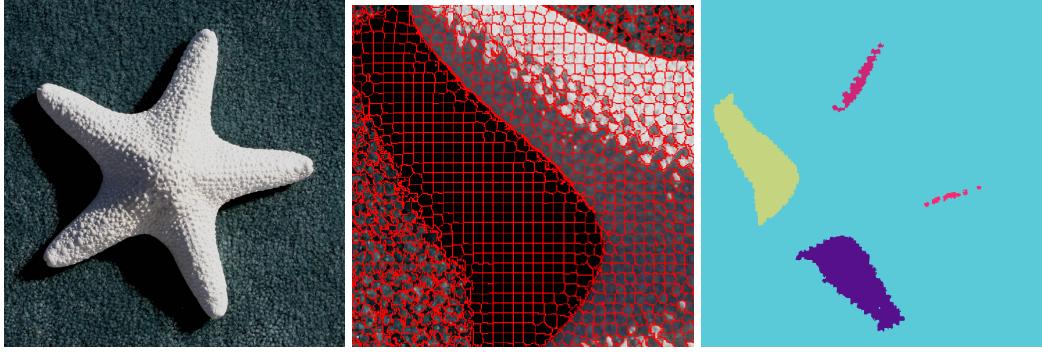
6.3.2 Estrela do Mar

Este experimento mostra o resultado obtido sobre uma imagem real de tamanho 1024×1024 . A imagem da Figura 6.6.a possui dois tipos de textura: a estrela e o fundo do mar apresentam texturas muito similares, mas diferentes cores. E a sombra da estrela apresenta outro tipo de textura. O valores dos parâmetros tanto para gerar os super pixels quanto para gerar o grafos são os mesmos valores usados nos experimentos com as imagens sintéticas.

O algoritmo *Fast Greed* retornou como resultado 5 comunidades, como mostra a Figura 6.6.c. O fundo do mar e a estrela, mesmo que objetos diferentes, foram considerados membros da mesma comunidade. O aparente erro de segmentação acontece devido a que ambos objetos possuem texturas muito semelhantes, assim é possível afirmar que o resultado é o esperado.

O tempo de processamento para todos os experimentos apresentados nesta seção são mostrados na Tabela 6.5.

Embora os tempos para gerar super pixels e detetar as comunidades sejam baixos, tempo extra é utilizado para aplicar o método LBP em cada super pixel e extrair os histogramas a serem comparados no momento de criar o grafo. Assim, em geral, o tempo



(a) Imagem Original 1004×1024 (b) Super pixels 15×15 (zoom) (c) Resultado: 5 comunidades

Figura 6.6: Segmentação de uma imagem real por meio da técnica *Local Binary Pattern*.

Imagen	Super Pixel	Grafo	Comunidades	Total
6.5.a	0.94	31.15	3.25	35.34
6.5.d	2.33	179.56	3.94	185.83
6.6.a	2.35	162.00	9.40	173.75

Tabela 6.5: Tempos de processamento em segundos para os experimentos usando textura.

para segmentar imagens usando textura é maior quando comparado com o tempo utilizado para segmentar imagens com as propriedades de cor.

6.4 Experimento 4: *Quadtree Speeded-up Turbo Pixels*

Neste experimento são apresentados os resultados obtidos para duas imagens pré-segmentadas por meio da técnica *Quadtree Speeded-up Turbo Pixels* proposta neste trabalho, descrita na Subseção 3.3.1. As segmentações são comparadas com os resultados obtidos sobre as mesmas imagens, mas com a pré-segmentação regular. Para todos os casos foram usados os mesmos parâmetros: $\lambda_1 = 1$, $\lambda_2 = 0.09$, raio 5 e limiar adaptativo 7 com incremento de 5% (Subseção 5.2). O modelo de cor usado neste experimento é o CIELAB, tanto para gerar os super pixels quanto para montar o grafo.

A Figura 6.7 mostra os resultados obtidos com a imagem do *Convento de Santa Catalina*. A Figura 6.7.a é a imagem original. As imagens 6.7.b, 6.7.c e 6.7.d são os resultados obtidos com o método *Quadtree Speeded-up Turbo Pixels*. As imagens 6.7.e, 6.7.f e 6.7.g ilustram os resultados gerados com o método *Speeded-up Turbo Pixels* original.

A Figura 6.8 mostra os resultados obtidos com a imagem da *Rupicola Peruviana*. Cuja imagem original é ilustrada na Figura 6.7.a. As imagens 6.8.b, 6.8.c e 6.8.d são

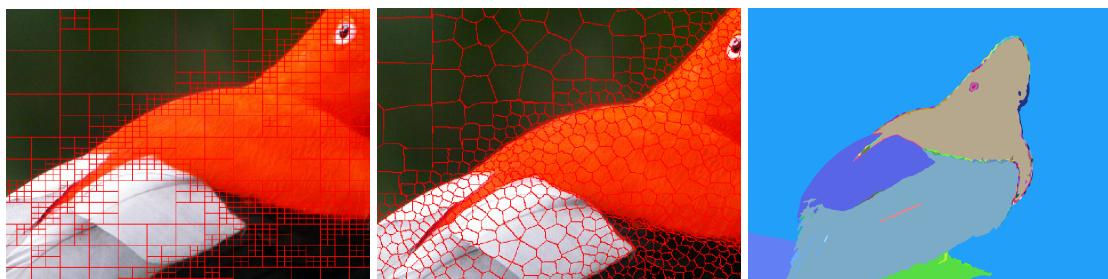


Figura 6.7: Experimento 1: Comparação do processo de geração de super pixels mediante a técnica *Quadtree Speeded-up Turbo Pixels* (b, c, d) e a técnica *Speeded-up Turbo Pixels* original (e, f, g). Além do resultado do algoritmo *Fast Greed* para ambas técnicas. (STP = Speeded-up Turbo Pixels)

os resultados obtidos com o método *Quadtree Speeded-up Turbo Pixels*. As imagens 6.7.e, 6.7.f e 6.7.g mostram os resultados gerados com o método *Speeded-up Turbo Pixels* original.

A Tabela 6.6 mostra a comparação do tempo do processamento para os dois experimentos apresentados anteriormente.

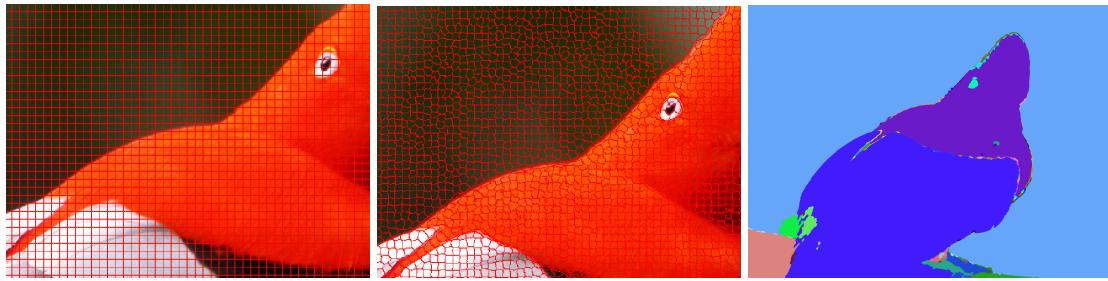
Os experimentos mostraram que a técnica *Speeded-up Turbo Pixels* gera super pixels de boa qualidade tanto para a grade regular quanto para a baseada no *Quadtree*. Mas a quantidade de super pixels diminui drasticamente quando é usado o *Quadtree*. Esta

(a) Imagem Original 1280×960 

(b) Quadtree (zoom)

(c) STP Quadtree (zoom)

(d) Resultado: 30 Comunidades



(e) Grade Regular (Zoom)

(f) STP original (zoom)

(g) Resultado: 35 comunidades

Figura 6.8: Experimento 2: Comparação do processo de geração de super pixels mediante a técnica *Quadtree Speeded-up Turbo Pixels* (b, c, d) e a técnica *Speeded-up Turbo Pixels* original (e, f, g). Além do resultado do algoritmo *Fast Greed* para ambas técnicas. (STP = Speeded-up Turbo Pixels)

Imagen	STP Original				STP Quadtree			
	SP	Grafo	FG	Total	SP	Grafo	FG	Total
Santa Catalina	1.21	1.03	0.25	2.49	1.69	0.59	0.14	2.42
Rupicola Peruviana	2.01	3.27	1.95	7.23	1.45	0.99	0.33	2.77

Tabela 6.6: Tempos de processamento em segundos para comprar o método *Speeded-up Turbo Pixels* original e nossa proposta *Quadtree Speeded-up Turbo Pixels*. (SP = Super pixels, FG=Fast Greed).

diminuição no número de super pixels se traduz na diminuição do tempo de processamento. Além disso, os resultados mostram que o tempo de processamento diminui significativamente quando a imagem possui grandes regiões similares, como por exemplo o fundo na imagem da *Rupicola Peruviana*. Por outro lado, a qualidade da segmentação final (resultado do algoritmo Fast Greed) pode-se notar uma melhora nas imagens pré-segmentadas com o método *Quadtree Speeded-up Turbo Pixels*. Neste caso a qualidade é avaliada apenas verificando a quantidade de regiões de diferentes cores misturadas em uma só comunidade. Os resultados do método baseado no *Quadtree* apresentaram menor quantidade de regiões misturadas. Notar que os parâmetros usados nos experimentos não foram alterados em nenhum dos casos.

6.5 Experimento 5: Avaliação Qualitativa

Os experimentos apresentados nesta seção têm por objetivo avaliar a qualidade das segmentações obtidas pelo método proposto, para diversos valores dos parâmetros utilizados pelo método. Para este experimento foram usadas as 300 imagens do conjunto Berkeley, onde cada imagem foi testada variando o raio de 1 a 5, limiar adaptativo de 0.5 a 40 (incrementos de 0.5), tamanho inicial dos super pixels 10×10 , $\lambda_1 = 1$ e $\lambda_2 = 0.09$ e modelo de cor CIELAB, tanto para gerar os super pixels quanto para montar o grafo. Assim, foram obtidas 400 segmentações diferentes para cada imagem e 120000 segmentações no total. Dado que o número de resultados obtidos é muito grande, foi preciso criar um banco de dados em *Postgresql*² para armazenar os dados. A média do tempo total de processamento para cada imagem foi de 0.3059 segundos.

Todos os resultados obtidos foram comparados com as segmentações manuais fornecidas pelo conjunto de imagens Berkeley. Como descrito na Seção 5.4.1 foi selecionada apenas uma segmentação manual por imagem e foi comparada com as 400 segmentações obtidas para cada imagem do Berkeley. A comparação dos resultados com as segmentações manuais foi realizada por meio de métrica proposta neste trabalho, descrita na seção 4.2. A seguir são apresentados os resultados obtidos das consultas executadas no banco de dados.

6.5.1 Parâmetro Raio

Como explicado na Seção 5.2, o raio é o parâmetro usado para evitar criar conexões entre super pixels muitos distantes na imagem, além de não ser avaliados pela função de similaridade. Para analisar o comportamento do raio, foi executada uma consulta que

²<http://www.postgresql.org/>

retorne a melhor qualidade (maior valor produzido pela MSSI) para cada imagem, com valores do raio entre 1 e 5. A partir do resultado da consulta foram plotados 5 histogramas, nos quais, o eixo x representa a qualidade no intervalo de 0 a 1 onde o valor 1 representa a máxima qualidade. Os histogramas são mostrados na Figura 6.9.

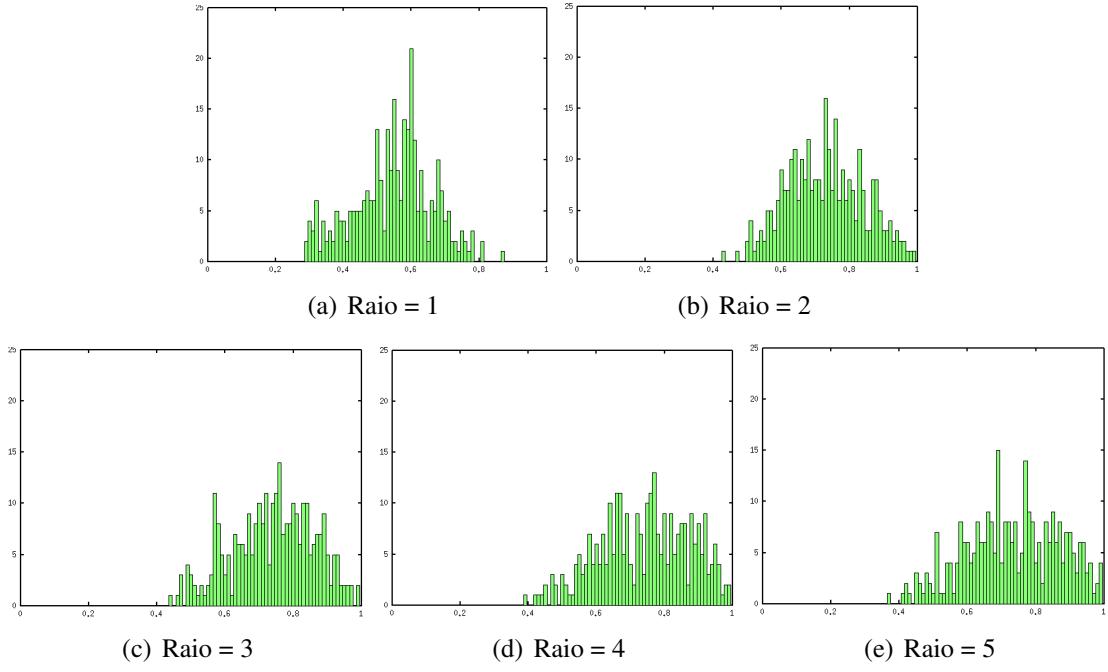


Figura 6.9: Histogramas gerados para as qualidades mais altas variando os valores do raio. O eixo x é a qualidade no intervalo de 0 a 1.

Os histogramas da Figura 6.9 revelaram que as maiores qualidades foram obtidas para valores do raio entre 4 e 5 e as mais baixas para os raios 1 e 2. Além disso, a média para o histograma do raio 4 é 0.74075 e o desvio padrão é 0.13260. A média para o histograma do raio 5 é 0.73129 e o desvio padrão é 0.13822. Pode-se observar que os valores da média e o desvio padrão para ambos raios são muito parecidos, $4 \approx 5$. Assim, é possível afirmar que os melhores resultados podem ser obtidos tanto para o raio 4 quanto para o raio 5.

6.5.2 Parâmetro Limiar

O limiar é um parâmetro adaptativo, que determina a criação de uma aresta entre dois super pixels, além de estabelecer o peso da aresta criada. Detalhes deste parâmetro são descritos na Seção 5.2. Para a análise deste parâmetro foi executada uma consulta no banco de dados, para extrair o valor do limiar usado no resultado da maior qualidade, para os 5 valores do raio. O resultado da consulta é ilustrado nos histogramas da Figura 6.10.

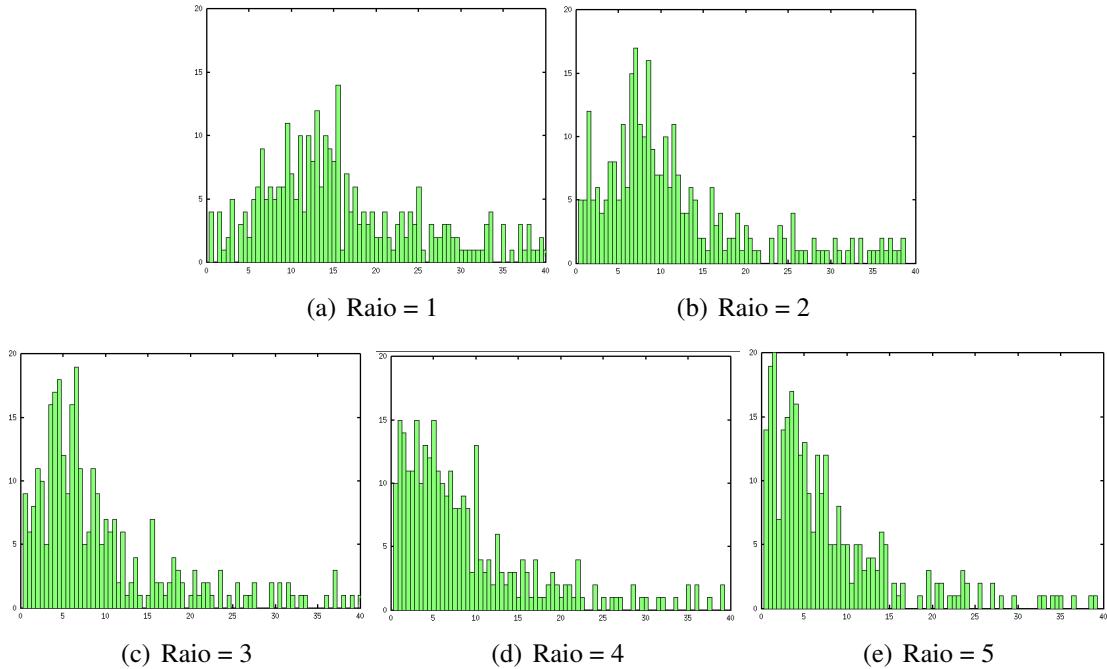


Figura 6.10: Histogramas gerados para os limiares usados nos resultados com a maior qualidade, variando os valores do raio. O eixo x são os valores dos limiares no intervalo de 0 a 40.

O gráficos da Figura 6.10 mostraram que as segmentações com a maior qualidade foram geradas com limiares no intervalo de 0.5 a 15 aproximadamente, para todos os valores do raio. Além disso, os valores da média e o desvio padrão dos histogramas para o raio 3,4,e 5 são muito semelhantes, como mostra a Tabela 6.7.

A Tabela 6.7 mostra os valores da média e o desvio padrão para todos os histogramas.

	Raio = 1	Raio = 2	Raio = 3	Raio = 4	Raio = 5
μ	15.7492	11.5100	9.4732	8.8512	7.7157
σ	9.1727	8.8453	8.3473	8.0659	7.6025

Tabela 6.7: Média e Desvio Padrão dos histogramas da Figura 6.10.

Portanto, é possível concluir que os valores ótimos do limiar estão no intervalo de 0.5 a 15 e que o parâmetro tem pouca influência no resultado final do processo de segmentação de imagens proposto neste trabalho.

6.5.3 Parâmetros do método *Speeded-up Turbo Pixels*

Todos as imagens testadas neste experimento foram pré-segmentadas com a técnica *Speeded-up Turbo Pixels* mantendo os mesmos valores dos parâmetros: $\lambda_1 = 1$, $\lambda_2 = 0.09$ e tamanho inicial dos super pixels de 10×10 . Em todas as imagens notou-se uma correta convergência dos super pixels, ou seja, foram obtidos super pixels de boa qualida-

dade. Notar que o parâmetro λ_1 tem valor 1 em todos os casos, e o λ_2 um valor baixo. Assim podemos concluir que o λ_1 comporta-se como uma constante fixa e um valor baixo do λ_2 , como por exemplo 0.09, garante ótimos resultados.

6.6 Experimento 6: *Graph Based Image Segmentation*

Nossos resultados foram comparados com o método de segmentação de imagens baseado em grafos *Graph Based Image Segmentation*, apresentado na Seção 1.1. As imagens foram testadas com dois conjuntos de parâmetros sugeridos pelos autores do método, em seguida foi calculada a qualidade mediante a comparação com segmentações manuais, processo explicado na Subseção 5.4.1. Para ambos conjuntos de parâmetros a média da qualidade foi muito similar, 0.52 e 0.48, que são médias baixas quando comparadas com o método proposto neste trabalho (Experimento 6.5.1), além de ser valores próximos ao aleatório. A Figura 6.11 mostra 4 exemplos das segmentações geradas com o método *Graph Based Image Segmentation* em comparação com nossos resultado para as mesmas imagens.

6.7 Considerações Finais

Este capítulo apresentou os experimentos realizados para avaliar os diferentes aspectos da abordagem proposta. Os experimentos mostraram que o método *Speeded-up Turbo Pixels* gera super pixels de boa qualidade, tanto para o algoritmo original quanto para a extensão baseada no *Quadtree*. Além disso, os experimentos mostraram que o parâmetro λ_1 pode ser fixado com o valor 1 e que o λ_2 precisa apenas de valores baixos, como por exemplo 0.009, e que o tamanho ótimo dos segmentos iniciais encontra-se no intervalo de 10×10 a 15×15 . No caso da segmentação inicial baseada no *Quadtree*, é preciso controlar o menor tamanho dos segmentos no mesmo intervalo de 10×10 a 15×15 e manter os mesmos parâmetros λ_1 e λ_2 . Por outro lado, os experimentos com o método *Quadtree Speeded-up Turbo Pixels* mostraram que a quantidade de super pixels apresenta uma diminuição importante, o que permite reduzir o tempo de processamento, mas esta redução no tempo é significativa quando a imagem possui grandes regiões similares.

Os experimentos com a técnica *Local Binary Pattern* para extrair a textura dos super pixels, mostraram que o método é capaz de extrair corretamente a textura de super pixels de tamanho maior ou igual a 15×15 . Além disso, os experimentos mostraram que o tempo de processamento incrementou-se significativamente, quando comparado com o tempo usando cor. Embora a técnica *Local Binary Pattern* seja capaz de extrair a textura dos super pixels, não foi obtida uma correta extração de super pixels em todas as imagens

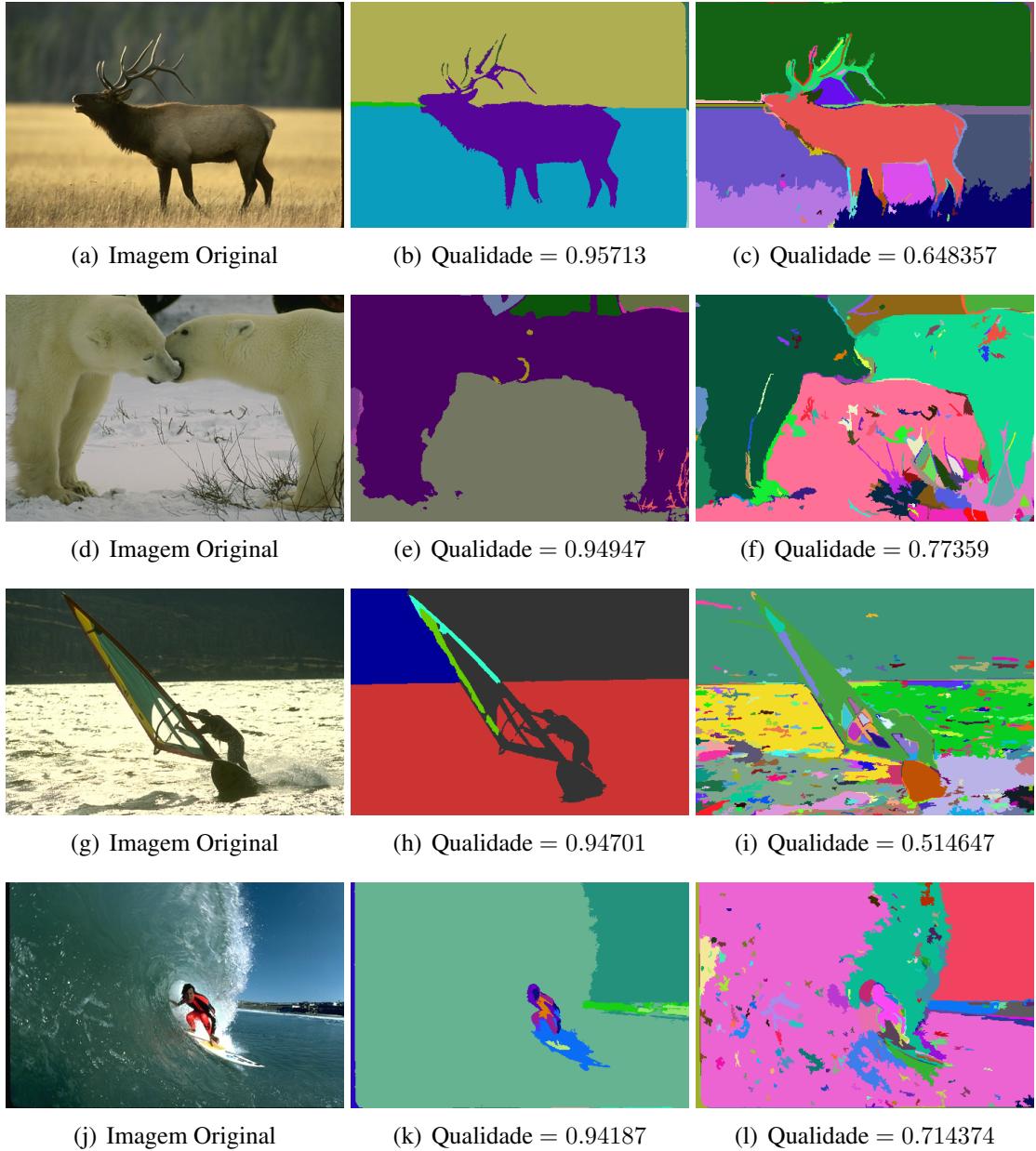


Figura 6.11: A primeira coluna mostra as imagens originais, a segunda coluna nossos resultados e a terceira coluna os resultados gerados pelo método *Graph Based Image Segmentation*.

com textura, devido à convergência ser realizada mediante a comparação da cor dos pixels e a textura não é considerada para o processo. Mesmo assim, foram obtidos resultados promissores.

Outro aspecto muito importante testado neste capítulo, é o compromisso dos parâmetros para montar o grafo e a qualidade da segmentação obtida. Os experimentos mostraram que o raio é o parâmetro que mais influencia na qualidade da segmentação, mas também mostraram que qualidades muito altas (0.7 - 0.9) são obtidas para valores do raio entre

4 e 5. Por outro lado, o limiar adaptativo é um parâmetro que não tem muita influência na qualidade da segmentação, mas os melhores resultados foram obtidos com valores do limiar no intervalo de 0.5 a 15.0. Para fins ilustrativos, mais exemplos de imagens segmentadas pelo método proposto são apresentados no Apêndice A.

O último experimento apresentou uma comparação da abordagem proposta neste trabalho com o método *Graph Based Image Segmentation*, os resultados mostraram as segmentações geradas pelo nosso método apresentam melhor qualidade e que o tempo de processamento é menor ou igual em todos os casos.



CAPÍTULO

7

Conclusões

Este trabalho explorou uma nova abordagem de segmentação de imagens de alta dimensão por meio de redes complexas e super pixels. Os resultados mostraram que esta combinação das técnicas fornece segmentações de imagens de alta dimensão precisas num tempo de processamento baixo. Diversas técnicas foram estudadas, tanto para a geração de super pixels quanto para a detecção de comunidades, além de descritores de textura e métricas para comparar segmentações. Os experimentos realizados ao longo do desenvolvimento deste projeto, mostraram que a técnica para gerar super pixels *Speeded-up Turbo Pixels* proposta produz resultados bastante satisfatórios. Em virtude da versatilidade do algoritmo, este pode ser aperfeiçoado mudando o modelo de cor ou iniciando o processo com segmentos de tamanhos diferentes como por exemplo o *Quadtree*. Assim a técnica proposta torna-se capaz de produzir melhores resultados e a influência dos parâmetros diminui significativamente.

Dos algoritmos de detecção de comunidades estudados, o *Fast Greedy* é o algoritmo que produz os melhores resultados. O algoritmo detecta comunidades em tempo de processamento equivalente ao algoritmo *Label Propagation*, mas a principal vantagem é a precisão das comunidades retornadas. O algoritmo *Label Propagation* apresenta a limitação de super-segmentar a rede, o que se traduz numa sobre-segmentação da imagem. No *Fast Greedy*, o problema da sobre-segmentação praticamente inexiste. Destaca-se que no processo de segmentar imagens, a sobre-segmentação é um problema que pode ser resolvido num pós-processamento. Por outro lado, a fusão de regiões é um problema muito mais difícil de resolver. Nenhum dos algoritmos estudados apresentou este último

problema.

Em relação ao grafo, nossa abordagem usa dois parâmetros que influenciam o resultado final: o raio e o limiar. Os experimentos mostraram que o parâmetro que mais influencia na qualidade das segmentações é o raio, mas os testes também mostraram que é possível garantir ótimos resultados para valores do raio 4 ou 5. Por outro lado, os experimentos revelaram que o limiar não tem muita influencia no resultado, mas as qualidades mais altas foram obtidas para valores no intervalo de 0.5 a 15.0. Acreditamos que a determinação dos “melhores” valores para os parâmetros raio e liminar, e a fixação dos parâmetros λ_1 e λ_2 para as segmentações mais precisas, constitui-se em uma das principais contribuições deste trabalho, uma vez que torna o método bastante independente dos parâmetros, o que não fora alcançado antes em nenhuma pesquisa da área.

Finalmente, nossos resultados apresentaram melhor qualidade quando foram comparados com o método GBIS. A média da qualidade do nosso método é de 0.7 enquanto a média do GBIS é 0.5.

7.1 Limitações

A segmentação de imagens é uma das tarefas mais complexas do processamento de imagens. As subjetividade implícita no processo, além do número muito grande de tipos de imagens e as diversas propriedades, como por exemplo a cor, textura, forma ou tamanho, a serem consideradas, fazem do processamento de imagens um dos problemas mais difíceis de resolver.

Assim, nossa abordagem, principalmente baseada na cor das imagens, é dependente da cor das regiões a serem segmentadas. Ou seja, podem existir regiões diferentes, mas com a mesma cor, as quais poderiam ser incluídas na mesma comunidade.

Uma outra questão a ser considerada é a textura das imagens. Os experimentos mostraram resultados muito promissores, mas ainda o tempo de processamento é excessivo e as segmentações para imagens reais não geraram os resultados esperados. Esta limitação deve-se principalmente ao fato de que os algoritmos de Super Pixels realizam o processo de convergência com base na cor dos segmentos e pixels a serem avaliados, ignorando a textura destes.

7.2 Contribuições

As principais contribuições deste trabalho são:

- Desenvolvimento de uma nova abordagem de segmentação de imagens por meio de redes complexas e super pixels, a qual permite segmentar imagens de alta di-

mensão em baixo tempo de processamento, além de apresentar ótima qualidade nas segmentações.

- Aprimoramento da técnica *Speeded-up Turbo Pixels*. No algoritmo original a convergência dos super pixels realizava a convergência com base no nível de cinza dos pixels testados. A nova versão é capaz de convergir mediante diversos modelos de cor, como por exemplo RGB ou CIELAB, além do nível de cinza. Tal alteração permite gerar super pixels em menor número de iterações e tempo de processamento.
- Desenvolvimento da técnica *Quadtree Speeded-up Turbo Pixels* baseada na técnica *Speeded-up Turbo Pixels*, a qual permite iniciar a convergência dos super pixels a partir de uma grade melhor adaptada à imagem, esta grade é baseada na estrutura de dados *Quadtree*. A aplicação desta técnica, permite diminuir a quantidade de segmentos, principalmente em imagens com grandes regiões similares.
- O desenvolvimento de uma nova métrica (MSSI) para medir a interseção entre segmentações.
- Estudo da influência dos parâmetros usados na abordagem proposta neste trabalho e a qualidade da segmentação.

7.3 Trabalhos Futuros

Em trabalhos futuros pretende-se investigar um método de convergência de super pixels que seja capaz de extrair super pixels por meio da textura da imagem ou uma combinação entre textura e cor.

Em relação ao algoritmo *Speeded-up Turbo Pixels* pretende-se desenvolver uma extensão para aplicar a técnica em imagens 3D, ou seja propor uma abordagem para extrair super voxels denominada *Speeded-up Turbo Voxels*.

Melhorar o processo de propagação de rótulos da técnica *Label Propagation* para evitar a sobre-segmentação das redes analisadas. O algoritmo propaga os rótulos dos nós com base nos rótulos de seus vizinhos em um tempo $t - 1$, ou seja considera os rótulos dos nós avaliados na iteração anterior para rotular o nó atual (tempo t). Mas, a propagação não é apenas baseada nos rótulos dos vizinhos. Quando não é possível definir o maior número de rótulos dos vizinhos (empate), não é usada essa informação para rotular o nó atual. Nesse caso o nó é rotulado aleatoriamente, selecionando o rótulo de qualquer vizinho. Assim, um trabalho futuro, é aplicar um modelo estatístico na propagação dos rótulos, baseado nas propriedades de Markov. A fim de evitar que a seleção de rótulos seja apenas aleatória.

Estudar novas formas e métricas para aperfeiçoar o processo de geração da rede a partir de super pixels. Até agora, foi explorada a criação do grafo mediante a média da cor ou descritores de textura dos super pixels. Como mostrado ao longo desta dissertação, os resultados obtidos foram de boa qualidade, mas acredita-se que o processo pode ser ainda aperfeiçoado. É possível considerar outras características dos super pixels, como por exemplo: a variação da gradiente no momento de ligar dois super pixels, ponderar a distância espacial dos super pixels, considerar informações fornecidas pelo usuário (processo supervisionado), entre outras. Por outro lado, em relação as métricas, na literatura existe uma grande diversidade destas, estudadas no agrupamento de dados em redes complexas (Rodrigues et al., 2011). Estas métricas podem ser avaliadas no contexto de segmentação de imagens por meio de redes complexas.

Apêndice A

A seguir são mostrados alguns resultados gerados pela abordagem de segmentação de imagens proposta neste trabalho.



Figura 7.1: Duas colunas de imagens. (Imagen Original - Segmentação).

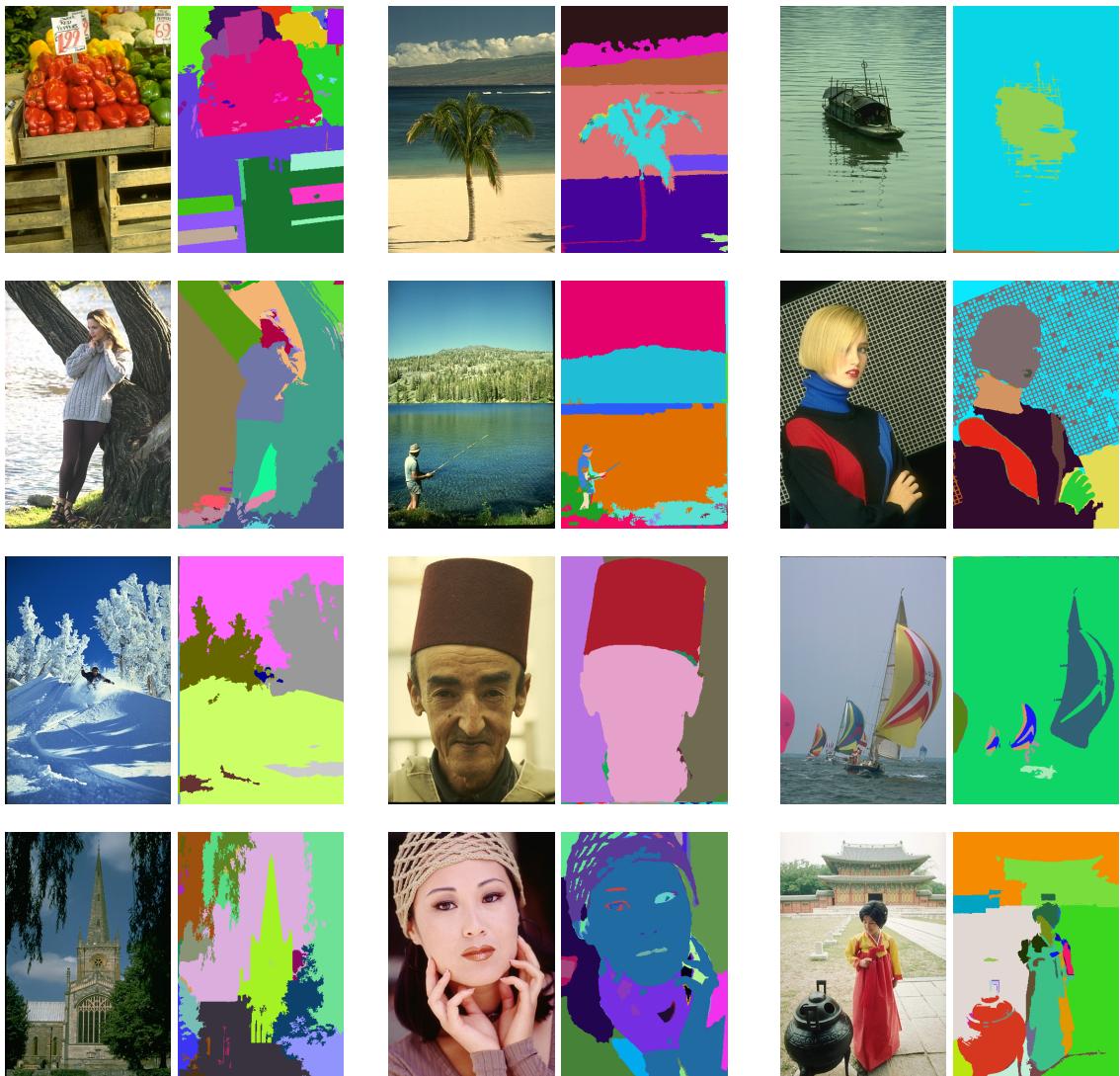


Figura 7.2: Três colunas de imagens. (Imagen Original - Segmentação).

Referências Bibliográficas

- ACHANTA, R.; SHAJI, A.; SMITH, K.; LUCCHI, A.; FU, P.; SÜSSTRUNK, S. Slic superpixels. *Technical Report 149300 École Polytechnique Fédéral de Lausanne*, 2010.
- ARBELÁEZ, P.; MAIRE, M.; FOWLKES, C.; MALIK, J. From contours to regions: An empirical evaluation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, p. 2294–2301.
- BAKAS, I. The algebraic structure of geometric flows in two dimensions. *Journal of High Energy Physics*, v. 2005, n. 10, p. 038, 2005.
- BARNES, E. An algorithm for partitioning the nodes of a graph. In: *20th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, 1981, p. 303–304.
- BELIZARIO, I. V. *Avaliação de algoritmos de agrupamento em grafos para segmentação de imagens*. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2012.
- BRANDES, U.; ERLEBACH, T. *Network analysis: Methodological foundations*. Springer-Verlag New York, Inc., 2005.
- CHALUMEUA, T.; COSTAB, L.; LALIGANTA, O.; MERIAUDEUA, F. Complex networks: application for texture classification. In: *8th International Conference on Quality Control by Artificial Vision*, 2007, p. 63561E–63561E.
- CIGLA, C.; ALATAN, A. Efficient graph-based image segmentation via speeded-up turbo pixels. In: *17th IEEE International Conference on Image Processing (ICIP)*, 2010, p. 3013–3016.

- CLAUSET, A.; NEWMAN, M. E.; MOORE, C. Finding community structure in very large networks. *Physical review E*, v. 70, n. 6, p. 066111, 2004.
- COUR, T.; BENEZIT, F.; SHI, J. Spectral segmentation with multiscale graph decomposition. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, p. 1124–1131.
- CUADROS, O.; BOTELHO, G.; RODRIGUES, F.; NETO, J. B. Segmentation of large images with complex networks. *25th SIBGRAPI Conference on Graphics, Patterns and Images*, v. 0, p. 24–31, 2012.
- FELZENZWALB, P.; HUTTENLOCHER, D. Efficient graph-based image segmentation. *International Journal of Computer Vision*, v. 59, n. 2, p. 167–181, 2004.
- FIDUCCIA, C. M.; MATTHEYSES, R. M. A linear-time heuristic for improving network partitions. In: *19th IEE Conference on Design Automation*, 1982, p. 175–181.
- FINKEL, R. A.; BENTLEY, J. L. Quad trees: A data structure for retrieval on composite keys. *Acta Informática*, v. 4, p. 1–9, 1974.
- FORTUNATO, S. Community detection in graphs. *Physics Reports*, v. 486, n. 3, p. 75–174, 2010.
- FORTUNATO, S.; CASTELLANO, C. Community structure in graphs. In: *Encyclopedia of Complexity and Systems Science*, Springer, p. 1141–1163, 2009.
- GONZALEZ, R.; WOODS, R. *Digital image processing*. Pearson/Prentice Hall, 2009.
- HARALICK, R.; SHAPIRO, L. Image segmentation techniques. *Computer vision, graphics, and image processing*, v. 29, n. 1, p. 100–132, 1985.
- KARGER, D. Minimum cuts in near-linear time. *Journal of the ACM (JACM)*, v. 47, n. 1, p. 46–76, 2000.
- KERNIGHAN, B.; LIN, S. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, v. 49, n. 2, p. 291–307, 1970.
- LEVINSHTEIN, A.; STERE, A.; KUTULAKOS, K. N.; FLEET, D. J.; DICKINSON, S. J.; SIDDIQI, K. Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 31, n. 12, p. 2290–2297, 2009.
- MÄENPÄÄ, T.; OJALA, T.; PIETIKÄINEN, M.; SORIANO, M. Robust texture classification by subsets of local binary patterns. In: *15th International Conference on Pattern Recognition*, 2000, p. 935–938.

- NAVON, E.; MILLER, O.; AVERBUCH, A. Color image segmentation based on adaptive local thresholds. *Image and vision computing*, v. 23, n. 1, p. 69–85, 2005.
- NEWMAN, M. The structure and function of complex networks. *SIAM review*, v. 45, n. 2, p. 167–256, 2003.
- NEWMAN, M. Fast algorithm for detecting community structure in networks. *Physical Review E*, v. 69, n. 6, p. 066133, 2004.
- NEWMAN, M.; GIRVAN, M. Finding and evaluating community structure in networks. *Physical review E*, v. 69, n. 2, p. 026113, 2004.
- NOBLE, J.; BOUKERROUI, D. Ultrasound image segmentation: A survey. *IEEE Transactions on Medical Imaging*, v. 25, n. 8, p. 987–1010, 2006.
- OGNIEWICZ, R.; KÜBLER, O. Hierarchic voronoi skeletons. *Pattern recognition*, v. 28, n. 3, p. 343–359, 1995.
- OJALA, T.; PIETIKAINEN, M.; MAENPAA, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 7, p. 971–987, 2002.
- OLIVEIRA, T.; ZHAO, L. Complex network community detection based on swarm aggregation. In: *4th International Conference on Natural Computation (ICNC)*, 2008, p. 604–608.
- OLIVEIRA, T.; ZHAO, L.; FACELI, K.; CARVALHO, A. Data clustering based on complex network community detection. In: *IEEE Congress on Evolutionary Computation*, 2008, p. 2121–2126.
- PALLA, G.; DERÉNYI, I.; FARKAS, I.; VICSEK, T. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, v. 435, n. 7043, p. 814–818, 2005.
- POTHEN, A. Graph partitioning algorithms with applications to scientific computing. *ICASE LaRC Interdisciplinary Series in Science and Engineering*, v. 4, p. 323–368, 1997.
- RAGHAVAN, U. N.; ALBERT, R.; KUMARA, S. Near linear time algorithm to detect community structures in large-scale networks. *Physics Review E*, v. 76, p. 036106, 2007.

- REKA, A.; BARABÁSI Statistical mechanics of complex networks. *Reviews of modern physics*, v. 74, n. 1, p. 47–97, 2002.
- REN, X.; MALIK, J. Learning a classification model for segmentation. In: *5th IEEE International Conference on Computer Vision*, 2003, p. 10–17.
- RODRIGUES, F.; ARRUDA, G.; COSTA, L. A complex networks approach for data clustering. *Physica A: Statistical Mechanics and its Applications*, 2011.
- SHI, J.; MALIK, J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22, n. 8, p. 888–905, 2000.
- SIDDIQI, K.; BOUIX, S.; TANNENBAUM, A.; ZUCKER, S. Hamilton-jacobi skeletons. *International Journal of Computer Vision*, v. 48, n. 3, p. 215–231, 2002.
- ZHANG, H.; FRITTS, J.; GOLDMAN, S. Image segmentation evaluation: A survey of unsupervised methods. *Computer Vision and Image Understanding*, v. 110, n. 2, p. 260–280, 2008.