Avaliação de algoritmos de agrupamento em grafos para segmentação de imagens

Ivar Vargas Belizario

Avaliação de algoritmos de agrupamento em grafos para segmentação de imagens

Ivar Vargas Belizario

Orientador: **Prof. Dr. João do Espirito 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
Novembro de 2012

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

VARGAS BELIZARIO, IVAR
V287a Avaliação de algoritmos de agrupamento em grafos para segmentação de imagens / IVAR VARGAS BELIZARIO; orientador João do Espirito Santo Batista Neto. -- São Carlos, 2012.
86 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, 2012.

1. Segmentação de imagens. 2. Agrupamento em grafos. 3. Qualidade de segmentação. 4. Qualidade de agrupamento. I. Batista Neto, João do Espirito Santo, orient. II. Título.

Aos meus pais
A meu amor por sempre Liz

Agradecimentos

Agradeço em primeiro lugar a Deus, por todo o que Ele me deu no tempo do mestrado.

Agradeço ao professor João do Espirito Santo Batista Neto por aceitar-me como orientando no mestrado, pelos conselhos e pela orientação nestos quase 3 anos, sempre lembrarei de sua boa personalidade e seu optimismo em todo este tempo, muito obrigado professor João.

Também quero agradecer ao professor Francisco Aparecido Rodrigues pela contribuição na realização desta dissertação.

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

Ao Instituto de Ciências Matemáticas e de Computação (ICMC/USP).

A todos os meus professores do ICMC, e em especial aos professores do grupo de pesquisa de Visualização, Imagens e Computação Gráfica (VICG).

E a todas aquelas pessoas com as quais foi possível o entendimento e culminação deste trabalho de mestrado. Particularmente quero agradecer a Renato, Vinícius e Alceu.

Resumo

A segmentação de imagens é, em visão computacional, uma tarefa de grande importância, para a qual existem várias abordagem. A complexidade de tais abordagens está relacionada à natureza da imagem e também ao grau de precisão da segmentação, que é um conceito bastante subjetivo, normalmente associado à semelhança que apresenta à segmentação produzida pela visão humana. Na segmentação de imagens baseada em algoritmos de agrupamento em grafos, geralmente os *pixels* da imagem compõem os nós do grafo e as arestas representam a similaridade entre estes nós. Assim, a segmentação pode ser obtida por meio do agrupamento dos nós do grafo. É importante salientar, no entanto, que as técnicas de agrupamento em grafos surgiram no contexto de reconhecimento de padrões, cujo objetivo primário era o tratamento de dados diversos que não envolviam imagens. O uso de tais técnicas para a segmentação de imagens é relativamente recente e revela alguns problemas desafiadores. O primeiro deles é a deficiente escalabilidade de alguns métodos, o que impede o seu uso efetivo em imagens de altas dimensões. Outra questão é a falta de estudos que avaliam as medidas de similaridade na montagem do grafo e critérios que aferem a qualidade do agrupamento para a área específica de segmentação de imagens. Em outras palavras, faltam na literatura análises mais específicas que indiquem quais algoritmos de agrupamento em grafos são mais efetivos para a segmentação de imagens e que procurem associar (ou correlacionar) as várias medidas de similaridade e métricas de qualidade de agrupamento que produzam segmentações mais precisas. Neste trabalho é apresentada a avaliação de 6 algoritmos de agrupamento em grafos formulados em base a 3 categorias identificadas (agrupamento espectral, algoritmos de particionamento multinível e algoritmos para detectar comunidades) e aplicadas na segmentação automática de imagens de cenas naturais com grandes dimensões. Esta avaliação objetiva aferir, sobretudo, a qualidade da segmentação, a escalabilidade, o desempenho de 7 funções de similaridade formuladas, e também visa corroborar a existência da correlação entre a qualidade do agrupamento e a qualidade da segmentação. Para reduzir o esforço computacional e contribuir com a escalabilidade dos algoritmos formulados é utilizado um algoritmo de pre-processamento (SLIC) que agrupa vários *pixels* da imagem em uma única região (*superpixels*), o que contribui para reduzir o tamanho do grafo e, consequentemente, reduzindo o custo computacional do agrupamento. Os resultados demostram que os algoritmos formulados LP (Label Propagation) e FG (Fast Greedy) apresentam boa escalabilidade e boa qualidade de segmentação. Seis das sete funções de similaridade avaliadas apresentam um bom desempenho, independentemente do algoritmo de agrupamento empregado. É mostrado também que exites correlação entre a medida de qualidade de agrupamento conhecido como índice de silhueta e a qualidade de segmentação, ou seja, quanto maior o valor de silhueta, melhor a segmentação. A qualidade de segmentação foi avaliada quantitativamente, utilizando-se um conjunto de imagens segmentadas manualmente.

Palavras-chave: Segmentação de imagens; Agrupamento em grafos; Qualidade de segmentação; Qualidade de agrupamento.

Abstract

Image segmentation is an important task within computer vision for which many approaches are available. The complexity of such approaches are intrinsically related with the nature of the image and also the desired accuracy aimed at. Image segmentation accuracy, however, is a subjective concept and is normally associated with how much it resembles segmentation produced by the human vision system. In graph-based clustering image segmentation algorithms, pixels are normally represented as nodes and edges convey the similarity between such nodes. Hence, segmentation may be attained by means of grouping node of a graph. It is important, though, to point out that graph-based clustering techniques first appeared in the context of pattern recognition and its primary data source were not images. The usage of such techniques for image segmentation is a recent trend and poses some challenge issues. The first is the poor scalability that many methods exhibit, impairing its application in images of larger dimensions. Another issues is that lack of studies that assess the goodness of similarity measures employed in graph computation and also clustering quality criteria assessments for the specific area of image processing. In other words, there is no evidences in the literature on how effective graph-based clustering algorithms are for image segmentation and no studies that associate similarity functions and clustering quality metrics with image processing quality. This work presents an evaluation of six graph-based clustering algorithms according to three major categories found in the literature (spectral clustering, multi layer partitioning algorithms and community detectors) applied to automatic segmentation of image of larger dimensions. This evaluation takes into account segmentation quality, scalability, the performance of seven similarity functions and, finally, bring some insights on the correlation between clustering and segmentation quality. To reduce computational costs and enhance scalability of the methods we employ a pre processing algorithm (SLIC) which combines many pixels into one single region (superpixel). This contributes to reduce the graph size and, consequently, the cost of clustering. Results have shown that the LP (Label Propagation) and FG (Fast Greedy) algorithms exhibit good scalability and good segmentation. Six out of the seven similarity functions presented good performance, regardless the clustering algorithm. We also have shown that there is correlation between clustering quality and image segmentation quality, when the Silhouette measure is employed. That is, the higher the Silhouette values, the better the segmentation. Image segmentation quality has been quantitatively assessed by means of ground-truth data or user segmented images.

Keywords: Image segmentation; Graph clustering; Segmentation quality; Grouping quality.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas e Siglas

MinCut	*Minimun Cut.*
NCut	*Normalized Cut.*
G	Grafo.
V	Nós do grafo.
E	Arestas do grafo.
W	Matriz de adjacência.
D	Matriz diagonal de graus.
SC	*Spectral clustering.*
GC	*Graclus.*
FG	*Fast greedy.*
LP	*Label propagation.*
Q	*Modularity.*
S_{index}	*Silhouette index.*
SUTP	Speeded-Up Turbo Pixels
SLIC	Simple Linear Iterative Clustering
TP	TurboPixel

Capítulo

1

Introdução

1.1 Contextualização e Motivação

A segmentação de imagens é o particionamento da imagem em múltiplas regiões, baseado em vários critérios como: **níveis de cinza**, **cor**, e **textura**. Ela tem importância na área de visão computacional por ser um dos primeiros processos para o reconhecimento de padrões. Formalmente é definida como o processo de particionar uma imagem em regiões não intersectadas, onde os *pixels* destas regiões compartilham propriedades semelhantes.

Dentro do contexto de segmentação de imagens encontra-se a abordagem baseada no **agrupamento perceptual**[1], como ilustra a Figura 1.1:

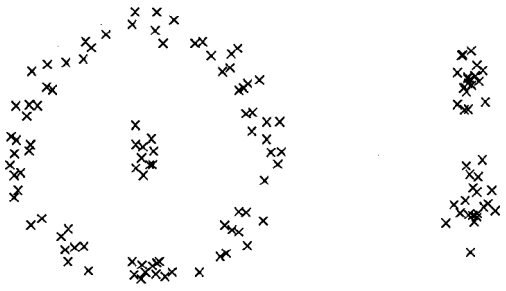


Figura 1.1: Agrupamento perceptual: quantos objetos podem ser encontrados? Adaptado de Shi e Malik (1997).

[1] Agrupamento Perceptual, baseia-se na teoria da Gestalt, apresentada por Wertheimer (1938). Nela se expôs a importância do agrupamento perceptivo e organização na percepção visual. **Similaridade**, **proximidade**, e **boa continuidade** são alguns dos principais fatores que esta teoria ressalta.

Na Figura 1.1, um observador pode, por exemplo, perceber quatro objetos na cena: um objeto circular, com outro objeto representado pelo agrupamento de pontos dentro deste, e dois grupos de pontos à direita. No entanto, este pode não ser o único agrupamento. Pode-se argumentar que existem três objetos, entendendo como um só, os dois grupos de pontos do lado direito. Também cabe a possibilidade da existência de somente dois objetos na cena: um objeto na direita, e outro no lado esquerdo. Também é possível argumentar que, na verdade, cada ponto poderia ser um objeto distinto.

Sob o ponto de vista de agrupamento perceptual, para a segmentação de imagens, qualquer tentativa para segmentar a imagem nos levaria à mesma conclusão: a existência de muitas possíveis partições no domínio da imagem. Um exemplo pode ser ilustrado analisando as imagens na Figura 1.2. No entanto surge a seguinte questão: como é que podemos escolher uma "boa" segmentação?

(a) **(b)**

Figura 1.2: Imagens empregadas na Teoria da *Gestalt*. Tanto (a) como (b), dão margem a duas interpretações.

O trabalho de Ren e Malik (2003) tenta responder esta questão: como definir uma boa segmentação? Nesse trabalho a questão foi abordada com os princípios da clássica Teoria da *Gestalt*, que são: proximidade, similaridade e boa continuidade. Uma segmentação com boa continuidade é aquela que possui boa:

1. similaridade intra-região: os elementos de uma mesma região são similares. Isso inclui similaridade de brilho, textura, e de baixa energia do contorno dentro da região;

2. dissimilaridade inter-região: os elementos de diferentes regiões são distintos. Isto, inclui brilho e textura diferentes, e de alta energia do contorno nos limites da região.

Uma proposta de solução para o problema gerado nesta última questão pode ser visto no trabalho de Shi e Malik (1997; 2000). Os autores propõem uma abordagem de segmentação de imagens baseada no problema do particionamento de grafos mediante o corte de grafos, como será discutido no capítulo 2.

Embora a solução para segmentação de imagens por meio de cortes em grafos tenha se mostrado um modelo promissor no que diz respeito ao agrupamento visual humano, este ainda apresenta alguns desafios. O principal deles é, sem dúvida, a não escalabilidade dos métodos. Como será mostrado mais adiante, a segmentação baseada no particionamento de grafos envolve uma enorme quantidade de nós e os primeiros trabalhos, na prática, se mostram aplicáveis apenas para imagens de baixa dimensão (na ordem de 100×100 *pixels*) (Shi e Malik, 2000).

1.2 Objetivos

A revisão da literatura revela 3 categorias principais de algoritmos aplicáveis ao problema de agrupamento em grafos: a) agrupamento espectral b) agrupamento multinível e c) algoritmos para identificar comunidades empregadas em redes complexas. O objetivo principal deste trabalho é estudar, implementar e avaliar algoritmos de segmentação mediante o agrupamento em grafos, respondendo as seguintes questões específicas:

- Quais das categorias de métodos descritas acima são mais adequadas para a segmentação de imagens, ou seja, quais algoritmos produzem segmentações com melhor qualidade?
- Quais métodos apresentam melhor escalabilidade e podem ser aplicados a imagens de grandes dimensões?
- Quais funções de similaridade, que são empregadas na criação dos grafos, são mais adequadas para a segmentação?
- Existe correlação entre as métricas que avaliam a qualidade do agrupamento em grafos e a qualidade da segmentação?

Para responder tais questões este trabalho investiga e sugere mecanismos para reduzir o tempo de execução dos algoritmos de particionamento e também sugere uma medida que avalia, quantitativamente, a qualidade de segmentação obtida.

Uma forma interessante de reduzir o esforço computacional na segmentação é por meio da aplicação de algoritmos de *superpixels* (Cigla e Alatan, 2010; Achanta et al., 2010, 2012). Estes algoritmos têm como objetivo agrupar um conjunto de *pixels* com propriedades semelhantes, no único *"pixel"*, conhecido por *superpixel*. Ao realizar este procedimento, o grafo que modela a imagem terá menor cardinalidade, consequentemente reduzindo o esforço computacional da segmentação. Portanto, outro objetivo deste trabalho é a implementação de algoritmos de *superpixels* para apoiar a segmentação de imagens de grandes dimensões por meio do agrupamento em grafos.

Para avaliação da qualidade da segmentação será utilizado o *ground truth*, *Berkeley Image Dataset*[2] (BSDS300) da universidade de *Berkeley*; que é composta por 300 imagens coloridas, cada imagem possui pelo menos 5 segmentações manuais. Esta avaliação está realizada por meio

[2]http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

da análise quantitativa da segmentação entre os resultados obtidos pelos algoritmos formulados e os resultados das segmentações manuais.

1.3 Organização do Trabalho

Esta dissertação está organizada da seguinte maneira:

- No Capítulo 2 é apresentada a revisão bibliográfica de algoritmos de agrupamento em grafos. Nesse capítulo são apresentados algoritmos que se adequam aos objetivos da presente dissertação.
- No Capítulo 3 é apresentada a revisão bibliográfica de algoritmos de *superpixel*. Nesse capítulo são apresentados algoritmos com complexidade computacional linear.
- No Capítulo 4 é apresentada a metodologia proposta para segmentar imagens de grandes dimensiones, empregando algoritmos de agrupamento em grafos.
- No Capítulo 5 são apresentados os resultados experimentais obtidos.
- Finalmente, no Capítulo 6, são apresentadas as considerações finais deste trabalho, as principais contribuições em segmentação de imagens utilizando agrupamento em grafos, as limitações e propostas de trabalhos futuros.

Capítulo

2

Algoritmos de Agrupamento em Grafos

2.1 Considerações Iniciais

No presente capítulo são apresentados os principais algoritmos de agrupamento baseados em grafos. Uma revisão de tais algoritmos de agrupamento em grafos pode ser obtida em Schaeffer (2007) e Aggarwal e Wang (2010). Segundo Aggarwal e Wang (2010), os algoritmos de agrupamento são classificados em duas categorias: os algoritmos baseados no agrupamento dos nós do grafo e os algoritmos baseados no agrupamento de um conjunto de grafos. No presente capítulo, serão abordados alguns algoritmos baseados no agrupamento dos nós. Esta classe de algoritmos foi selecionada levando em conta o objetivo do presente trabalho de mestrado: segmentação de imagens por grafos, em que os nós do grafo correspondem a pequenas regiões da imagem. Idealmente, algoritmos desta classe devem compartilhar as seguintes características: uma boa qualidade do agrupamento e escalabilidade.

A organização da revisão bibliográfica no presente capítulo é feita abordando 3 tipos de algoritmos de agrupamento baseados em grafos: algoritmos de agrupamento espectral, algoritmos de particionamento multinível e algoritmos de identificação de comunidades.

O presente capítulo está organizado como segue: Na Seção 2.2 são apresentados os principais conceitos de grafos e agrupamento em grafos. Na Seção 2.3 são apresentados algoritmos de agrupamento espectral. Na Seção 2.4 são apresentados algoritmos particionamento multinível. Na Seção 2.5 são apresentados algoritmos de agrupamento para identificar comunidades. Finalmente, na Seção 2.6, são apresentadas as considerações finais.

2.2 Conceitos Teóricos

Esta seção apresenta os principais conceitos referentes a grafos e agrupamento em grafos, que são importantes para a compreensão do tipo de técnica de segmentação de imagens adotada nesta dissertação.

2.2.1 Conceitos Básicos Sobre Grafos

Um grafo é definido por $G = (V,E)$, por um conjunto finito e não vazio V, cujos elementos são denominados **nós**, e um conjunto E de subconjuntos dos elementos de V, denominados **arestas**. Indicamos por $|V|$ e $|E|$, respectivamente, o número de nós e arestas de G, também conhecido como cardinalidade para ambos casos. Cada aresta de E é definida por uma dupla $e = \{u,v\}$, sendo que $u,v \in V$, e se diz que e_i **incide** em u e v.

Um grafo G pode ser definido como dirigido ou não-dirigido, indicando respectivamente, se existe ou não uma direção entre os dois nós de cada aresta. Um grafo também pode ser definido como um grafo ponderado quando as arestas apresentam valores numéricos.

Neste trabalho, são considerados apenas os grafos sem laços (isto é, arestas ligando um vértice a ele mesmo), sem arestas múltiplas (mais de uma aresta incidindo no mesmo par de nós) e sem direção. Algumas definições para este tipo de grafos são dados a seguir:

Definição 2.1 (Função de pesos) *Para um grafo ponderado não dirigido, uma aresta incidente aos nós u e v, é atribuído um peso segundo uma determinada função $w(u,v)$. Esta função tem as seguintes propriedades: $w(u,v) \geq 0; w(u,v) = w(v,u)$*

Definição 2.2 (Matriz de adjacência) *Uma forma de representar um grafo é mediante a matriz de adjacência W, que é uma matriz simétrica, quadrada e de ordem n, onde n é $|V|$, e cujas entradas são definidas pela Equação 2.1:*

$$W_{ij} = \begin{cases} w(i,j) & \text{se } \{i,j\} \in E; \\ 0 & \text{caso contrario.} \end{cases} \tag{2.1}$$

Definição 2.3 (Grau do nó) *A somatória dos pesos das arestas incidentes em um determinado nó v é conhecido com o grau do nó $deg(i)$, e é definida pela Equação (2.2):*

$$deg(i) = \sum_{j=1}^{n} W_{ij} \tag{2.2}$$

Definição 2.4 (Matriz diagonal de graus) *A matriz diagonal de graus está defina pela Equação 2.3*

$$D_{ij} = \begin{cases} deg(i) & \text{se } i = j; \\ 0 & \text{caso contrario.} \end{cases} \tag{2.3}$$

Definição 2.5 (Volume de um subconjunto de nós) *Sendo $V' \subseteq G$, o volume de V' é dado pela somatória de todos os graus dos nós de V', e é definido mediante a seguinte Equação 2.4:*

$$Vol(V') = \sum_{i \in V'} deg(i) \tag{2.4}$$

2.2.2 Particionamento de Grafos

O particionamento de grafos, ou mais especificamente agrupamento dos nós do grafo, é um problema de agrupamento em grafos (Aggarwal e Wang, 2010). Os **problemas de particionamento de grafos**[1] podem ser classificados em dois grupos: quando se deseja particionar o grafo em dois (bi-particionamento) ou em mais grupos (k-particionamento).

O k-particionamento de grafos pode ser definido formalmente, segundo Fan e Pardalos (2010):

Definição 2.6 (Particionamento de grafos) *Dado $G = (V,E)$ um grafo não dirigido com um conjunto de nós $V = \{v_1, v_2, ..., v_N\}$, e um conjunto de arestas $E = \{e_1, e_2, ..., e_M\}$. O K-particionamento consiste em dividir o conjunto de nós em K subconjuntos disjuntos: $V_1, V_2, ..., V_K$, onde $V_1 \cup V_2 \cup, ..., \cup V_K = V$, e $V_1 \cap (V_2 \cap (...V_{K-1} \cap V_K)...) = \varnothing$. Quando k=2, tem-se a definição de bi-particionamento.*

2.2.3 Critérios de Corte de Grafos

O problema do particionamento de grafos pode ser reescrito da seguinte forma: deseja-se encontrar k-partições (**grupos**)[2] do grafo G, tal que, as arestas entre os diferentes grupos tenham pesos muito baixos (o que significa que nós de diferentes grupos são dissimilares), e as arestas dentro de um mesmo grupo, tenham pesos muito altos (o que significa que os nós dentro do mesmo grupo são semelhantes entre si). O problema do particionamento pode ser abordado por critérios de cortes, que têm como objetivo eliminar arestas de um grafo de forma a produzir K componentes conexas. Geralmente, o objetivo destas formulações é fornecer grupos de nós bem separados e com alta conectividade entre membros do mesmo grupo. A Figura 2.1 ilustra um exemplo de particionamento mediante o corte de grafos.

A seguir são formuladas duas definições de corte de grafos:

Definição 2.7 (Corte de grafos - Bi-particionamento) *Seja o grafo G, W a matriz de adjacência, e $\{A,B\}$ subgrafos de G. O corte de grafos pode ser definido como a somatória dos pesos das arestas que compartilham estes dois subgrafos. A Equação (2.5) define o corte do grafo.*

$$cut(A,B) = \sum_{i \in A, j \in B} W_{ij} \tag{2.5}$$

[1] Usualmente, o término "problema do particionamento de grafos" se refere a encontrar partições, entre todas as possíveis partições de um grafo, que possuem um valor de corte mínimo.

[2] É possível interpretar as partições de um grafo como grupos, posto que a ideia de particionar grafos implica na formação de grupos.

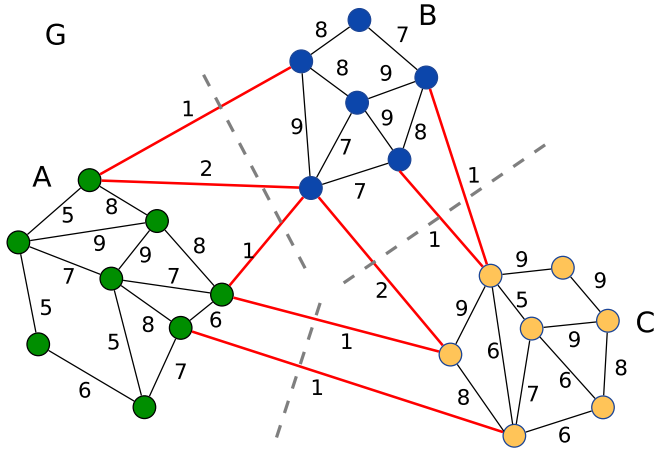


Figura 2.1: Particionamento do grafo G em três grupos: eliminação das arestas com a menor somatória de pesos (arestas vermelhas), e tentativa de agrupar nós que compartilham arestas com pesos altos (arestas pretas).

Definição 2.8 (Corte de grafos - k-particionamento) *Seja o grafo G, W a matriz de adjacência, e $A_1,..,A_K$ subgrafos de G. O corte de grafos pode ser definido como a somatória dos pesos das arestas que compartilham os subgrafos de G. A Equação (2.6) define o corte do grafo.*

$$cut(A_1,...,A_K) = \sum_{i \in A_i, j \in \bar{A}_j} W_{ij} \tag{2.6}$$

Na literatura, os critérios de corte podem ser encontrados como uma proposta ou formulação de um problema. A seguir, são apresentados três tipos de critérios de corte:

Corte Mínimo (MinCut)

MinCut é a primeira formulação de corte que aborda o problema de particionamento. Ele consiste na seleção do conjunto das arestas cuja soma dos pesos seja a menor possível. Tal formulação é apresentada pelas Equações (2.7) e (2.8), para o bi-particionamento e o k-particionamento do grafo, respectivamente.

$$MinCut(A,B) = cut(A,B) \tag{2.7}$$

$$MinCut(A_1,...,A_K) = \sum_{i=1}^{K} cut(A_i,\bar{A}_i) \tag{2.8}$$

Em particular, para k = 2, *MinCut* é um problema relativamente fácil e pode ser resolvido de forma eficiente (Wu e Leahy (1993)). No entanto, na prática, muitas vezes ele não leva a partições satisfatórias. O problema é que, em muitos casos, a solução de *MinCut* simplesmente separa um único nó (nós isolados) dos demais nós do grafo. A Figura 2.2 ilustra esse problema.

Pode-se observar que não é o que se deseja alcançar, pois as partições devem ser uma aglomeração razoavelmente grande de grupos de nós. No caso da segmentação de imagens, deseja-se obter segmentos que sejam mais homogêneos. Porém o *MinCut* não é uma boa solução a ser empregada.

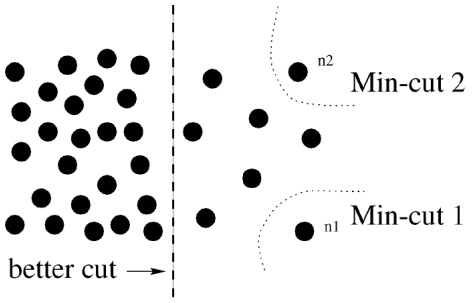


Figura 2.2: Nós isolados: $n1$ e $n2$ são o resultado depois de empregar o critério corte *NimCut*. Adaptado de Shi e Malik (2000).

Corte Proporcional (RatioCut)

A procura de novas formulações, para evitar o problema de partições com nós isolados, levou Hagen e Kahng (1992) a formularem o problema da minimização do corte do raio. Esta formulação resultou da divisão do corte e o número de nós de cada grupo, com o objetivo de balancear os particionamentos. Esta formulação é apresentada pelas Equações (2.9) e (2.10)

$$RatioCut(A,B) = cut(A,B)\left(\frac{1}{|A|}+\frac{1}{|B|}\right) \tag{2.9}$$

$$RatioCut(A_1,...,A_K) = \sum_{i=1}^{K}\frac{cut(A_i,\bar{A}_i)}{|A_i|} \tag{2.10}$$

onde, $|A_i|$ é igual ao número de nós da partição A_i.

Corte Normalizado (NCut)

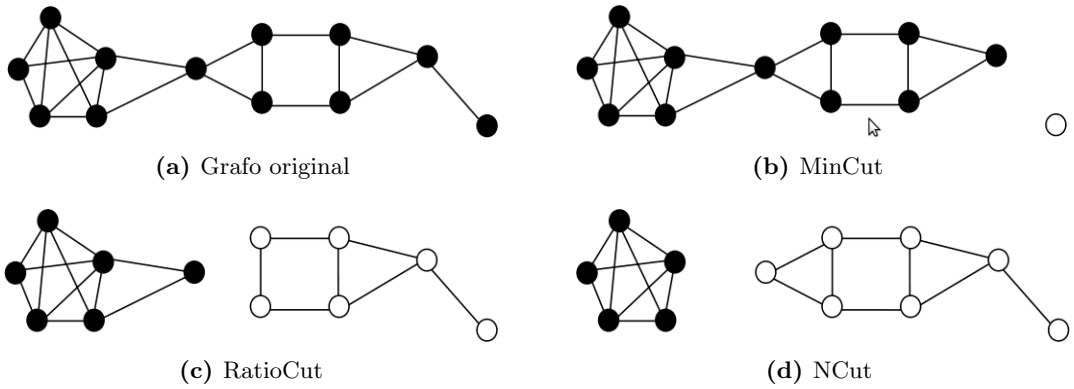
Outro critério de corte que tem sido investigado para o particionamento de grafos, é o **problema de minimização do corte normalizado**. Nesse problema, cada parcela do corte das arestas, $cut(A_i,\bar{A}_i)$, é dividida pela soma dos graus dos nós dentro de cada partição A_i, que é dada por $Vol(A_i) = \sum_{j\in A_i} Dj$. A formulação desse problema é apresentada nas Equações 2.11 e 2.12.

$$NCut(A,B) = cut(A,B)\left(\frac{1}{Vol(A)}+\frac{1}{Vol(B)}\right) \tag{2.11}$$

$$NCut(A_1,...,A_K) = \sum_{i=1}^{K}\frac{cut(A_i,\bar{A}_i)}{Vol(A_i)} \tag{2.12}$$

Esta formulação foi proposta por Shi e Malik (1997; 2000). Os mesmos autores propuseram o uso de **algoritmos espectrais** para resolver esse problema. É por isto que o *NCut* é também conhecido como uma técnica agrupamento espectral (ver a seção 2.3), com aplicação na segmentação de imagens. Qualquer que seja o critério de corte empregado, o k-particionamento é o problema difícil de resolver (Aggarwal e Wang, 2010). Em particular, o critério *NCut* é NP-completo (Shi e Malik, 2000).

A Figura 2.3 ilustra a diferença da aplicação das três formulações de corte de grafos apresentadas. Nela pode-se verificar que a aplicação da formulação do corte normalizado evita a formação de nós isolados.



(a) Grafo original **(b)** MinCut **(c)** RatioCut **(d)** NCut

Figura 2.3: Resultado do bi-particionamento do grafo (a), mediante o corte mínimo (b), proporcional (c) e normalizado (d), com $|V| = 12$. Adaptado de Fan e Pardalos (2010)

2.2.4 Medidas de Dissimilaridade e Similaridade

Medidas de dissimilaridade e similaridade têm uma grande importância porque elas são usadas em múltiplas técnicas de mineração de dados, como é no agrupamento de dados. Geralmente, e por conveniência, o termo de **proximidade** é usado para referir-se à dissimilaridade ou à similaridade que pode existir entre dois objetos[3]. A proximidade entre dois objetos é uma função que calcula a proximidade entre os correspondentes atributos de dois objetos (Tan et al., 2005). Neste contexto serão apresentadas, as definições de dissimilaridade e similaridade, e para ambos casos, são apresentadas algumas métricas que calculam a proximidade entre dois objetos.

Dissimilaridade

A dissimilaridade entre dois objetos é um valor numérico que representa o grau de diferença que existe entre dois objetos x e y, denotada por $d(x,y)$. Enquanto os valores de dissimilaridade forem baixos, existirá uma maior semelhança entre os objetos (serão mais próximos). Caso contrário, para valores altos, os objetos serão muito diferentes. Porém a dissimilaridade apresenta, geralmente, valores na faixa $[0,\infty]$. Também a dissimilaridade, com frequência, é conhecida como

[3]Para o caso do agrupamento em grafos, os objetos representam os nós do grafo.

distância. Para que a distância seja considerada uma métrica esta deve satisfazer as seguintes condições:

- Não negatividade: $d(x,y) \geq 0$
- Identidade: $d(x,y) = 0$, Se $x = y$
- Simetria: $d(x,y) = d(y,x)$
- Desigualdade triangular: $d(x,z) \leq d(x,y) + d(y,z)$

onde x, y, z são vetores de características que representam 3 objetos. A seguir, são apresentadas algumas funções de distância que são comumente utilizadas:

- ***Minkowski*** (Tan et al., 2005):

$$d(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^r \right)^{1/r}, r \geq 1 \tag{2.13}$$

  onde x e y são vetores com n atributos que representam respetivamente 2 objetos. Quando $r = 1$, tem-se a distância *City Block* (também conhecida como *Manhatatan*, *taxicab*, e norma L_1). Quando $r = 2$, tem-se a distância Euclidiana.

- ***Chebyshev*** (Tan et al., 2005):

$$d(x,y) = \max_{i=1}^{n} \{|x_i - y_i|\} \tag{2.14}$$

- ***Mahalanobis*** (Xiang et al., 2008):

$$d(x,y) = \sqrt{(x-y)^T C^{-1} (x-y)} \tag{2.15}$$

  onde C é a matriz de covariância do conjunto de vetores ao qual x e y pertencem.

Similaridade

A similaridade é representada por um valor numérico que expressa o grau de semelhança entre dois objetos. Quando os valores da similaridade são altos, os objetos serão muito semelhantes (próximos). Usualmente a similaridade não apresenta valores negativos. Na maioria dos casos, estes valores estão na faixa $[0,1]$, em que 0 denota objetos não similares e 1 objetos completamente similares. A seguir, são apresentadas algumas funções de similaridade encontradas na literatura.

- **Exponencial** (Rodrigues et al., 2011):

$$s(x,y) = e^{-d(x,y)} \tag{2.16}$$

  onde d é uma função de distância que calcula proximidade entre x e y.

- **Cosseno** (Tan et al., 2005):

$$s(x,y) = \frac{x.y}{||x||\ ||y||} = \frac{\sum_i x_i.y_i}{\sqrt{\sum_i (x_i)^2}\ \sqrt{\sum_i (y_i)^2}} \tag{2.17}$$

- **Fu** (Rodrigues et al., 2011):

$$s(x,y) = 1 - \frac{d_2(x,y)}{||x|| + ||y||} \tag{2.18}$$

  onde d_2 é distância euclidiana.

- **Tanimoto** (Rodrigues et al., 2011):

$$s(x,y) = \frac{x^T y}{||x||^2 + ||y||^2 - x^T y} \tag{2.19}$$

- **Gaussiana** (Shi e Malik, 2000):

$$s(x,y) = e^{\frac{-d(x,y)^2}{\sigma^2}} \tag{2.20}$$

  onde d uma função de distância, σ é um parâmetro de escala (Kunegis et al., 2008). Portando, a similaridade dependerá também, para este caso, do valor de σ. Desta forma a função de similaridade gaussiana é sensível a σ.

2.2.5 Grafo de Similaridade

Um grafo de similaridade tem a forma de um grafo ponderado não direcionado simples $G = (V,E,W)$, sendo W a matriz de adjacência, denominada matriz de similaridade. Os elementos de W representam medidas de similaridade entre os nós do grafo, segundo uma **função de similaridade** $w(i,j)$, tal que $i,j \in V$. A seguir, são apresentadas duas versões da função de similaridade gaussiana:

- Sendo dois vetores x,$y \in \mathbb{R}^d$, a similaridade entre esses vetores é dada pela função de similaridade definida na Equação 2.21.

$$w(x,y) = e^{\frac{-\|x_i - y_i\|_2^{\ 2}}{\sigma^2}} \tag{2.21}$$

  onde σ toma valores entre 10 e 20 porcento de toda a faixa de e distância (Shi e Malik, 2000).

- Com o objetivo de representar uma imagem em um grafo G de similaridade, os nós do grafo são construídos geralmente atribuindo-se os valores dos *pixels* (níveis de cinza, cor, textura) a cada nó do grafo. E os pesos das arestas são construídas de acordo com uma

função de similaridade, segundo a Equação 2.22, proposta por Shi e Malik (2000)

$$w(i,j) = \begin{cases} e^{-\left(\frac{F^2}{\sigma_F{}^2}\right)-\left(\frac{P^2}{\sigma_p{}^2}\right)}, & se\ P < r, \\ 0, & \text{caso contraio} \end{cases} \tag{2.22}$$

onde $F = \|F(i) - F(j)\|_2$ e $P = \|P(i) - P(j)\|_2$ são, respectivamente, a diferença das intensidades e a distância entre os *pixels* i e j. $P(i)$ e $P(j)$ representam a localização espacial dos *pixels*; $\|F(i) - F(j)\|_2$ representa a distância euclidiana entre cada par de *pixels*; r é o valor de tolerância que define a quantidade de vizinhos para cada *pixel*; σ_F e σ_P são parâmetros que tipicamente tomam valores entre 10 e 20 porcento de toda a faixa de intensidade e distância, respectivamente (Shi e Malik, 2000); $F(i)$ é um vetor de características correspondente para cada *pixel*, podendo tomar as seguintes formas:

- $F(i)$, pode representar o valor do *pixel* em níveis de cinza, para segmentar imagens em nível de cinza.
- $F(i) = [v, v{\cdot}s{\cdot}sin(h), v{\cdot}s{\cdot}cos(h)](i)$, onde h, s, v são os valores de HSV, para segmentar imagens pela cor.
- $F(i) = [|I * f_1|, ..., |I * f_n|](i)$, onde f_i são os filtros DooG (*Difference of Offset Gaussian filters*) em varias escalas e orientações, para segmentar imagens pela textura.

2.2.6 Qualidade de agrupamento do grafo

A avaliação de algoritmos de agrupamento em grafos é realizada avaliando a qualidade do agrupamento do resultado que este obteve. Existem varias métricas para avaliar a qualidade do agrupamento do grafo (*Silhouette index*, *modularity*, *Conductance*, etc.). Não obstante, não existe a ideia de uma melhor métrica que avalie a qualidade de agrupamento do grafo. Ela dependerá da forma de como o grafo foi construído e da quantidade de nós que apresenta (Almeida et al., 2011). A seguir, são apresentadas duas das mais conhecidas métricas par avaliar a qualidade de agrupamento do grafo: *Silhouette index* e *Modularity*.

Silhouette index

Esta métrica é baseada em conceitos de coesão e separação. A avaliação da qualidade de agrupamento é dada pelo cálculo de distâncias entre os nós do grafo, o que permite medir a sua similaridade (Tan et al., 2005). A métrica *silhouette index*, para um determinado resultado de agrupamento de um grafo, está definida na Equação 2.23

$$S_{index} = \frac{1}{|V|} \sum_{v \in V} \frac{b_v - a_v}{max\{a_v, b_v\}} \tag{2.23}$$

onde V é o conjunto de nós do grafo, a_v é a distância média entre o nó v e todos os outros nós do mesmo grupo de v, e b_v é a distância média entre o nó v e todos os nós do grupo mais próximo

que não seja o mesmo grupo de v. A *silhouette index* pode assumir valores entre $[-1, 1]$. Ela apresenta valores negativos quando a média das distâncias internas dos grupos são maiores do que a média das distâncias externas. Enquanto os valores da *silhouette* próximos a 1, indicam uma boa qualidade de agrupamento.

Modularity

Em redes complexas, exatamente na identificação de comunidades, a modularidade é uma das mais conhecidas métricas para avaliar a qualidade do agrupamento da rede (grafo). Esta métrica foi proposta por (Newman e Girvan, 2004). Considerando a divisão de uma rede em k comunidades (grupos), é definida uma matriz simétrica e da ordem $k \times k$, onde os elementos e_{ij} estão compostos pela fração de todas as arestas da rede que unem as comunidades i e j. Em relação à matriz e, a modularidade é definida pela seguinte Equação 2.24

$$Q = \sum_{i} (e_{ii} - {a_i}^2) \tag{2.24}$$

onde $a_i = \sum_j e_{ij}$, que é simplesmente a somatória das j linhas ou j colunas da matriz e. A modularidade apresenta valores entre $[0, 1]$. Não obstante, a modularidade pode apresentar valores negativos devido à existência de *singletons*[4]. A modularidade com valores próximos a 1 representam uma boa qualidade de agrupamento. Isso é possível devido à existência de uma grande quantidade de arestas no interior dos grupos.

2.3 Agrupamento Espectral

Um dos primeiros algoritmos de agrupamento espectral foi apresentado no trabalho de Shi e Malik (2000). Nesse trabalho os autores formularam um algoritmo de agrupamento, baseado no particionamento de grafos com aplicação em segmentação de imagens. Esse trabalho foi formulado com base em estudos realizados sobre agrupamento perceptual (Wertheimer, 1938) e a teoria espectral de grafos (Chung, 1997). Posteriormente, foram apresentados novos trabalhos denominados **algoritmos de agrupamento espectral** que se baseiam na teoria espectral de grafos. Portanto, nesta seção é feita uma introdução aos principais conceitos desta teoria.

2.3.1 Conceitos da Teoria Espectral de Grafos

De acordo com Abreu (2005), a teoria espectral de grafos iniciou-se na Química Quântica, por meio de um modelo teórico de moléculas de hidrocarbonetos não saturadas. Tais moléculas possuem ligações químicas com diversos níveis de energia de elétrons. Alguns desses níveis de energia podem ser representados por autovalores de um grafo, o que caracteriza o estudo de teoria espectral. A fundamentação teórica começou em 1957, e se consolidou em 1980 com

[4]Na literatura de agrupamento de dados um *singleton* é o nome para um grupo que possui somente um membro, no caso do agrupamento em grafos, faz referencia a um grupo possui um só nó.

o primeiro livro *"Spectra of Graphs: Theory and Application"*, publicado por Cvetkovic et al. (1980). Desde 1997 vem sendo difundido por Chung.

Chung (1997), tenta definir a teoria espectral de grafos da seguinte maneira: *"Assim como os astrônomos estudam espectros estelares para determinar a composição de estrelas distantes, um dos principais objetivos da teoria espectral de grafos é deduzir as principais propriedades e a estrutura de um grafo a partir do seu espectro (autovalores e autovetores)"*.

Matriz Laplaciana do Grafo

As principais ferramentas do agrupamento espectral são as matrizes laplacianas do grafo e constituem-se em um dos primeiros termos de estudo na teoria espectral de grafos.

Para tais matrizes é considerada a seguinte definição de grafo. Seja $G = (V, E)$ um grafo não dirigido e ponderado, com uma matriz de adjacência W, onde $w_{ij} \geq 0$, e com uma matriz de graus D.

Definição 2.9 (Matriz laplaciana não normalizada do grafo) *Uma matriz laplaciana não normalizada do grafo G é definida pela seguinte Equação:*

$$L = D - W \tag{2.25}$$

onde, $L \in \mathbb{R}^{n \times n}$, $n = |V|$

Definição 2.10 (Matriz laplaciana normalizada do grafo) *Na literatura existem duas definições para uma matriz laplaciana normalizada do grafo.*

$$L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}, \tag{2.26}$$

$$L_{rw} = D^{-1} L = I - D^{-1} W \tag{2.27}$$

onde, L_{sym} é denotada com uma matriz simétrica, I é uma matriz de identidade, e L_{rw} tem relacionamento direto com o *random walk*[5].

2.3.2 Algoritmos de Agrupamento Espectral

No primeiro algoritmo de agrupamento espectral proposto por Shi e Malik (2000), para resolver o problema da busca do corte mínimo normalizado no particionamento de grafos, os autores concluiriam que a solução aproximada para esse problema é obtida computando-se o sistema de autovalor generalizado definido na Equação 2.28.

$$(D - W)x = \lambda D x \tag{2.28}$$

[5] *random walk* em um grafo é um processo estocástico para passar aleatoriamente de um vértice para outro (Von Luxburg, 2007).

onde, λ representa o conjunto de autovalores e x o conjunto de autovetores. Shi e Malik (2000) empregam o segundo menor autovetor com o segundo menor autovalor $x(\lambda_2)$ para bi-particionar o grafo. Também formularam a proposta para obter k-partições do grafo, empregando um conjunto dos p primeiros menores autovetores com os menores autovalores $\{x(\lambda_1),...,x(\lambda_p)\}$.

A seguir, são apresentadas duas formulações de algoritmos de agrupamento espectral encontradas na literatura: bi-particionamento e k-particionamento.

O Algoritmo 2.1 apresenta o algoritmo para o bi-particionamento, onde o grafo de similaridade pode-se referir à representação de uma imagem onde os nós do grafo são os *pixels* da imagem (ver Seção 2.2.5). O detalhamento deste algoritmo é dado a seguir:

1. Computar a matriz laplaciana não normalizada do grafo, segundo a Equação (2.25).
2. Computar o sistema generalizado de autovetores e autovalores, segundo a Equação (2.28), obtendo $x(\lambda_2)$.
3. Particionar o conjunto de nós V utilizando $x(\lambda_2)$

   $x(\lambda_2) = \begin{bmatrix} x_1 \\ . \\ . \\ . \\ x_n \end{bmatrix}$ onde, $n = |V|$, $\forall x_i \in \mathbb{R}$, que podem presentar valores entre $\{-1.0, 1.0\}$. Idealmente para o primeiro grupo entre $\{-1.0, 0.0\}$, e para o segundo entre $\{0.0, 1.0\}$. Desta maneira, o particionamento é realizado, separando os nós em dois grupos.
4. Como o algoritmo é recursivo, um critério de parada deve ser definido: estabelecendo um valor máximo de tolerância do corte (*NCut*). Se o valor do *NCut* supera este valor de tolerância, o algoritmo termina.

Entrada:
$G = (V, E, W, D)$: grafo de similaridade.
Saída:
k : partições de V em $\{V_1,...,V_k\}$, sendo $V_1 \cap (V_2 \cap (...V_{K-1} \cap V_K)...) = \varnothing$.

início
  Calcular $L = D - W$
  Calcular $x(\lambda_2)$ de $(D-W)x = \lambda Dx$
  Utilizar o segundo menor autovetor com o segundo menor autovalor $x(\lambda_2)$, para bi-particionar o grafo.
  Decidir se é preciso reparticionar recursivamente cada uma das duas novas partições. Se for o caso, iniciar novamente o algoritmo para a partição a ser realizada.
fim

Algoritmo 2.1: Algoritmo espectral (Bi-particionamento recursivo) (Shi e Malik, 2000)

O Algoritmo 2.2 apresenta o algoritmo para o k-particionamento de um grafo. A seguir, é fornecida uma descrição mais detalhada deste algoritmo:

1. Computar a matriz laplaciana não normalizada do grafo (Equação (2.25)).

2. Computar o sistema generalizado de vetores e autovalores (Equação (2.28)) obtendo os p primeiros autovetores $x(\lambda_1),...,x(\lambda_p)$

3. A matriz $X \in \mathbb{R}^{n \times p}$ é construída da seguinte forma:

$$X(x(\lambda_1),...,x(\lambda_p)) = \begin{bmatrix} x_{1,1} & . & . & . & x_{1,p} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ x_{n,1} & . & . & . & x_{n,p} \end{bmatrix}$$ onde, cada coluna representa um autovetor, e cada linha um nó.

4. Um algoritmo de agrupamento, como o *k-means*, pode ser empregado para agrupar os dados da matriz X, entendendo que cada linha desta matriz $\{x_{1,1}, x_{1,2}, ..., x_{1,(p-1)}, x_{1,p}\}$ representa um nó.

Entrada:
$G = (V, E, W, D)$: grafo de similaridade; k partições a ser geradas, $k > 2$; p autovetores a serem selecionados.

Saída:
k : partições de V em $\{V_1, ..., V_k\}$, sendo $V_1 \cap (V_2 \cap (...V_{K-1} \cap V_K)...) = \varnothing$.

início
  Calcular $L = D - W$
  Calcular os p autovetores $x(\lambda_1),...,x(\lambda_p)$ de $(D-W)x = \lambda Dx$
  Criar uma matriz $X \in \mathbb{R}^{n \times p}$, contendo os p autovetores nas colunas de X
  Agrupar as filas da matriz X em k grupos, entendendo que as filas desta matriz representam os nós do grafo G
fim

Algoritmo 2.2: Algoritmo espectral (k-particionamento) (Shi e Malik, 2000)

Outras versões de algoritmos podem ser encontrados em Von Luxburg (2007). A maioria dos algoritmos apresentam o mesmo formato. Alguns variam no tipo de matriz laplaciana (não normalizada ou normalizada). Também podemos encontrar algoritmos iterativos, que calculam um certo número de autovetores por iteração (Tolliver e Miller, 2006). No entanto, todos os algoritmos de agrupamento espectral compartilham a tarefa de calcular **autovetores e autovalores** da matriz laplaciana do grafo.

Os algoritmos de agrupamento espectral mostraram produzir resultados com boa qualidade de agrupamento, mas só para um número reduzido de nós. A **desvantagem** é o alto custo computacional, que no pior caso é da ordem $O(n^3)$, onde n é o número de nós do grafo. O alto custo computacional destes algoritmos se deve ao cálculo dos autovetores da **matriz laplaciana do grafo**, na sua **forma normalizada**, que na realidade tem complexidade $O(n^3)$ (Shi e Malik, 2000). Considerando o número de nós necessários para representar uma imagem de tamanho consideravelmente grande, qualquer algoritmo baseado no agrupamento em grafos enfrentaria o mesmo problema.

Por exemplo, se tentássemos segmentar uma imagem de 400 x 300 *pixels*, com 120000 nós no grafo, a matriz de adjacência W precisaria de 14.4 bilhões de entradas que são aproximadamente 53.6 GB de memoria. Na prática isso seria inaplicável para a segmentação de imagens ainda maiores. Por isso, nos trabalhos inicias foram empregadas imagens com tamanho reduzido, como é apresentado em Shi e Malik (2000). Outras propostas são baseadas na formulação de estruturas de grafos, de maneira tal que seja minimizada a cardinalidade do grafo, para dessa forma poder segmentar imagens maiores (Cour et al., 2005; Sun e He, 2009). Nesses trabalhos é proposto um agrupamento de *pixels* empregando uma estrutura hierárquica de grafo similar ao *quadtree*, onde cada nível da hierarquia é uma escala. A Figura 2.4 ilustra esta abordagem. Para a abordagem de segmentação multi-escala da imagem, mais especificamente no trabalho de Cour et al. (2005), os autores concluíram que sua técnica tinha custo computacional linear $O(n)$, onde n é o número de *pixels* da imagem.

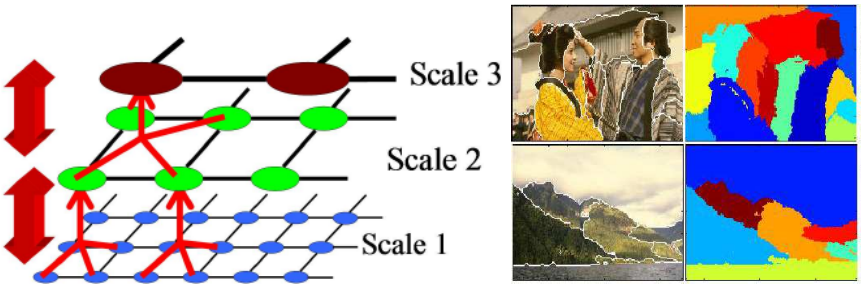


Figura 2.4: Segmentação de imagens multi-escala: (a) cada nó em uma escala representa a união de 4 *pixels*. (b) resultado da segmentação. Adaptado de Cour et al. (2005).

No entanto, a boa qualidade dos resultados de agrupamento dos algoritmos espectrais, faz que fossem propostos trabalhos para melhorar o custo computacional deste tipo de algoritmos. A meta dos trabalhos apresentados na atualidade é a linearidade da complexidade. Há muitos métodos para tentar melhorar a complexidade destes algoritmos, desde técnicas de álgebra lineal como *Lanczos*, ou o *Nystrom* empregado no trabalho de Fowlkes et al. (2004).

Outra abordagem pode ser encontrada em Yan et al. (2009), onde os autores propõem um algoritmo iterativo de rápida aproximação, baseado em *k-means*. Na primeira etapa do Algoritmo 2.3, os autores propõem empregar o algoritmo *k-means*. Com isto, tem-se como resultado y_k centroides dos k grupos, que são as amostras representativas de cada grupo. A seguir, o algoritmo de agrupamento espectral é aplicado nestes y_k centroides, reduzindo assim o custo computacional. A complexidade computacional da etapa 1, *k-means*, é $O(knt)$, onde t é o número de iterações para calcular o *k-means*. A complexidade da etapa 2 é $O(k^3)$ e a complexidade na etapa 3 é $O(n)$. A complexidade computacional total do KASP é $O(k^3)+O(knt)$.

Em Sakai e Imiya (2009), foi proposta uma técnica para agrupar grandes quantidades de dados. Esse trabalho é baseado nas técnicas *random projection* e *random sampling*, que são empregadas para reduzir a **dimensionalidade** e a **cardinalidade** dos dados, respectivamente.

Entrada:
n : dados $x_1,...,x_n$, k número de pontos representativos, K grupos a gerar.
Saída:
k : grupos do conjunto de dados de entrada.

início
  Aplicar *k-means* sobre $x_1,...,x_n$ para:
    Calcular $y_1,...,y_k$ centroides, e considerá-los como amostras representativas.
    Construir um tabela de correspondência para associar cada x_i com o centroide y_j.
  Iniciar o algoritmo de agrupamento espectral sobre $y_1,...,y_k$ para obter k-grupos.
  Recuperar os pontos de x_i correspondentes à tabela de associação com y_j.
fim

Algoritmo 2.3: Fast spectral clustering with k-means (KASP) (Yan et al., 2009)

O objetivo principal destas técnicas é reduzir a matriz de similaridade do grafo, para isto são selecionados aleatoriamente r amostras do conjunto dados projetado $\hat{P}$, e criada uma matriz de similaridade reduzida $W_{\hat{P}\hat{Q}} \in \mathbb{R}^{n\times r}$. A Figura 2.5 mostra este processo. Os resultados são ilustrados na Figura 2.6, em que as r amostras estão marcadas com círculos vermelhos.

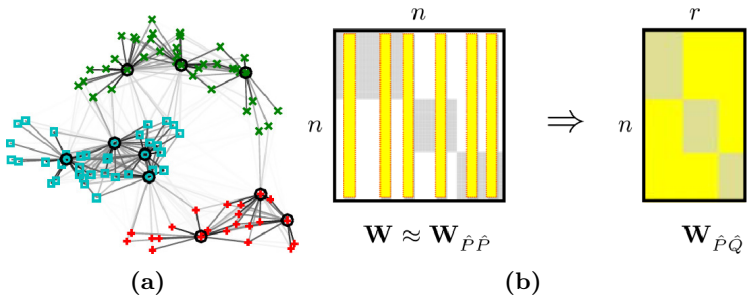


Figura 2.5: Redução da matriz por *random sampling*: (a) r amostras são selecionados aleatoriamente (círculos pretos) (b) é reduzida a matriz de similaridade. Adaptado de Sakai e Imiya (2009).

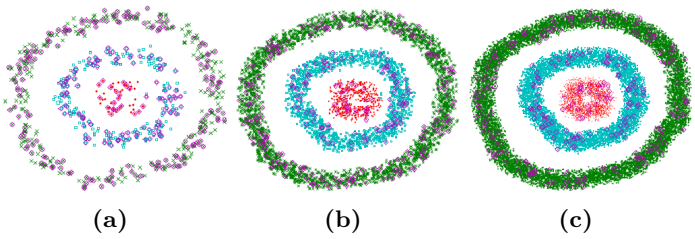


Figura 2.6: Agrupamento de pontos: $r = 300$ (a) $n = 700$ (b) $n = 5000$ (c) $n = 20000$. Adaptado de Sakai e Imiya (2009).

Esta técnica emprega o *Singular Value Decomposition* (SVD) ao invés de *EigenValue Decomposition* (EVD), empregado por outros algoritmos espectrais. O custo computacional chega a ser quase linear, mas isto só para dados de baixa dimensionalidade. A Figura 2.7 ilustra o rendimento desta técnica, em termos de tempo e o tamanho da cardinalidade dos dados.

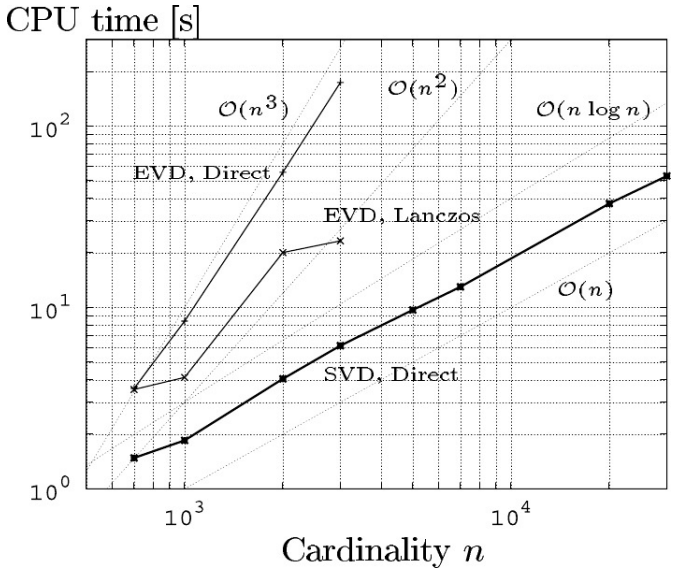


Figura 2.7: Técnicas para calcular autovetores. Adaptado de Sakai e Imiya (2009).

2.4 Algoritmos de Particionamento Multinível

Os algoritmos de particionamento multinível têm sido amplamente usados para encontrar partições de boa qualidade (Hendrickson e Leland, 1995; Karypis e Kumar, 1998; Dhillon et al., 2007; Kong, 2008; Sanders e Schulz, 2010), e têm se mostrado eficientes devido à sua robustez e estabilidade (Nascimento, 2010).

Um algoritmo de particionamento multinível consiste em: dado um grafo $G_0 = (V_0, E_0)$, gerar reduções sucessivas do grafo G_0, até encontrar um grafo $G_x = (V_x, E_x)$, tão pequeno como seja desejável. Seguidamente é realizado o particionamento de G_x em k subgrafos. O particionamento expande-se sucessivamente aos grafos intermediários até que se retorne ao grafo original, obtendo o particionamento de G_0. As três fases (**contração (*coarsening*)**, ***particionamento***, **refinamento ou expansão (*uncoarsening*)**), que geralmente compõem um algoritmo de particionamento multinível, são ilustradas na Figura 2.8, e descritas a seguir.

2.4.1 Fase de contração (coarsening)

Nesta fase, o grafo G_0 é contraído sucessivamente gerando uma sequência de grafos $(G_1, ..., G_i, G_{i+1}, ...G_x)$, de modo que $|V_i| > |V_{i+1}|$. Um conjunto de nós de G_i, representado por V_i^v, é combinado para dar origem a um único nó v no grafo reduzido G_{i+1}, Este agrupamento de nós é denominado **super-nó**.

Para construir um grafo reduzido que seja uma boa representação do grafo original, o peso do nó v é dado pela soma dos pesos de cada nó em V_i^v. Alem disso, para também preservar a conectividade, as arestas incidentes em v representam a união das arestas incidentes nos nós de V_i^v. No caso onde dois ou mais nós de V_i^v sejam adjacentes a um mesmo outro nó u, o peso da

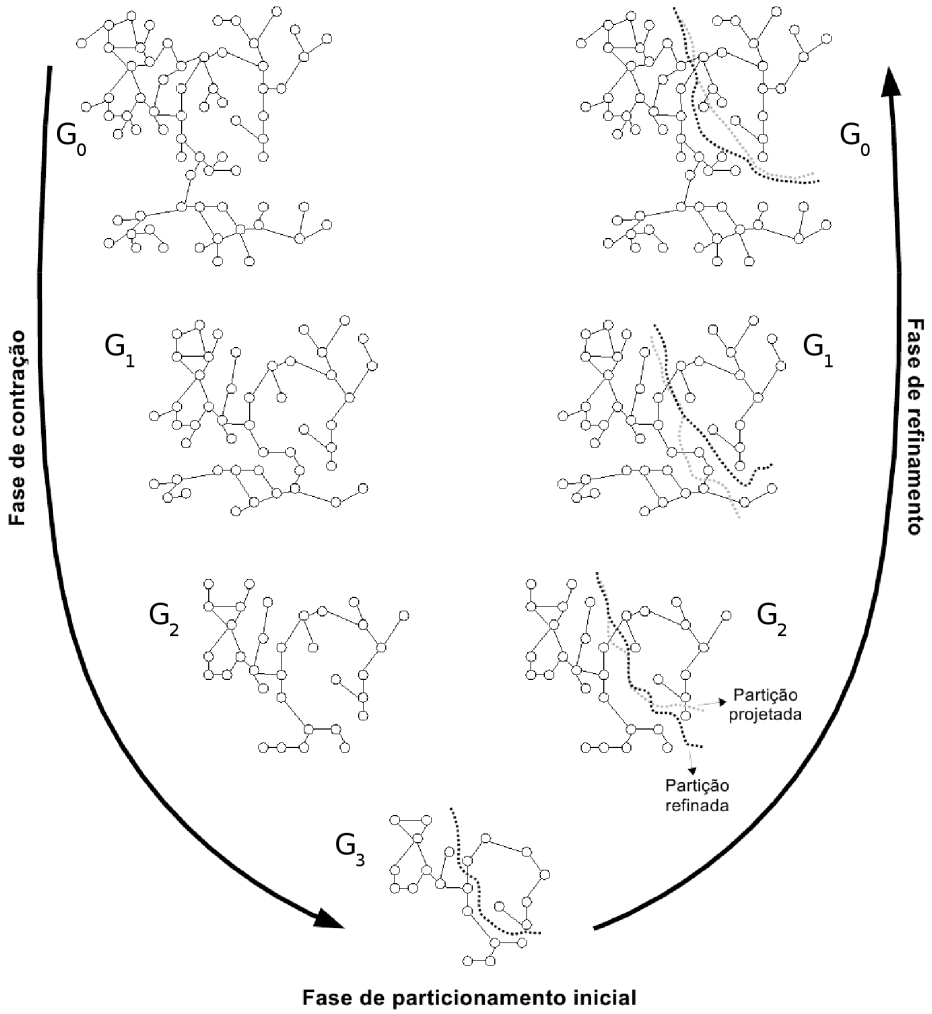
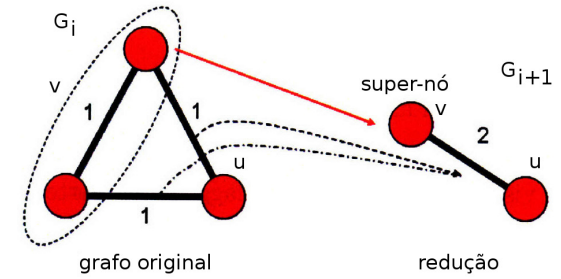


Figura 2.8: Fases do particionado multinível: neste caso bi-particionamento. Adaptado de Nascimento (2010).

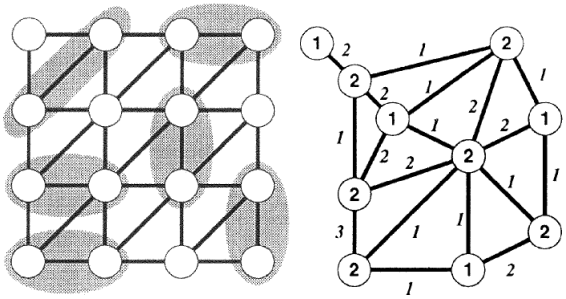
aresta que ligará ao super-nó e ao nó u, no grafo reduzido G_{i+1}, será dado pela soma dos pesos das arestas no grafo de origem G_i, a Figura 2.9**(a)** ilustra esta operação.

A construção do grafo reduzido G_{i+1}, resulta do cálculo do *matching* de G_i. O *matching* é definido como um conjunto de arestas selecionadas de G_i, não havendo duas arestas incidentes no mesmo nó. Os nós de G_i formam super-nós em G_{i+1}, de acordo com a junção de nós obtidos desde o *matching*.

O objetivo de combinar nós e arestas é reduzir o tamanho do grafo G_i em G_{i+1}. Para isso, o *matching* deve conter o maior número de arestas possível. Por isso é ideal obter um *matching* maximal. Na Figura 2.9 é ilustrado um exemplo de redução de um grafo utilizando o conceito de *matching*. Cada um dos seis pares de nós da Figura 2.9**(b)** é combinado para formar um super-nó em 2.9**(c)**. Cada super-nó recebe um peso, e o conjunto de arestas incidentes é dado pela união das arestas incidentes nos vértices que lhe deram origem.



(a)



(b) G_i (c) G_{i+1}

Figura 2.9: Dois exemplos de *matching*: (a) adaptado de (Kong, 2008), (b) e (c) adaptados de (Karypis e Kumar, 1998). Como é ilustrado nas figuras o *matching* é a união de nós e arestas.

Dentre os algoritmos para reduzir o grafo temos: *random matching*, *heavy edge matching* e *light edge matching*. Um breve resumo destes algoritmos é apresentado a seguir:

- ***Ramdom Matching* (RM)**: É um algoritmo para encontrar o conjunto maximal de nós candidatos para a redução do grafo (*matching*). Inicialmente todos os nós do grafo são não-marcados, e a seguir visitados em ordem aleatória. As visitas continuam até que todos os nós tenham sido visitados. Ao visitar um determinado nó v, que ainda esteja não-marcado, o algoritmo busca entre seus vizinhos algum nó u que também esteja não-marcado. Se existir um nó u que satisfaça a condição requerida, o par (u,v) é selecionado para *matching*, e ambos são definidos como marcados. Caso contrario o nó v é definido como marcado, já que todos os seus vizinhos também já estão.

- ***Heavy edge matching* (HEM)** : O *heavy edge matching* utiliza um método aleatório semelhante ao RM. No entanto, utiliza um critério de seleção para buscar entre todos os nós vizinhos não-marcados, um nó cuja aresta possua um maior peso. O objetivo do método LEM é minimizar os pesos totais na geração do grafo reduzido.

- ***Light edge matching* (LEM)**: Utiliza um critério oposto ao HEM para formar o *matching* máximo. Ao invés de buscar uma aresta que possua o maior peso, é buscada a aresta que possua o menor peso. O objetivo do método LEM é maximizar os pesos totais das arestas na geração do grafo reduzido.

2.4.2 Fase de particionamento

Nesta fase é computado o particionamento P_x do grafo $G_x = (V_x, E_x)$, separando V_x em k partes distintas. Na fase de particionamento é possível empregar algoritmos de particionamento de grafos que possuem um alto custo computacional. Isto é possível porque o grafo empregado para o particionamento possui um número reduzido de vértices e de arestas. Ao ser atribuído um super-nó a uma partição, na verdade é atribuído todo o conjunto de nós internos à partição. De essa forma o custo do algoritmo de particionamento não contribui significativamente para o custo total do processo.

2.4.3 Fase de refinamento ou expansão (uncoarsening)

Aqui o resultado da fase de particionamento é projetado sucessivamente a cada grafo intermediário $(G_{x-1}, ..., G_i, ..., G_1)$ até o grafo original G_0. O particionamento encontrado para o grafo reduzido G_x é utilizado para formar o particionamento de G_{x-1} da seguinte maneira: Se um super-nó em G_x está em uma partição i, então todos os vértices que lhe deram origem em G_{x-1} também estarão na mesma partição.

Nesta fase, em cada etapa o novo particionamento, pode ser empregado um algoritmo de refinamento para melhorar a qualidade do particionamento. Esta abordagem permite que, cada vez que os nós recém adicionados ao particionamento, possam ser trocados de partição.

Na literatura pode-se encontrar uma grande quantidade de algoritmos de refinamento. O algoritmo de Kernighan-Lin (KL) (Kernighan e Lin, 1970) foi proposto para refinar bi-partições. Nele são realizadas sucessivas trocas de vértices entre as partições de modo a reduzir o valor de corte, e manter o balanceamento.

Karypis e Kumar (1998) propuseram uma simplificação do algoritmo (KL) utilizando os graus dos nós para o refinamento de k-partições. Este algoritmo permite que os nós de fronteira v (nós que são só adjacentes com os nós de sua partição) possam ser movidos para qualquer outra partição.

Um exemplo de um algoritmo multinível é abordado no trabalho de Dhillon et al. (2007), essa abordagem é conhecida com o nome de GRACLUS (GC), nela é discutida uma equivalência entre a função objetivo do algoritmo *weighted kernel k-means* e a função objetivo do algoritmo de agrupamento espectral. Os autores afirmaram que o GC pode ser aplicável para grandes tarefas de agrupamento de dados, tais como segmentação de imagens, análise de redes sociais, e análise de redes de genes. Os autores testaram o algoritmo em segmentação de imagens, obtendo valores muito baixos do corte do grafo (o que produz um bom particionamento) em comparação aos obtidos pelo algoritmo de agrupamento espectral. No entanto, os autores não fizeram uma análise mais detalhada da segmentação, como é a validação da qualidade de segmentação. A complexidade computacional que apresenta o GC é da ordem $O(n)$, devido às fases de contração e refinamento, e também devido ao *kernel k-means* (Nishida e Nguyen, 2011).

2.5 Identificação de Comunidades

Em redes complexas a definição de uma rede é similar à definição matemática de um grafo. No entanto, o que caracteriza uma rede complexa é o comportamento complexo que apresenta. A presença de complexidade na rede surge porque esta é uma representação de um sistema complexo. Alguns exemplos de redes complexas podem ser: redes informáticas (P2P, *links* entre paginas web, redes sociais), redes biológicas (redes de iteração de proteínas, redes de transição de enfermidades), redes físicas de distribuição de recursos (energia, água, cabeamento telefônico).

A representação de redes complexas está fundamentada sob o conceito de grafos, mas geralmente o grafo em redes complexas é conhecido como rede. Em uma rede, os nós representam indivíduos e as interações entre estes indivíduos são representadas por meio de arestas. Dependendo do grau de interação entre os indivíduos, as comunidades em uma rede são conjuntos de indivíduos que compartilham características similares. O problema de encontrar comunidades entre os indivíduos se reduz também ao fato de encontrar agrupamentos em grafos (Nascimento, 2010).

Uma das mais relevantes caraterísticas das redes complexas é a estrutura de comunidades (também conhecida como agrupamento dos nós da rede). A estrutura de comunidades é a organização dos nós em grupos ou comunidades, onde existe uma maior densidade de arestas entre os nós da mesma comunidade, e uma baixa densidade de arestas entre nós de diferentes comunidades. A Figura 2.10 ilustra um exemplo de estrutura de comunidades.

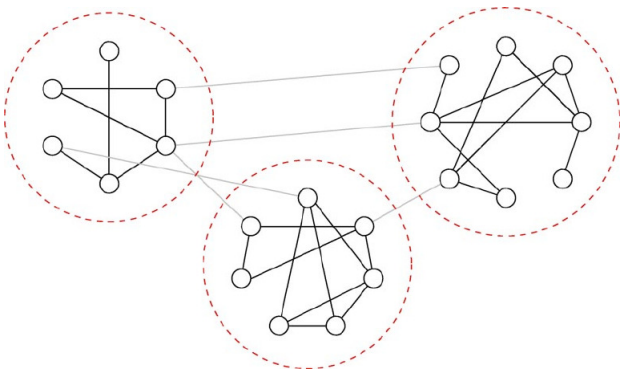


Figura 2.10: Estrutura de comunidades: a pequena rede apresenta para este caso 3 comunidades denotadas por circunferências vermelhas tracejadas, onde as comunidades têm grande densidade de arestas entre os membros de uma mesma comunidade, mas têm baixa densidade de arestas entre comunidades. Adaptado de Newman e Girvan (2004).

Na literatura existem várias propostas para descobrir a estrutura das comunidades de uma rede (Newman e Girvan, 2004; Clauset et al., 2004; Raghavan et al., 2007). Um resumo de propostas para a identificação de comunidades pode ser encontrado em (Santo e Fortunato, 2010). A seguir, são apresentadas duas abordagens para a detecção de comunidades, em que a quantidade de comunidades é calculada de maneira automática, não sendo um parâmetro de entrada.

2.5.1 Fast Greedy (FG)

Clauset et al. (2004) apresentam um algoritmo hierárquico aglomerativo para detectar comunidades de uma rede, cuja complexidade computacional é da ordem $O(md\ log\ n)$, onde d é a profundidade do dendrograma, m é o número de arestas, e n é número de nós. Para redes esparsas com $m \sim n$ e $d \sim log\ n$, a complexidade computacional para este caso é da ordem $O(n\ log^2\ n)$. O *fast greedy*, tem como objetivo obter o máximo valor da modularidade Q, para desta forma determinar uma ótima divisão da rede. O Algoritmo 2.4 apresenta o algoritmo do *fast greedy*, sendo principais passos do algoritmo dados a seguir:

1. O *fast greedy*, assim como qualquer algoritmo hierárquico aglomerativo, inicializa definindo cada nó do grafo como uma comunidade. Em seguida, define uma matriz esparsa simétrica $\triangle Q$ inicialmente da ordem $n \times n$, onde n é número de nós do grafo. Os elementos da matriz $\triangle Q_{ij}$ são atualizados conforme a Equação 2.29. Também é definido um *max-heap* H onde serão armazenados as máximas modularidades entre cada par de comunidades ($\triangle Q_{ij}$).

$$\triangle Q_{ij} = \begin{cases} \dfrac{1}{2m} - \dfrac{k_i k_j}{(2m)^2}, & \text{se } i, j \text{ estão conectados,} \\ 0, & \text{caso contrário} \end{cases} \tag{2.29}$$

   onde m é o número de arestas do grafo, $k_i k_j$ é o número de arestas que compartilham as comunidades i e j. Posteriormente o algoritmo faz a inserção dos máximos valores por cada linha da matriz $\triangle Q$ no *max-heap* H.

2. Neste passo, é criada uma nova k comunidade combinando-se as comunidades i e j presentes no máximo valor de H. Depois são atualizados a matriz $\triangle Q$ e *max-heap* H. A atualização da matriz $\triangle Q$ é realizada da seguinte forma: os membros da comunidade i são incluídas na comunidade j, $(C_i \rightarrow C_j)$, desta forma só é preciso eliminar a linha e coluna de i na matriz $\triangle Q$, os valores da nova comunidade $\triangle Q_{jk}$ são atualizadas aproveitando a linha e coluna de j. Esta atualização segue as seguintes regras:

   (a) Se a nova comunidade k está conectada com i e j, então:

$$\triangle Q_{jk} = \triangle Q_{ik} + \triangle Q_{jk}, \tag{2.30}$$

   (b) Se a comunidade k está conectada com i, mas não com j, então:

$$\triangle Q_{jk} = \triangle Q_{ik} - 2a_j a_k, \tag{2.31}$$

   (c) Se a comunidade k está conectada com j, mas não com i, então:

$$\triangle Q_{jk} = \triangle Q_{jk} - 2a_i a_k, \tag{2.32}$$

onde a_i é definida pela seguinte equação:

$$a_i = \frac{k_i}{2m}, \tag{2.33}$$

as Equações (2.30), (2.31) e (2.32) implicam que só existe um único ponto máximo de Q na execução do algoritmo. Após a ocorrência do maior incremento de $\triangle Q$, todos os demais valores de $\triangle Q$ tendem a decrescer.

3. Enquanto o número de comunidades não chegar a 1, retornar ao passo 2.

Entrada:
$G = (V, E)$: o grafo G, com nós V, com arestas E.
Saída:
$\{V_j, ..., V_k\}$: k grupos de nós.

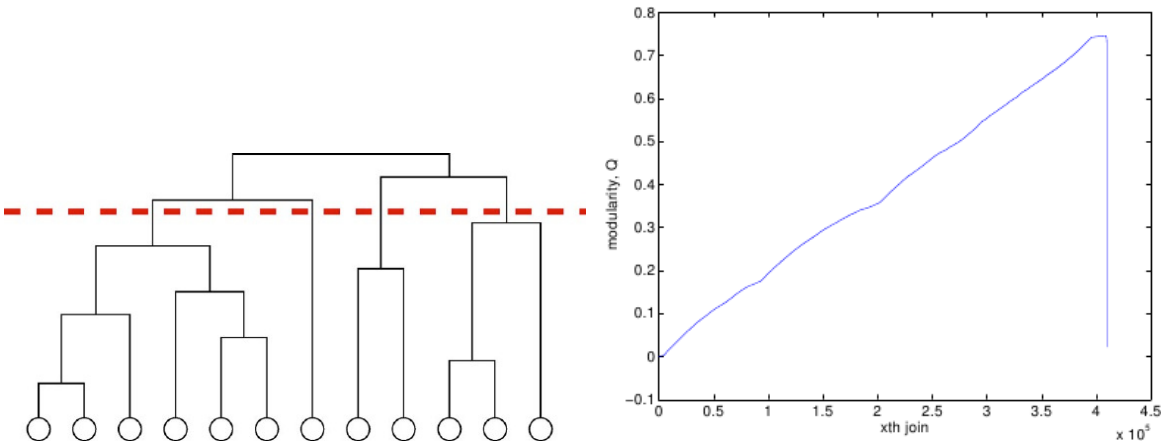
```
1  início
2      def(△Q)                          /* matriz esparsa que salva os valores para △Q_ij */
3      def(max − heap : H)   /* para salvar as modularidades △Q_ij das comunidades i e j */
4      C = {C_i, ..., C_n}              /* cada nó do grafo inicialmente é uma comunidade */
5      init(H)
6      enquanto |C| > 1 faça
7          △Q_ij = max(H)
8          join(C_i, C_j)    /* juntar as comunidades C_i e C_j, então V'_j = V_j ∪ V_i, |C| = |C| − 1 */
9          update(△Q)          /* atualizar a matriz △Q em torno á linha e coluna j */
10         update(H)                                   /* atualizar o max-heap H */
11         se decrease(△Q_ij^(t), △Q_ij^(t−1)) então
12             break
13         fim
14     fim
15 fim
```

Algoritmo 2.4: *Fast Greedy*

Uma possível ilustração do algoritmo *fast greedy*, é apresentado na Figura 2.11: a Figura 2.11**(a)** apresenta o resultado do agrupamento do algoritmo. Os círculos na parte inferior do gráfico representam os nós, que inicialmente são considerados comunidades. Ao longo da execução as comunidades são unidas formando novas comunidades. Isto é feito até chegar a formar uma única comunidade. A linha vermelha representa o corte do dendrograma, onde é utilizado o valor máximo do $\triangle Q$ para determinar a divisão ótima de comunidades. A Figura 2.11**(b)** ilustra o incremento da modularidade $\triangle Q$, onde ela tem um único valor máximo. Neste ponto, o algoritmo alcança o seu objetivo, que é a maximização da modularidade para obter uma boa divisão da rede.

2.5.2 Label Propagation (LP)

Raghavan et al. (2007) apresentam o *label propagation*, um algoritmo iterativo para detectar comunidades em uma rede. Esta abordagem apresenta complexidade computacional quase linear $\sim O(n)$. A principal ideia do *label propagation* é a seguinte: supondo que um nó x tem como



(a) Dendrograma **(b)** $\triangle Q$

Figura 2.11: (a) Representação do dendrograma, para o algoritmo *fast greedy*, a linha vermelha representa o corte do dendrograma empregando o máximo valor de $\triangle Q$. Adaptado de Newman e Girvan (2004). (b) No eixo x é mostrado o número de uniões realizadas entre duas comunidades ao longo do algoritmo, nela é mostrada o acrescentamento de $\triangle Q$, onde $\triangle Q$ apresenta um único valor máximo. Adaptado de Clauset et al. (2004).

vizinhos: $x_1, x_2, ..., x_k$ e cada vizinho possui uma etiqueta indicando a comunidade à que pertence, então a comunidade do nó x é determinada com base nas comunidades dos seus vizinhos (é tomada a etiqueta que tem maior frequência nos vizinhos de x). A Figura 2.12 ilustra a atualização de etiquetas em um subconjunto de nós. No *label propagation*, a divisão da rede está baseada na propagação das etiquetas dos nós de maneira iterativa, onde a atualização das etiquetas pode assumir duas formas: **síncrona** e **assíncrona**. Para a forma síncrona a atualização da etiqueta do nó x em uma iteração t, está definida em função das etiquetas dos seus vizinhos em uma iteração $t-1$. A atualização de etiquetas da forma síncrona é definida pela Equação (2.34),

$$C_x(t) = f(C_{x_1}(t-1), ..., C_{x_k}(t-1)). \tag{2.34}$$

No caso da forma assíncrona, a etiqueta do nó x em uma iteração t é atualizada segundo a Equação (2.35),

$$C_x(t) = f(C_{x_{i1}}(t), ..., C_{x_{im}}(t), C_{x_{i(m+1)}}(t-1), ..., C_{x_{ik}}(t-1)), \tag{2.35}$$

onde $x_{i1}, ..., x_{im}$ são vizinhos de x cujas etiquetas são atualizadas na iteração t atual, enquanto $x_{i1}, ..., x_{im}$ são vizinhos de x cujas etiquetas não são atualizadas na iteração t atual. Nas Equações (2.34) e (2.35) a função f determina a etiqueta com maior frequência dos vizinhos de x, que será atribuída ao nó x. O Algoritmo 2.5 apresenta o algoritmo, cujos principais passos são descritos a seguir:

1. O algoritmo *label propagation* é inicializado atribuindo a cada nó uma única etiqueta ($t = 0$).
2. Iterativamente para $t = t + 1$, todos os nós são dispostos em uma ordem aleatória $X = random(V)$, onde X é um conjunto que contém os nós em uma ordem aleatória. Então

a etiqueta de cada nó $x \in X$, é atualizada de forma assíncrona (Equação (2.35)). São selecionados m vizinhos a serem atualizados na iteração t atual, e também são selecionados $k-m$ vizinhos cujas etiquetas foram atualizadas em uma interação anterior $t-1$.

3. Em cada iteração é avaliado um critério de parada. Neste critério são avaliadas todas as atualizações das etiquetas dos nós. Segundo a Equação (2.35), se todas as etiquetas dos nós, anterior à atualização, possuem a mesma etiqueta que retorna o f, então o algoritmo terminará.

Entrada:
$G = (V,E)$: o grafo G, com nós V, com arestas E.
Saída:
$\{V_j,...,V_k\}$: k grupos de nós.

```
início
    i = 0
    para cada x ∈ V faça
        C_x(0) = i
        i = i + 1
    fim
    enquanto true faça
        t = t + 1
        X = random(V)
        isOut = true
        para cada x ∈ X faça
            C_x(t) = f(C_{x_i1}(t),...,C_{x_im}(t),C_{x_i(m+1)}(t−1),...,C_{x_ik}(t−1))
            se {C_x(t)! = C_x(t−1)} então
                isOut = false
            fim
        fim
        se (isOut) então
            break
        fim
    fim
fim
```

Algoritmo 2.5: *Label propagation*

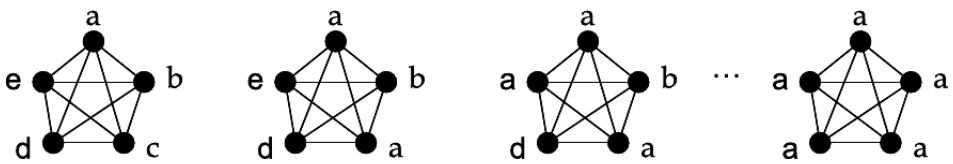


Figura 2.12: Atualização de etiquetas no *label propagation*: na figura de esquerda a direita, os nós são atualizados um a um. Neste caso exite uma grande densidade de arestas, isto permite que todos os nós adquiram a mesma etiqueta. Adaptado de Raghavan et al. (2007).

2.6 Considerações Finais

O presente capítulo apresentou as revisões bibliográficas de 3 tipos de algoritmos de agrupamento em grafos. Foi apresentada a abordagem dos algoritmos de agrupamento espectral, mostrando bons resultados, com respeito à qualidade do agrupamento. No entanto, algoritmos desta categoria têm custo computacional $O(n^3)$ devido ao cálculo dos autovetores e autovalores da matriz laplaciana do grafo. Em contraste, outros trabalhos que tentam reduzir este custo computacional, têm surgido (Fowlkes et al., 2004; Yan et al., 2009; Sakai e Imiya, 2009).

Também foi apresentada uma revisão de algoritmos de particionamento multinível, que apresentam heurísticas para reduzir o grafo, baseado na formação de super-nós, mediante a união dos nós e as arestas. O resultado do particionamento depende da utilização do tipo de heurística para reduzir o grafo, como também depende do tipo de algoritmo que faz o particionamento do grafo.

Foram apresentados dois algoritmos baseados na detecção de comunidades: o *fast greedy* e o *label propagation*. Neste tipo de algoritmos o agrupamento dos nós está baseado na descoberta da estrutura das comunidades, caraterizadas pela alta densidade de arestas entre os membros de uma mesma comunidade, e com baixa densidade de arestas entre comunidades.

A principal diferença entre algoritmos de particionamento (agrupamento espectral, multinível) e algoritmos para identificação de comunidades (*fast greedy*, *label propagation*), é que no primeiro grupo busca-se o corte mínimo das arestas (minimização do corte), enquanto, no segundo, procura-se uma maior densidade de arestas no interior dos grupos (em alguns casos empregando a maximização da modularidade).

Finalmente na Tabela 2.1 são resumidas as principais características dos algoritmos abordados neste capítulo. Para os algoritmos de agrupamento espectral existem duas formas de se criar o grafo: emprega uma matriz de adjacência W simétrica, ou uma matriz de adjacência não simétrica. Nesta ultima são empregados métodos *sampling*, como foi visto em (Sakai e Imiya, 2009). Isto possibilita a diminuição da complexidade computacional. No entanto, isto também poderia significar uma perda de informação no grafo.

Tabela 2.1: Tabela de comparação entre os 4 tipos de algoritmos de agrupamento em grafos: agrupamento espectral (SC), *graclus* (GC), *fast greedy* (FG) e *label propagation* (LP). O k representa o número de partições do grafo, o sigma o parâmetro propô do SC no cálculo da similaridade.

Algoritmo	Complexidade	Abordagem	Parâmetros
SC	$W\ simtrica:\ O(n^3)$ $W\ nosimtrica:\ O(n^2), \sim O(n)$	teoria espectral de grafos	k,σ
GC	$O(n)$	multinível, *kernel k-means*	k
FG	$O(md\ log\ n)$	$max(\triangle Q)$	–
LP	$\sim O(n)$	propagação de etiquetas	–

Observando os custos computacionais dos algoritmos apresentados neste capítulo, nota-se que aqueles que identificam comunidades mostram-se os mais adequados para aplicação à segmentação de imagens de grandes dimensões. Assim como também, os algoritmos de particionamento multinível, onde o grafo é reduzido e posteriormente particionado. Neste trabalho mostraremos como aplicar estes 4 tipos de algoritmos para segmentar imagens avaliando seus desempenhos e comparando com outras propostas clássicas. Maiores detalhes são dados nos capítulos 4 e 5.

Capítulo

3

Algoritmos de *Superpixels*

3.1 Considerações Iniciais

Os métodos de pre-processamento conhecidos como *superpixels* são uma proposta relativamente recente (Moore et al., 2008; Levinshtein et al., 2009; Cigla e Alatan, 2010; Zeng et al., 2011; Achanta et al., 2012), sendo que as principais abordagens focaram na qualidade[1] dos *superpixels* e na complexidade computacional. A grande importância das técnicas de *superpixel* está associada à redução do custo computacional que estas conferem às varias tarefas de analises de imagens. Além da **segmentação** de imagens, estas tarefas podem ser: **reconhecimento** (Kaufhold et al., 2006), **rastreamento** (Rasmussen, 2007) e **localização** de objetos (Fulkerson et al., 2009). Na segmentação de imagens baseada no agrupamento em grafos, o custo computacional, ao se representar regiões (*superpixels*) da imagem como os nós do grafo, ao invés de usar os *pixels* da imagem individualmente, permite uma redução considerável na quantidade dos nós do grafo (de milhões para milhares). Uma outra importância dos *superpixels* aplicados em segmentação de imagens é que ela possibilita uma melhora na qualidade dos resultados da segmentação (Achanta et al., 2012).

Recentemente, Achanta et al. (2012) realizaram um estudo do estado de arte de técnicas de *superpixels*. Também, os autores fizeram uma classificação das técnicas de *superpixel*, categorizando em: técnicas que empregam algoritmos **baseados em grafos** (Ren e Malik, 2003; Felzenszwalb e Huttenlocher, 2004; Moore et al., 2008; Veksler et al., 2010), e técnicas baseadas em métodos de **gradiente ascendente** (Vincent e Soille, 1991; Comaniciu e Meer, 2002; Vedaldi e Soatto, 2008; Levinshtein et al., 2009; Cigla e Alatan, 2010; Achanta et al., 2012). Nas técnicas

[1] A qualidade dos *superpixels* é denotada por a uniformidade ou homogeneidade da forma (convexidade) que eles podem apresentar, e principalmente, pela capacidade dos *superpixels* atingir as bordas da imagem.

baseadas em grafos, os nós do grafo compõem os *pixels* da imagem, e as arestas representam pesos que denotam a similaridade entre cada par de *pixels*, segundo um critério de vizindade. Os *superpixels* são criados pela minimização do custo de uma função definida sobre o grafo. Por outro lado, nas técnicas que empregam métodos de gradiente ascendente, os *superpixels* são obtidos a partir de um agrupamento inicial de *pixels*, onde iterativamente estes agrupamentos são refinados até atingir algum critério de convergência.

Neste capítulo são apresentados 2 principais algoritmos de *superpixel* baseados em métodos de gradiente ascendente: o *Speeded-Up Turbo Pixels* (SUTP) e o *Simple Linear Iterative Clustering* (SLIC). A caraterística comum a ambos é a complexidade computacional, que é da ordem $O(n)$. Há também outro algoritmo, *TurboPixel* (TP), proposto por Levinshtein et al. (2009), baseado em técnicas de fluxos geométricos, onde progressivamente um conjunto de *pixels* é dilatado para obter os *superpixels*. Embora esta técnica produza *superpixels* quase uniformes[2] e seja da ordem de complexidade é quase linear $\sim O(n)$, onde n é o número de *pixels* da imagem, ela não é na pratica, uma técnica muito rápida, como foi observado em (Cigla e Alatan, 2010; Achanta et al., 2012). Com relação à qualidade dos *superpixels*, o TP não precisamente apresenta bons resultados, como foi comprovado no trabalho de Zeng et al. (2011).

3.2 Speeded-Up Turbo Pixels (SUTP)

No trabalho de Cigla e Alatan (2010) é proposto um algoritmo mais eficiente que a técnica *TurboPixels*, denominado ***Speeded-Up TurboPixels***, tanto no que diz respeito à complexidade (ordem $O(n)$) à geração de *superpixels* mais uniformes. O método utiliza o algoritmo *k-means* para gerar k *superpixels* em um processo de refinamento visando a convexidade dos *superpixels*. O Algoritmo 3.1 ilustra o algoritmo SUTP, enquanto os items 1, 2 e 3 apresentam os principais passos executados pelo STUP:

1. Inicialmente uma imagem é dividida em k regiões retangulares como é mostrada na Figura 3.1**(a)**. Portanto, inicialmente, no inicio cada *superpixel* tem uma forma retangular de dimensões[3] SxS, onde $S = \sqrt{\frac{N}{k}}$, e N é igual ao número de *pixels* da imagem. Desta forma os *superpixels* possuem centróides espalhados nestas regiões retangulares.

2. Iterativamente, os *pixels* que estão posicionados sobre o contorno dos segmentos são intercambiados para novos segmentos, minimizando a função de custo da Equação 3.1.

$$C_{x,y}{}^{i} = \lambda_1 |I(x,y) - I_i| + \lambda_2 |(x - C_x{}^{i})^2 + (y - C_y{}^{i})^2| \tag{3.1}$$

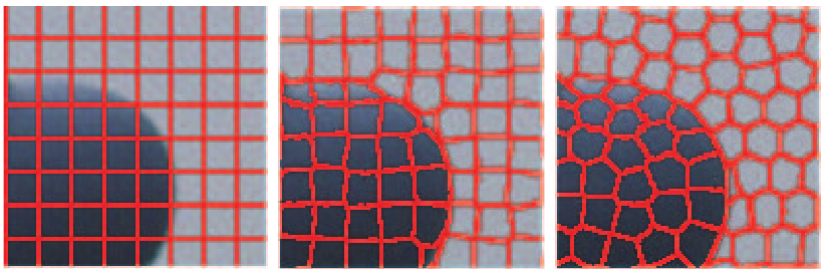
   onde, I_i indica a média da intensidade do i^{th} segmento, x e y são as posições dos *pixels* testados entre os diferentes segmentos. $C_x{}^{i}$ e $C_y{}^{i}$ são as posições do centróide do i^{th} segmento; λ_1 e λ_2 correspondem, respectivamente, às ponderações de similaridade e à restrição de

[2] A uniformidade dos *superpixels* é referida à forma comum que eles podem apresentar.
[3] Nesta dissertação as dimensões inicias SxS dos *superpixels* também sera denominado como o tamanho dos *superpixels*

convexidade. O primeiro termo garante similaridade dos *pixels* coloridos a serem incorporadas, enquanto o segundo termo permite aos *superpixels* possuírem formas convexas, impedindo que *pixels* distantes sejam mesclados. As convexidades dos segmentos podem ser modificadas por uma variação no λ_2. Portanto, λ_1 e λ_2 têm uma faixa de valores entre $[0,1]$.

3. Um possível critério de parada do algoritmo é estabelecer, para cada iteração, um número máximo de *pixels* a ser intercambiados. Se o número de *pixels* intercambiados está abaixo do estabelecido, o passo da geração de *superpixel* será terminado. Mas, geralmente, o critério de parada é feito fixando-se o número de iterações.

A Figura 3.1 ilustra os passos para gerar *superpixels* com a abordagem STUP.



(a) **(b)** **(c)**

Figura 3.1: Processo para gerar *superpixels* com a abordagem STUP. *Pixels* no contorno do segmento (vermelho) são atualizadas a cada iteração. (a) Inicialização, (b) passo intermediário e (c) passo final. Adaptado de Cigla e Alatan (2010).

3.3 Simple Linear Iterative Clustering (SLIC)

No trabalho de Achanta et al. (2010, 2012) é proposto um algoritmo iterativo para gerar *superpixels* de boa qualidade em imagens coloridas, empregando o espaço de cor CILAB. A complexidade computacional é da ordem $O(n)$. Também esta abordagem está baseada no algoritmo de agrupamento *k-means*. No entanto, em contraste ao *k-means* tradicional, o SLIC apresenta uma redução do espaço de busca, como é ilustrado na Figura 3.2. O Algoritmo 3.2 apresenta o algoritmo SLIC, cujos principais passos são dados a seguir:

1. Tendo como entrada uma imagem, é realizado um particionamento da imagem em k regiões retangulares, onde cada região compõe um *superpixel* inicial de dimensões SxS, onde $S = \sqrt{\frac{N}{k}}$, e o N é o número de *pixels* da imagem. A seguir é atribuído para cada j^{th} *superpixel* um vetor composto por 5 dimensões $C_j = |l_j, a_j, b_j, x_j, y_j|^T$, onde l_j, a_j, b_j corresponde à média da cor dos *pixels* que compõem cada *superpixel*, tanto x_j quanto y_j correspondem à média das localizações dos *pixels* em cada *superpixel*, que na realidade é o centróide $C(x_j, y_j)$ do j^{th} *superpixel*.

Entrada:
I : a imagem, N a quantidade de pixel da imagem.
nI : número de iterações.
k : número de *superpixels*.
λ_1 : variável de ponderação para a similaridade entre as intensidades dos *pixels* e *superpixels*.
λ_2 : variável de ponderação para a restrição da convexidade do *superpixel*.
Saída:
$l(i)$: i *pixels* agrupados em k *superpixels*

início
  $S = \sqrt{\frac{N}{k}}$ `/* calcular o comprimento aproximado do lado dos superpixels */`
  $grid(I,S)$ `/* particionar a imagem em regiões retangulares */`
  $l(i) = k$ `/* agrupar os i pixels com os respectivos k superpixels */`
  $C_k = [I_k, x_k, y_k]$ `/* inicializar os k centroides */`
  enquanto *o nI tenha sido completado* **faça**
    $d(i) = \infty$ `/* inicializar as distâncias para cada pixel i */`
    para cada *superpixel k* **faça**
      para cada *pixel i que está sobre o contorno do superpixel k* **faça**
        $D = \lambda_1 |I(i) - I_k| + \lambda_2 |(x_i - x_k)^2 + (y_i - y_k)^2|$
        se $D < d(i)$ **então**
          $d(i) = D$ `/* atualizar a distância mínima entre` C_k `e i */`
          $l(i) = k$ `/* agrupar o pixel i no superpixel k */`
        fim
      fim
    fim
    atualizarCentroides() `/* atualizar os k centroides */`
  fim
fim

Algoritmo 3.1: *Speed Turbo Superpixel*

2. Tendo em consideração um determinado número de iterações (os autores propõem 10 iterações para obter *superpixels* de ótima qualidade), são percorridos os centróides dos k *superpixels* e é realizado o intercâmbio dos *pixels* localizados na região $2S$x$2S$ em torno do centróide $C(x_j, y_j)$ do j^{th} *superpixel* (ver a Figura 3.2**(b)**). Este intercâmbio de *pixels* é realizado calculando a minimização de D na Equação 3.2. Esta minimização ocorre em função do cálculo de duas distâncias: d_c(no espaço da cor) e d_s (na localização espacial).

$$\begin{aligned} d_c &= \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2} \\ d_s &= \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \\ D &= \sqrt{d_c^2 + (\frac{d_s}{S})^2 m^2} \end{aligned} \tag{3.2}$$

onde j representa ao *superpixel* atual, e o i representa aos *pixels* que estão localizados na área $2S$x$2S$ em torno do centróide $C(x_j, y_j)$ do j^{th} *superpixel*. Como d_c emprega o espaço de cor CILAB, m pode tomar uma faixa de valores entre $[1, 40]$ (Achanta et al., 2012, 2010).

Enquanto existir uma minimização em D, um intercambio de i *pixels* é realizado entre os k diferentes *superpixels*.

3. Finalmente, após as iterações, existem alguns *pixels* que não correspondem a uma mesma componente conexa[4]. Estes *pixels* são reagrupados em uma operação de pós-processamento.

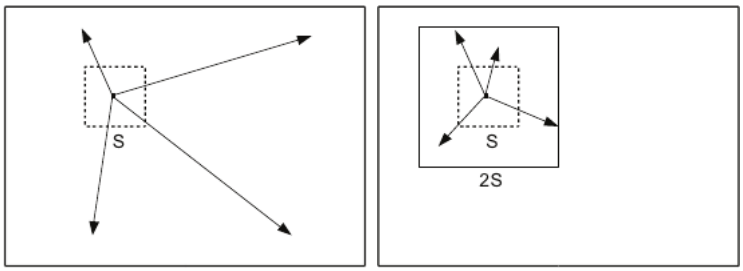


Figura 3.2: Redução do espaço da busca na imagem para o refinamento dos *superpixels*: (a) espaço de busca empregado por um algoritmo *k-means* convencional, (b) espaço de busca reduzido utilizado por o SLIC representado por a região $2S$x$2S$. A complexidade do SLIC é linear sobre o número de *pixels* da imagem $O(n)$. No entanto, a complexidade para um algoritmo *k-means* convencional é de $O(knI)$, onde I é o número de iterações. Adaptado de Achanta et al. (2012).

3.4 Considerações Finais

Neste capitulo foi apresentada a revisão bibliográfica de técnicas de *superpixel*. Foram abordadas 3 principais técnicas baseadas em métodos de gradiente ascendente. As 3 técnicas possuem complexidade computacional linear $O(n)$ sobre o número de *pixels* da imagem.

De acordo com a revisão bibliográfica, o **TP** (Levinshtein et al., 2009), não é a melhor alternativa para extrair *superpixels* em imagens de grandes dimensões. Além disso a qualidade dos *superpixels* gerados é outra desvantagem que apresenta, se comparado aos demais métodos.

Por outro lado, o **SUTP** (Cigla e Alatan, 2010) é uma boa alternativa para a extração de *superpixels* em imagens de grandes dimensões. No entanto, como o intercâmbio de *pixels* entre os *superpixels* é limitado somente para os *pixels* que estão localizados sobre os contornos dos *superpixels*, é preciso um maior número de iterações para o algoritmo convergir. Somente assim é possível que os contornos dos *superpixels* possam atingir às bordas da imagem. O STUP não possui muitos parâmetros que necessitam ajustes além do k, λ_1 pode permanecer constante com valor igual a $1,0$. No entanto, λ_2 é diretamente proporcional à convexidade dos *superpixels*, isto é, enquanto existirem valores de λ_2 próximos a $1,0$, os *superpixels* apresentarão maior convexidade. Na prática, em *superpixels* que não possuem uma maior convexidade, os contornos dos *superpixels* possuem maior probabilidade de atingir as bordas da imagem.

[4]Chame-se componente conexa ao conjunto de *pixels* que pertencem a um mesmo *superpixel*, onde todos os *pixels* sejam conexos entre si.

Entrada:
I : a imagem, N a quantidade de pixel da imagem.
nI : número de iterações.
k : número de *superpixels*.
m : constante que garante a qualidade dos *superpixels*.
Saída:
$l(i)$: i *pixels* agrupados em k *superpixels*

início
  $S = \sqrt{\frac{N}{k}}$ /* calcular o comprimento aproximado do lado dos *superpixels* */
  $grid(I,S)$ /* particionar a imagem em regiões retangulares */
  $l(i) = -1$ /* para cada pixel i */
  $C_k = [l_k, a_k, b_k, x_k, y_k]$ /* inicializar os k centroides */
  enquanto *o nI tenha sido completado* **faça**
    $d(i) = \infty$ /* inicializar as distâncias para cada pixel i */
    para cada *centroide* C_k **faça**
      para cada *pixel i da região* $2Sx2S$ *em torno ao* C_k **faça**
        $d_c = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2}$
        $d_s = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}$
        $D = \sqrt{d_c^2 + (\frac{d_s}{S})^2 m^2}$ /* calcular distância D entre C_k e i */
        se $D < d(i)$ **então**
          $d(i) = D$ /* atualizar a distância mínima entre C_k e i */
          $l(i) = k$ /* agrupar o *pixel* i no *superpixel* k */
        fim
      fim
    fim
    atualizarCentroides() /* atualizar os k centroides */
  fim
  corrigirError()
fim

Algoritmo 3.2: *SLIC Superxpixel*

Também o **SLIC** (Achanta et al., 2012) é aplicável para imagens de grandes dimensões. Nesta abordagem não é necessário um número elevado de iterações para que o algoritmo chegue a convergir (os autores fixaram em 10 o número de iterações, obtendo uma boa qualidade nos *superpixels*). Nesta abordagem o tamanho dos *superpixels* não sofre influência do número de iterações, porque o intercâmbio de *pixels* entre os *superpixels* é realizado com todos os *pixels* que estejam na região $2S$x$2S$ em torno dos centróides dos *superpixels*. Desta forma, os contornos dos *superpixels* alcançam rapidamente as bordas da imagem sem precisar de muitas iterações, como é ilustrado nas Figuras 3.3(l-u). O SLIC não possui muitos parâmetros, além de k. O parâmetro m influência diretamente a convexidade dos *superpixels*. Para valores de m próximos a 0.0, os *superpixels* não serão convexos, podendo assumir formas irregulares, ao mesmo tempo os contornos dos *superpixels* têm maiores probabilidades de atingir as bordas da imagem.

As duas técnicas STUP e SLIC são muito rápidas em comparação ao TP. Isto se deve a suas formulações simples baseadas em métodos de gradiente ascendente e conexão com o algoritmo de agrupamento *k-means*. O SUTP e o SLIC são técnicas muito parecidas, no entanto, a maior diferença entre elas é o procedimento para intercambiar *pixels* no processo do refinamento dos *superpixels*. Quando o tamanho dos *superpixels* é maior, o STUP precisa de um maior número

de iterações para atingir as bordas da imagem. No caso do SLIC, o tamanho dos *superpixels* é indiferente ao número de iterações, porque converge rapidamente às bordas da imagem. Desta forma, o SLIC torna-se a melhor alternativa para imagens coloridas de grandes dimensões. Portanto, a técnica SLIC será utilizada na extração de *superpixels*. A Tabela 3.1 apresenta as principais características destas 3 técnicas. A Figura 3.3 apresenta uma comparação entre as técnicas STUP e SLIC.

Tabela 3.1: Tabela de comparação entre 3 técnicas que geram *superpixels*: o parâmetro k representa o número de *superpixels* a serem gerados. λ_1, e λ_2 são parâmetros de ponderação para o cálculo da distância no intercâmbio de *pixels* no STUP. m é o parâmetro que influência no cálculo da distância espacial entre os *pixels* a serem intercambiados e os centróides dos *superpixels* no SLIC.

Algoritmo	Complexidade	Abordagem	Parâmetros
TP	$\sim O(n)$	*Geometric flows*	k
SUTP	$O(n)$	*k-means*	k, λ_1, λ_2
SLIC	$O(n)$	*k-means*	k, m

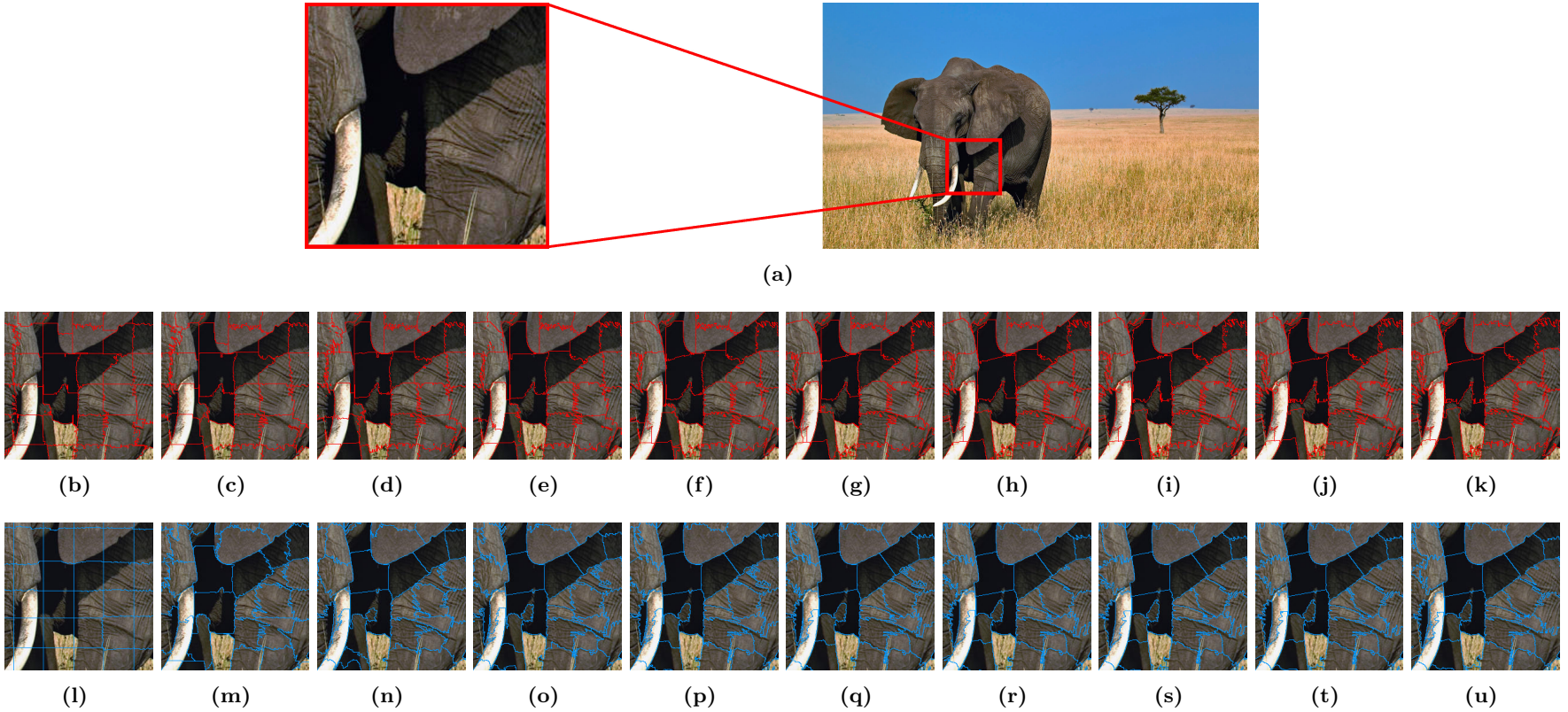


Figura 3.3: Comparação dos resultados do SUTP e do SLIC: (a) segmento (250x250) de uma imagem com dimensões 1920x1080, (b-k) resultados das 10 primeiras iterações do SUTP, (l-u) resultados das 10 primeiras iterações do SLIC. O SUTP tem como configuração $\lambda_1 = 1.0$ $\lambda_2 = 0.005$, para o SLIC $m = 15.0$, e para ambas técnicas inicialmente os *superpixels* têm dimensões 50x50 ($S = 50$).

Capítulo

4

Metodologia

4.1 Considerações Iniciais

Neste capítulo é apresentada a metodologia proposta para a segmentação de imagens empregando algoritmos de agrupamento em grafos. A Seção 4.2 apresenta as etapas que compõem a metodologia proposta. E a Seção 4.3 apresenta a configuração das avaliações experimentais.

4.2 Metodologia Proposta

O diagrama da Figura 4.1 ilustra as etapas que constituem a metodologia proposta para este trabalho. Inicialmente, dada uma imagem, computam-se os *superpixels* (1). A seguir, é realizada a extração de características sobre os *superpixels* (2). Após a criação de um grafo (3), realiza-se a segmentação a partir dos *superpixels* da imagem empregando algoritmos de agrupamento em grafos (4). Finalmente é avaliada a qualidade da segmentação para cada um dos algoritmos de agrupamento em grafos (5).

Cada uma das etapas da metodologia proposta é descrita com maiores detalhes, a seguir.

4.2.1 Extração de Superpixels

A extração de *superpixel* para cada imagem é obtida empregando-se o algoritmo proposto por Achanta et al. (2010, 2012). Este algoritmo está detalhado no Algoritmo 3.2, e requer um parâmetro de entrada: a dimensão do lado do *superpixel* (**tamanho do *superpixel***). A saída desta

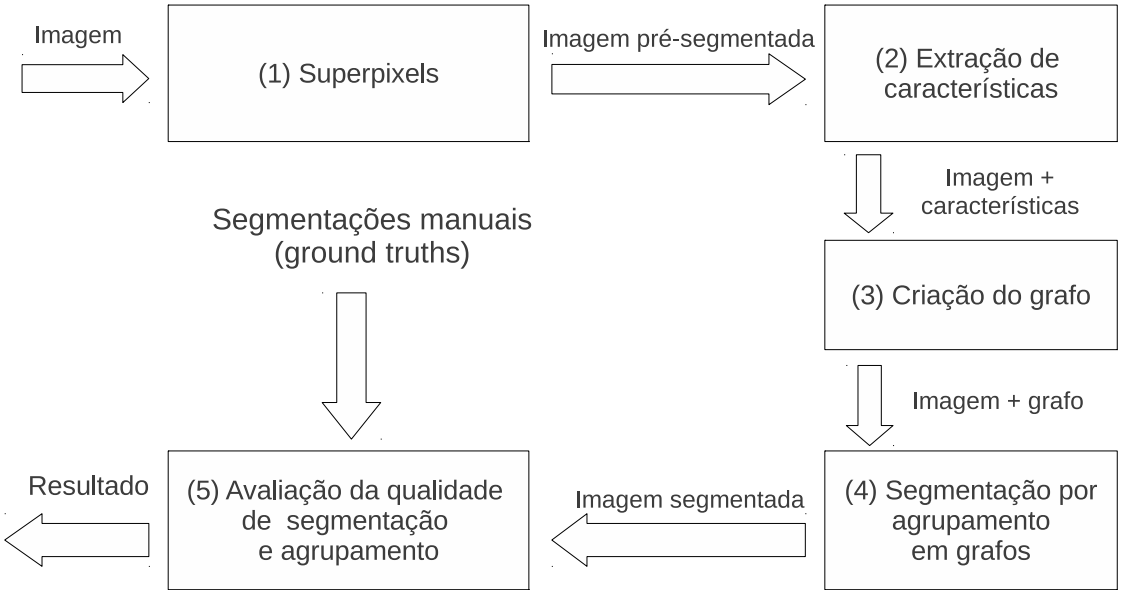


Figura 4.1: Proposta da metodologia para a segmentação de imagens empregando algoritmos de agrupamento em grafos.

operação é uma pre-segmentação da imagem, representada em pequenos segmentos denominados *superpixels*. A Figura 4.2 ilustra um exemplo da pre-segmentação por *superpixels*.

(a) **(b)**

Figura 4.2: Extração de *superpixels*: (a) imagem original, (b) extração de *superpixels*. A técnica empregada para este caso é a implementação proposta por Achanta et al. (2010, 2012).

A qualidade dos *superpixels* gerados influência diretamente na qualidade da segmentação final da imagem. No que se refere ao tamanho dos *superpixels*, este é inversamente proporcional à cardinalidade do grafo. Na Figura 4.2 foi empregado *superpixels* com dimensão inicial do lado (s) igual a 10. Como demostraremos nos experimentos, esta dimensão $s = 10$ foi o que proporcionou os melhores resultados de segmentação, com tempo de processamento baixos mesmo para imagens de alta dimensão.

4.2.2 Extração de Características

Sobre cada *superpixels* (região) é aplicado um **descritor de características** de cor. Nesta dissertação foram implementados 2 tipos de descritores: a média de cores é um descritor tridimensional computado pela Equação 4.4, onde os termos $M_{(1,C_x)}$ são calculados pela Equação 4.1, sendo x um dos 3 canais de cores de cada um dos espaços de cores empregados (RGB, HSI, HSV e CILAB). O segundo descritor baseia-se nos 3 momentos de cor que são computados em um vetor de 9 dimensões, de acordo com a Equação 4.5.

$$M_{(1,C_x)} = \frac{1}{N}\sum_{i \in S}(i_{C_x}) = m, \tag{4.1}$$

$$M_{(2,C_x)} = \sqrt{(\frac{1}{N}\sum_{i \in S}(i_{C_x} - m)^2)}, \tag{4.2}$$

$$M_{(3,C_x)} = \sqrt[3]{(\frac{1}{N}\sum_{i \in S}(i_{C_x} - m)^3)}, \tag{4.3}$$

onde S é o conjunto de N *pixels* de um *superpixel*, i_{C_x} representa um dos 3 canais de cor de cada *pixel*.

$$DMedia3 = (M_{(1,C_1)}, M_{(1,C_2)}, M_{(1,C_3)}), \tag{4.4}$$

$$DMomentos9 = (M_{(1,C_1)}, M_{(2,C_1)}, M_{(3,C_1)}, M_{(1,C_2)}, M_{(2,C_2)}, M_{(3,C_2)}, M_{(1,C_3)}, M_{(2,C_3)}, M_{(3,C_3)}), \tag{4.5}$$

onde $M_{(1,C_1)}$, $M_{(2,C_1)}$, $M_{(3,C_1)}$, $M_{(1,C_2)}$, $M_{(2,C_2)}$, $M_{(3,C_2)}$, $M_{(1,C_3)}$, $M_{(2,C_3)}$ e $M_{(3,C_3)}$ representam os 3 primeiros momentos de cor para cada canal (C_1, C_2 e C_3) de cada espaço de cor (RGB, HSI, HSV, CILAB).

A Tabela 4.1 apresenta os descritores de cor utilizados neste trabalho com seus componentes e respectivos nomes.

A Tabela 4.2 ilustra exemplos de vetores obtidos pela Equação 4.5.

Tabela 4.1: Descritores de cor empregados nesta dissertação.

Descritor	vetor de características
RGB3	$(M_{(1,R)}, M_{(1,G)}, M_{(1,B)})$
HSI3	$(M_{(1,H)}, M_{(1,S)}, M_{(1,I)})$
HSV3	$(M_{(1,H)}, M_{(1,S)}, M_{(1,V)})$
CILAB3	$(M_{(1,L)}, M_{(1,A)}, M_{(1,B)})$
RGB9	$(M_{(1,R)}, M_{(2,R)}, M_{(3,R)}, M_{(1,G)}, M_{(2,G)}, M_{(3,G)}, M_{(1,B)}, M_{(2,B)}, M_{(3,B)})$
HSI9	$(M_{(1,H)}, M_{(2,H)}, M_{(3,H)}, M_{(1,S)}, M_{(2,S)}, M_{(3,S)}, M_{(1,I)}, M_{(2,I)}, M_{(3,I)})$
HSV9	$(M_{(1,H)}, M_{(2,H)}, M_{(3,H)}, M_{(1,S)}, M_{(2,S)}, M_{(3,S)}, M_{(1,V)}, M_{(2,V)}, M_{(3,V)})$
CILAB9	$(M_{(1,L)}, M_{(2,L)}, M_{(3,L)}, M_{(1,A)}, M_{(2,A)}, M_{(3,A)}, M_{(1,B)}, M_{(2,B)}, M_{(3,B)})$

Tabela 4.2: Representação de um conjunto de vetores: 1000 vetores com 9 características ($c1$... $c9$) normalizadas na faixa $[0,1]$. Cada amostra v_i corresponde a um *superpixel* denotado na imagem.

Vetor	c1	c2	c3	c4	c5	c6	c7	c8	c9
v_1	0.212	0.678	0.637	0.135	0.675	0.764	0.334	0.793	0.678
v_2	0.512	0.483	0.129	0.457	0.564	0.231	0.861	0.461	0.461
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
v_{999}	0.351	0.451	0.876	0.097	0.937	0.434	0.441	0.431	0.941
v_{1000}	0.041	0.751	0.566	0.347	0.174	0.747	0.214	0.671	0.685

4.2.3 Criação do Grafo

Nesta etapa é construído um grafo ponderado não dirigido, onde os nós são representados pelos vetores de características obtidos na etapa anterior e os pesos das arestas são calculados segundo as funções de similaridade formuladas nesta dissertação, baseadas na Equação 4.6:

$$w_F(i,j) = e^{-\left(\frac{F^2}{\sigma^2}\right)}, \tag{4.6}$$

onde i, j são dois vetores que representam dois nós do grafo, e F uma das seguintes funções de distância entre i e j: Euclidiana, *Manhattan*, *Chebyshev*, *1-Coseno*, *1-Tanimoto*, *1-Fu* e *Mahalanobis*. Todas estas funções foram descritas na Seção 2.2.4 do Capítulo 2. O parâmetro σ, seguindo as investigação de trabalhos existentes, pode assumir valores entre 10 e 20 por cento de toda a faixa de distâncias do conjunto de vetores de características (Shi e Malik, 2000). A Tabela 4.3 mostra as funções de similaridade propostas nesta dissertação, baseadas na função de similaridade *Gaussiana*, amplamente utilizada no agrupamento em grafos em vários trabalhos (Shi e Malik, 2000; Sakai e Imiya, 2009; Yan et al., 2009; Tung et al., 2010; Cigla e Alatan, 2010).

Tabela 4.3: Funções de similaridade formuladas neste trabalho: funções de similaridade propostas estão marcadas com um asterisco.

Denotação	Formulação
gEU	$w_{gEU}(i,j) = e^{-\left(\frac{Euclidean(i,j)^2}{\sigma^2}\right)}$
gMH*	$w_{gMH}(i,j) = e^{-\left(\frac{Manhatatan(i,j)^2}{\sigma^2}\right)}$
gCH*	$w_{gCH}(i,j) = e^{-\left(\frac{Chebyshev(i,j)^2}{\sigma^2}\right)}$
gCO*	$w_{gCO}(i,j) = e^{-\left(\frac{1-Cosine(i,j)^2}{\sigma^2}\right)}$
gTA*	$w_{gTA}(i,j) = e^{-\left(\frac{1-Tanimoto(i,j)^2}{\sigma^2}\right)}$
gFU*	$w_{gFU}(i,j) = e^{-\left(\frac{1-Fu(i,j)^2}{\sigma^2}\right)}$
gMB*	$w_{gMB}(i,j) = e^{-\left(\frac{Mahalanobis(i,j)^2}{\sigma^2}\right)}$

Para a construção do grafo, além da função de similaridade, dois outros parâmetros são necessários: raio e limiar. O raio r centrado em um *superpixel* i delimita a vizindade de *superpixels* a ser considerados na montagem do grafo e consequente criação do grafo. Qualquer *superpixel* j é considerado um potencial candidato a ser conectado a i, portanto, gerando uma nova aresta no grafo. O limiar define a existência da aresta ou não. Se $w(i,j) \geq limiar$, então a aresta é adicionada ao grafo. Neste trabalho, no entanto, será também avaliado o desempenho de alguns algoritmos sem a utilização do limiar, ou seja, existe uma aresta conectando todo *superpixel* i e j. A Figura 4.3 ilustra o processo de criação após a geração de n *superpixels* e dos vetores de características, dados um *superpixel* i, um limar t e um raio r centrado em i, criam-se, para cada *superpixel* j dentro do alcance de r, as arestas do grafo. Para cada aresta do grafo é atribuído o peso w para o par de *superpixels* (i,j), definido pela função de similaridade $w(i,j)$. Note que o número de nós no grafo é igual ao número de *superpixels*.

4.2.4 Segmentação por Agrupamento em Grafos

A segmentação de imagens é computada para cada imagem cumprindo os seguintes passos:

1. **Agrupamento dos nós do grafo**: para cada grafo são executados os 6 algoritmos de agrupamento em grafos: *spectral clustering* SC1 sem limiar (1) e com limiar SC2 (2), *label propagation* (3), *fast greedy* (4) e multinivel - *graclus* ML1 sem limar (5) e *graclus* ML2 com limiar (6). O processo de agrupamento consiste em rotular nós (*superpixels*) como membros de um mesmo grupo. Ao final do processo nós com o mesmo rótulo indicam um certo grupo, o que representa uma ou mais regiões na imagem.

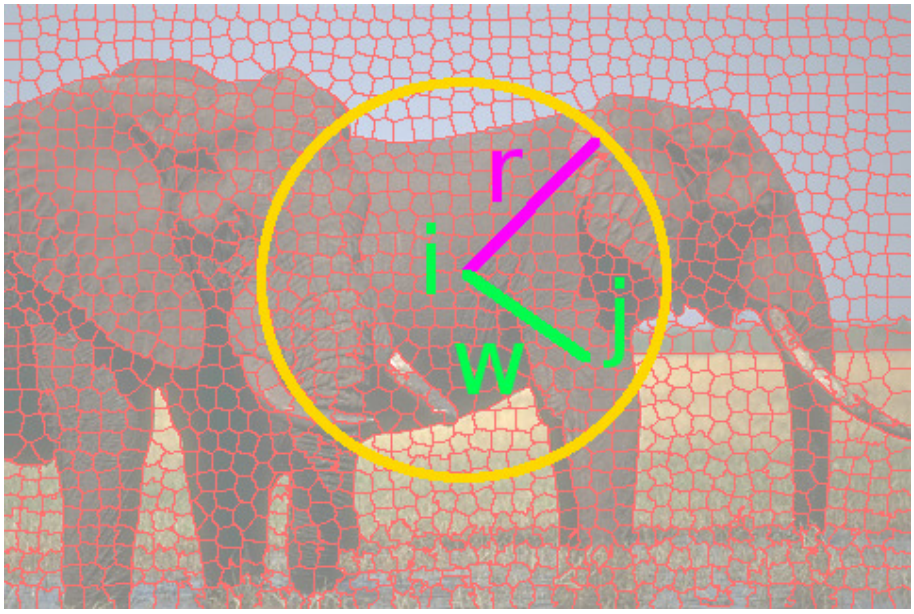


Figura 4.3: Criação do grafo: *i superpixels* atual, *j* representa aos *superpixels* vizinhos de *i* delimitados por *r*, que é o raio que define a vicindade para o *superpixel i*, *w* representa o peso da aresta que é calculada por meio de uma função de similaridade.

2. **Obtenção dos segmentos/regiões**: depois do agrupamento dos nós do grafo, a segmentação é obtida pela geração de segmentos a partir dos nós do grafos (*superpixels*). Os segmentos são gerados mapeando os *pixels* dos *superpixels* sobre a imagem. Posteriormente, é preciso mapear os grupos da etapa anterior em regiões da imagem. Este processo é feito utilizando-se um algoritmo de componentes conexas como descrito no Algoritmo B.1 do Apêndice A e ilustrado na Figura 4.4. Na imagem da Figura 4.4**(a)**, observa-se que o grupo do rotulo 1 não é conectado. Após a aplicação do algoritmo de componentes conexas, os *pixels* deste grupo serão divididos em duas regiões distintas na imagem (1 e 9) como é ilustrado na Figura 4.4**(b)**.

4.2.5 Avaliação da Qualidade da Segmentação e do Agrupamento em Grafos

Avaliação da Qualidade da Segmentação

Em visão computacional a segmentação de imagens não supervisionada além de ser uma das tarefas mais difíceis de realizar, requer também a **quantificação da qualidade da segmentação**. Uma das abordagens utilizadas baseia-se no conceito de informações de referência (*ground truth*). Nela, as imagens são segmentadas por pessoas (também conhecidas como segmentações manuais), enquanto os resultados dos algoritmos que fazem segmentação de imagens são avaliados em função ao grau de semelhança que têm os resultados destes algoritmos com os resultados das segmentações feitas por pessoas. Neste contexto, a qualidade da segmentação é obtida por meio da comparação quantitativa entre os segmentos das imagens segmentadas por pessoas e os segmentos das imagens obtidas pelos algoritmos que fazem segmentação.

1	1	1	1	3	3	3	3	3	3
1	1	1	1	3	3	7	7	3	3
2	2	1	1	3	7	7	7	3	3
2	2	2	2	7	7	7	7	1	1
6	2	2	2	7	7	7	1	1	1
6	6	4	4	4	1	1	1	1	1
6	6	6	4	4	1	1	1	1	1
6	6	6	5	4	4	1	8	8	8
5	5	5	5	5	5	8	8	8	8
5	5	5	5	5	5	8	8	8	8

(a)

1	1	1	1	3	3	3	3	3	3
1	1	1	1	3	3	7	7	3	3
2	2	1	1	3	7	7	7	3	3
2	2	2	2	7	7	7	7	9	9
6	2	2	2	7	7	7	9	9	9
6	6	4	4	4	9	9	9	9	9
6	6	6	4	4	9	9	9	9	9
6	6	6	5	4	4	9	8	8	8
5	5	5	5	5	5	8	8	8	8
5	5	5	5	5	5	8	8	8	8

(b)

Figura 4.4: Obtenção de regiões com *pixels* conexos: (a) regiões com *pixels* não conexos (b) regiões com *pixels* conexos. Cada requadro enumerado representa o valor da região para um pixel. Para ambas figuras respectivamente os *pixels* de cor rosa representam *pixels* de regiões não conexas e *pixels* de regiões conexas.

Sejam, respectivamente, S' e S o conjunto de regiões pertencentes ao resultado de um algoritmo M e o conjunto de regiões pertencentes à segmentação manual. A Equação 4.8 computa a qualidade da segmentação baseada na cobertura de regiões (***covering***) (Crevier, 2008; Arbelaez et al., 2009) que existe entre S' e S, onde as regiões de S' tentam cobrir as regiões de S. Desta forma é calculado o grau de semelhança entre S' e S.

$$\mathscr{O}(R,R') = \frac{|R \cap R'|}{|R \cup R'|} = \frac{|R \cap R'|}{|R| + |R'| - |R \cap R'|}, \tag{4.7}$$

$$\mathscr{C}(S' \to S) = \frac{1}{A} \sum_{R \in S} |R| \cdot \max_{R' \in S'} \{\mathscr{O}(R,R')\}, \tag{4.8}$$

onde $\mathscr{O}(R,R')$ é a sobreposição (*overlap*) (Ge et al., 2006; Malisiewicz e Efros, 2007) que existe entre as regiões R e R', $|R|$ é o número de *pixels* da região R, e A é o número de pixel da imagem. Os valores que apresenta o *covering* da Equação 4.8 estão na faixa $[0,1]$. Enquanto existir valores que estiveram perto de 1, as regiões de S' serão semelhantes às regiões de S, portanto S' apresenta uma boa qualidade de segmentação.

Considerando que a qualidade da segmentação de uma imagem pode ser calculada com mais de uma segmentação manual, a Equação 4.9 define a qualidade da segmentação que apresenta S' com as G segmentações manuais de uma imagem.

$$A_C = \frac{1}{|G|} \sum_{S \in G} \mathscr{C}(S' \to S), \tag{4.9}$$

onde $|G|$ é o número de segmentações manuais de uma imagem. Desta forma a quantificação do resultado da segmentação S' de um determinado algoritmo é calculada como a média dos valores do *covering* de S' sobre todas as G segmentações manuais de uma imagem.

Seja um conjunto de imagens $\{I_i,...,I_N\}$, onde cada imagem está associada a um outro conjunto $\{G_i,...,G_N\}$ de segmentações manuais. A quantificação da qualidade das segmentação das $\{S'_i,...,S'_N\}$ segmentações do algoritmo M está definida na Equação 4.10.

$$A_M = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{|G_i|}\sum_{S\in G_i}\mathscr{C}(S'_i \to S)\right), \tag{4.10}$$

onde N é o número de imagens. Neste caso a qualidade da segmentação para o algoritmo M é calculada simplesmente como a média das qualidades de segmentação das $\{S'_i,...,S'_N\}$ segmentações obtidas por o M.

Avaliação da Qualidade do Agrupamento

Nesta dissertação os algoritmos de agrupamento em grafos descritos no Capítulo 2 serão avaliados empregando duas métricas: o índice de silhueta e a modularidade, que respectivamente estão definidas nas Equações 2.23 e 2.24 do capítulo 2.

4.3 Configuração das Avaliações Experimentais

4.3.1 Conjunto de Imagens

As imagens utilizadas para os experimentos pertencem a *Berkeley Image Dataset* (BSDS300) (Martin et al., 2001) da Universidade de *Berkeley*. BSDS300 é um conjunto de imagens naturais que apresenta várias segmentações manuais. Este banco de imagens contém 300 imagens, com dimensões $481x321$, e estão divididas em um conjunto de 200 imagens para treinamento, e outras 100 destinadas para testes. Cada imagem possui pelo menos 5 segmentações manuais, com ao menos um objeto, posicionado na maioria dos casos no centro da imagem.

A Figura 4.5 ilustra 4 amostras de imagens BSDS300, com 5 segmentações manuais para cada imagem. Note que em alguns casos existe uma grande variação no número de segmentos entre as diferentes segmentações manuais, como é o caso da estrela (Figura 4.5**(a)**). Nossa avaliação quantitativa de qualidade de segmentação empregará todas as segmentações manuais.

Para efeito de medição de tempo, os testes dos experimentos foram executadas em um computador *Intel*® core *i*7 de 2.67 GHz.

4.3.2 Parâmetros Empregados

A Tabela 4.4 apresenta todos os parâmetros empregados nos experimentos e seus respectivos símbolos. O raio (r), limiar (τ) e o tamanho do *superpixel* (s) influenciam diretamente a montagem do grafo, como é descrito na Seção 4.2.3. Seis configurações distintas de algoritmos de

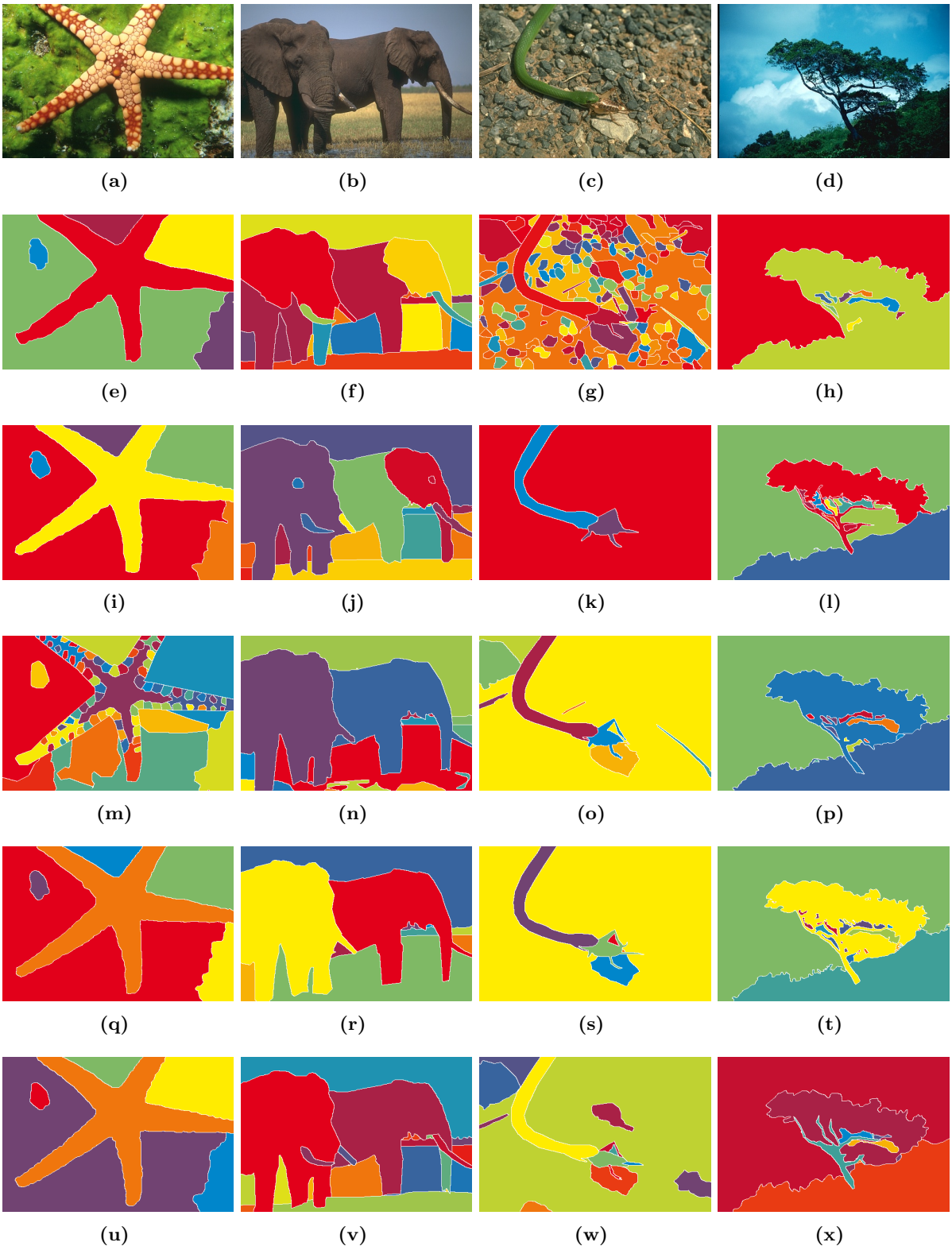


Figura 4.5: Exemplo de 4 imagens do conjunto de imagens do BSDS300: (a-d) imagens de cenas naturais, (e-x) segmentações manuais para cada imagem (regiões coloridas).

agrupamento em grafos foram adotadas nos experimentos: LP (*label propagation*), FG (fast greedy), SC1 (*spectral clustering* sem limiar), SC2 (spectral clustering com limiar), ML1 (*Graclus* multinível sem limiar) e ML2 (*Graclus* multinível com limiar). O limiar τ usado para calcular arestas é descrito na Seção 4.2.3. LP e FG sempre utilizam o limiar no cálculo das arestas. Varias funções de similaridade (fs) (gEU, gMH, gCH, gCO, gTA, gFU, gMB) e descritores de cor (dc) (RGB3, HSI3, HSV3, CILAB3, RGB9, HSI9, HSV9, CILAB9) serão avaliados e combinados com os diversos algoritmos de agrupamento. As funções de similaridade e os descritores de cores são descritos, respectivamente, nas Seções 4.3 e 4.2.2. O parâmetro (σ) é empregado no cálculo da similaridade gaussiana, presente em todas as funções de similaridade. O parâmetro (s) representa a dimensão do lado inicial dos *superpixels* e (k) representa, para os algoritmos de agrupamento SC1, SC2, ML1 e ML2, o número de partições ou agrupamentos que têm que gerar.

Tabela 4.4: Parâmetros identificados nos passos da segmentação de imagens utilizando agrupamento em grafos.

Parâmetro	símbolo
Raio	r
Limiar	τ
Tamanho *superpixel*	s
Número de partições	k
Parâmetro similaridade gaussiana	σ
Descritor de características	dc
Função de similaridade	fs

Para melhor compreender a relação entre agrupamento em grafos e a segmentação é importante avaliar a correlação que possa existir entre a qualidade do agrupamento do grafo (Equações 2.23 e 2.24) e a qualidade de segmentação da imagem (Equação 4.9). Esta avaliação é realizada segundo a Equação 4.11 que define o grau de correlação entre 2 variáveis, conhecida também como "Coeficiente de Correlação de Pearson".

$$p = \frac{n\sum xy - (\sum x)(\sum y)}{\sqrt{n\sum x^2 - (\sum x)^2}\sqrt{n\sum y^2 - (\sum y)^2}} \tag{4.11}$$

onde x representa a qualidade da segmentação e y representa a qualidade de agrupamento do grafo.

Capítulo

5

Resultados Experimentais

Este capítulo apresenta os resultados para a segmentação de imagens por meio do agrupamento em grafos. Quatro experimentos são fornecidos. **O experimento 1** tem por objetivo avaliar os 8 descritores de características de cor em combinação com os 6 algoritmos de agrupamento formulados. **O experimento 2** tem por objetivo avaliar as 7 funções de similaridade em combinação com os 6 algoritmos de agrupamento formulados. **O experimento 3** visa avaliar a correlação entre a qualidade de segmentação e a qualidade de agrupamento em grafos. Finalmente, **o experimento 4** traz uma análise sobre a escalabilidade para os métodos de agrupamento implementados neste trabalho. Pela grande quantidade de resultados gerados neste trabalho, foi disponibilizado um *site*[1] onde são visualizados todos os resultados qualitativos dos quatro experimentos.

A qualidade de segmentação será avaliada tanto qualitativa quanto quantitativamente. A avaliação qualitativa é feita visualmente e está sujeita à subjetividade na interpretação. A avaliação quantitativa é feita como descrita no capítulo anterior e utiliza-se de amostras segmentadas manualmente (*ground truth*) para gerar uma medida da qualidade da imagem segmentada pelo método proposto, conforme as Equações 4.9 e 4.10. Uma segmentação é dita de boa qualidade quando o valor médio dado pela Equação 4.10 for alto, além de considerar a baixa dispersão dos resultados para todo o conjunto de imagens analisada. Esta dispersão pode ser observada nos *boxplots* fornecidos neste capítulo.

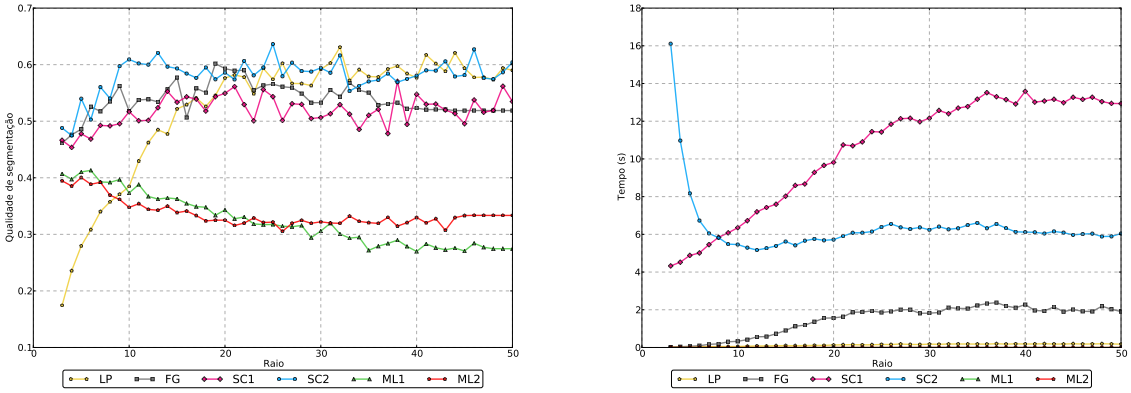
Como descrito na metodologia, a segmentação baseada em particionamento de grafos requer alguns parâmetros. Neste trabalho, os valores de raio r, limiar τ e o lado s do *superpixel* foram determinados empiricamente.

[1] https://imgsgc.googlecode.com/svn/trunk/tests/index.html

Para determinar o valor de raio mais apropriado, foram realizadas diversas segmentações para variados valores, computando-se a qualidade da segmentação e o tempo consumido para cada um deles. As Figuras, 5.1**(a)** e 5.1**(b)** ilustram, respectivamente, os valores para a qualidade e o tempo de execução, para raios variando entre 5 e 50. Note que a qualidade mantem-se relativamente constante para valores de $r \geq 15$, enquanto que o tempo de execução tem um comportamento menos previsível, pois varia de acordo com o algoritmo aplicado. Um valor compromisso entre a qualidade de segmentação e tempo de execução foi definido para $r = 15$.

De maneira análoga, foi realizada a avaliação do melhor valor para o limiar τ. As Figuras 5.2**(a)** e 5.2**(b)** ilustram, respectivamente, a variação da qualidade de segmentação e tempo para diversos limiares. Combinando-se ambos gráficos, definiu-se o valor de $\tau = 0.46$ como o valor ideal de segmentação. Note que enquanto permite a computação em tempo execução bastante reduzido, este limiar ainda preserva os melhores valores para a qualidade de segmentação, antes de conduzir a uma queda substancial para todos os algoritmos testados.

As Figuras 5.3**(a)** e 5.3**(b)** mostram, respectivamente, os valores de qualidade de segmentação e tempo de execução para diversos valores de lado de *superpixel*. Note que tanto tempo quanto a qualidade de segmentação caem drasticamente, à medida que cresce o tamanho do *superpixel*. Para privilegiar a qualidade da segmentação, em detrimento do tempo de execução, o valor adotado nestes experimentos foi fixado em $s = 10$. Note que para este valor, ainda assim é possível produzir, para alguns dos métodos de agrupamento, segmentações em tempos baixos.

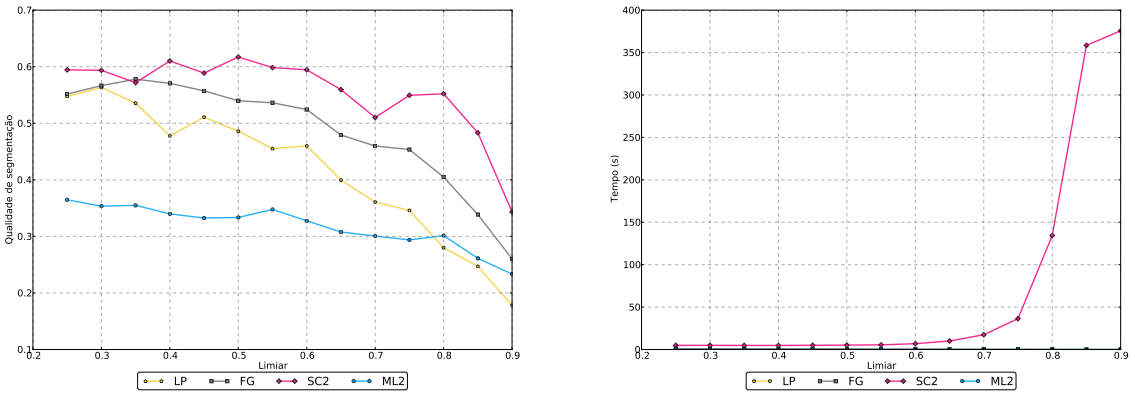


(a) **(b)**

Figura 5.1: Variação do raio: (a) qualidade de segmentação e (b) tempo de processamento em segundos.

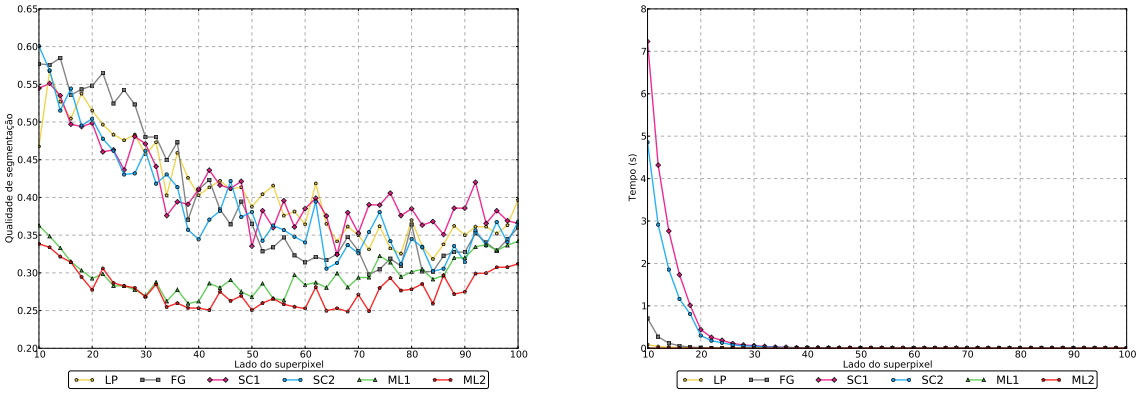
5.1 Experimento 1

Para este experimento foram empregadas 10 imagens do conjunto BSDS300 (Figuras A.1 e A.2 do Apêndice A). Para todos os 6 algoritmos formulados foram utilizados os seguintes parâmetros: $r = 15$, $\tau = 0.46$, $s = 10$, $k = 7$, $\sigma = 15\%$ e $fs = gEU$. Cada um dos 6 algoritmos foi avaliado para cada um dos descritores de cor, $dc = \{RGB3, HSI3, HSV3, CILAB3, RGB9, HSI9, HSV9, CILAB9\}$.



(a) **(b)**

Figura 5.2: Variação do limiar: (a) qualidade de segmentação e (b) tempo de processamento em segundos.



(a) **(b)**

Figura 5.3: Variação do lado do *superpixel*: (a) qualidade de segmentação e (b) tempo de processamento em segundos.

O valor de k foi estimado por meio da observação direta das imagens do conjunto BSDS300, em que o número máximo de regiões se limitava a 7. A escolha de $fs = gEU$ deve-se ao fato de que é geralmente empregada em vários trabalhos (Shi e Malik, 2000; Cigla e Alatan, 2010; Maier et al., 2011). Os resultados quantitativos são ilustrados nas Figuras 5.4 e 5.5, e a segmentação das 10 imagens empregadas é ilustrada na Figura 5.6.

A Figura 5.4 mostra o *boxplot* em que cada um descritores *dc* é avaliado para cada um dos seis algoritmos de agrupamento. A Figura 5.5 ilustra um histograma que indica a taxa de precisão para cada descritor *dc*. Cada barra do histograma representa, para uma determinada qualidade de segmentação com variação de 10 pontos percentuais, a quantidade total de imagens segmentadas para cada um dos 4 algoritmos de agrupamento (os algoritmos ML1 e ML2 não foram incluídos).

As Figuras revelam que não há uma diferença marcante entre os vários descritores de cores empregados, embora possa ser observada uma leve vantagem no caso do descritor CILAB9. Note

que o histograma é levemente deslocado para a direita (melhores resultado) quando comparado aos demais. Observa-se também que o desempenho de cada descritor varia conforme o algoritmo empregado, razão que explica a ausência de um descritor que se destaca na tarefa de segmentação.

O experimento revela que os algoritmos multiníveis não são adequados para a segmentação de imagens, pois produzem valores muito baixos para a qualidade de segmentação, como revela o *boxplot* da Figura 5.4. Já os algoritmos de *spectral clustering* (SC), *fast greedy* (FG) e *label propagation* (LP) produzem bons resultados, com leve vantagem para a configuração (SC2). No entanto, estes sofrem do problema da escalabilidade, como mostram as Figuras 5.1**(b)**, 5.2**(b)** e 5.3**(b)**. A conclusão, portanto, é de que os algoritmos de FG e LP são os mais apropriados para a segmentação de imagens.

Com o intuito de melhor compreender o significado da avaliação quantitativa fornecida por meio do *boxplot* da Figura 5.4 e dos histogramas da Figura 5.5, são apresentadas, na Figura 5.7 e na Tabela 5.1, os resultados das melhores e piores imagens segmentadas para cada um dos 6 algoritmos avaliados. Por melhor e pior imagens segmentadas, entenda-se aquelas que produziram os maiores e menores valores de qualidade, respectivamente. De cima para baixo, são ilustrados os pares obtidos para os algoritmos LP, FG, SC1, SC2, ML1 e ML2, respectivamente. Note que as piores segmentações correspondem a imagens que são ou super segmentadas ou sub segmentadas e quando comparadas às segmentações manuais, diferem consideravelmente. As imagens que produziram os melhores valores de qualidade de segmentação são aquelas que, na média, são mais parecidas às segmentações manuais, como era de se esperar.

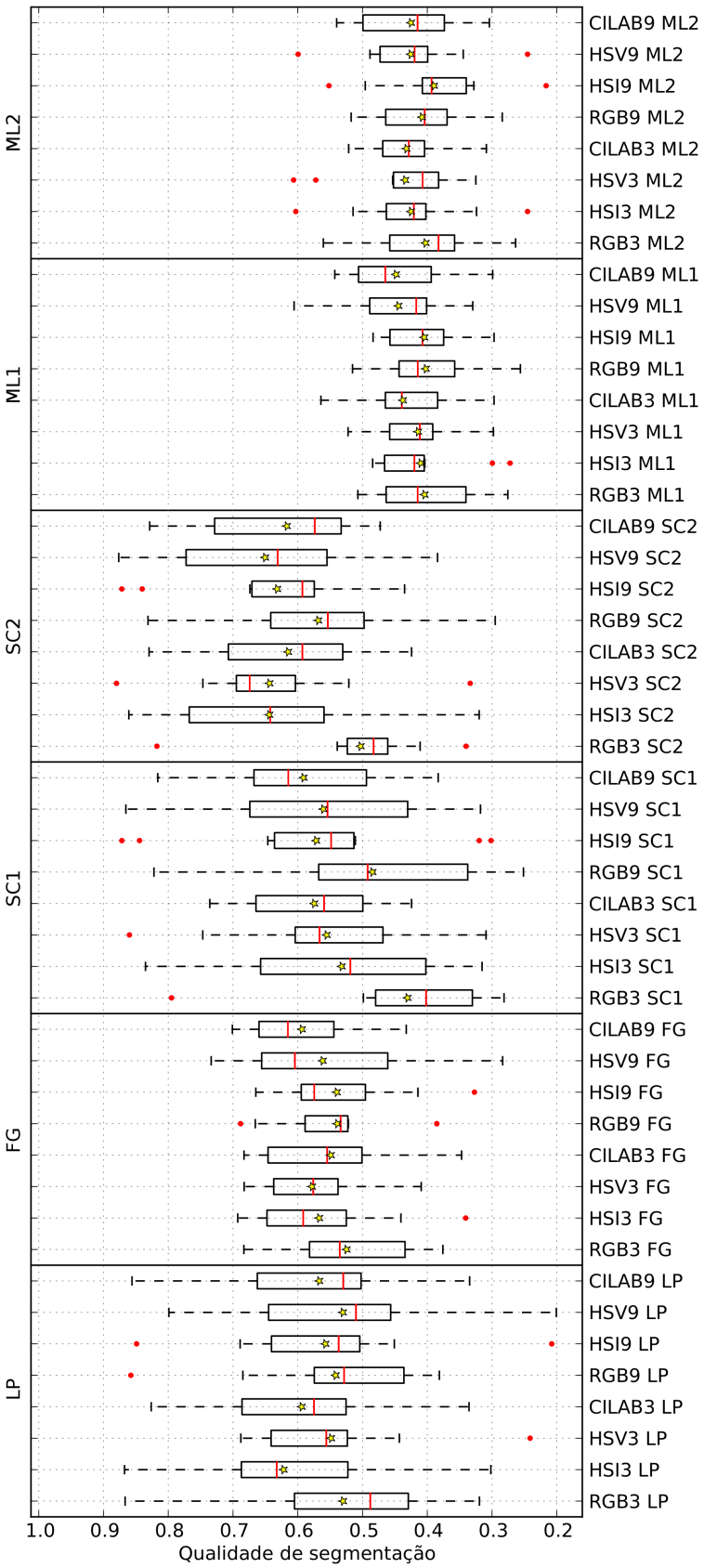
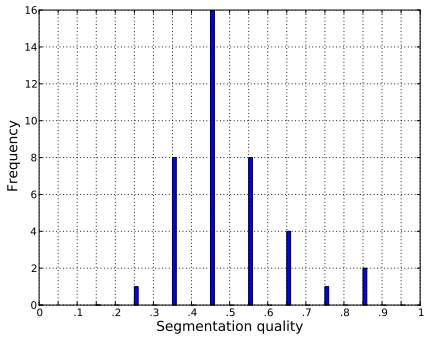
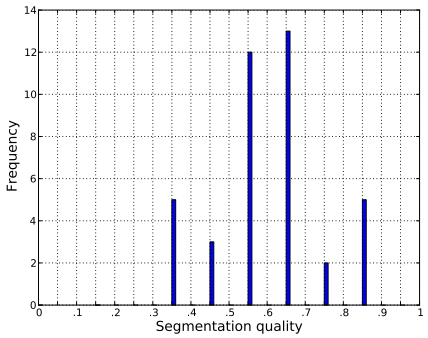


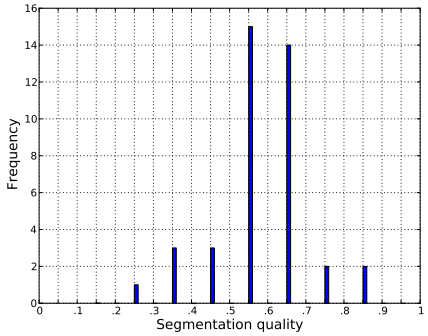
Figura 5.4: Resultados da avaliação dos descritores de características formulados nesta dissertação: as linhas vermelhas representam as medianas, as estrelas representam as médias e os pontos vermelhos representam os *outliers*. Cada item do *boxplot* descreve o desempenho de cada um dos 6 algoritmos de agrupamento para os 8 descritores. A função de similaridade empregada foi $fs = gEU$.



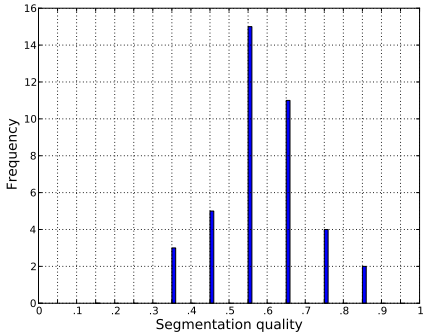
(a) RGB3



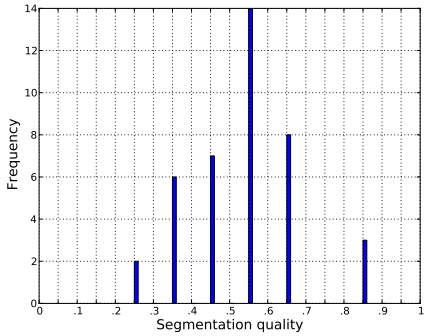
(b) HSI3



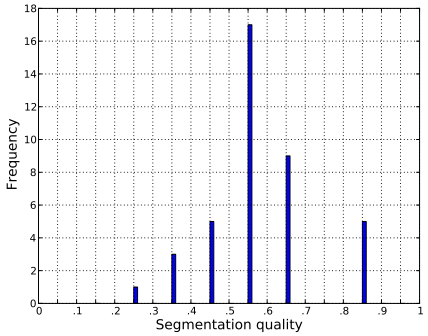
(c) HSV3



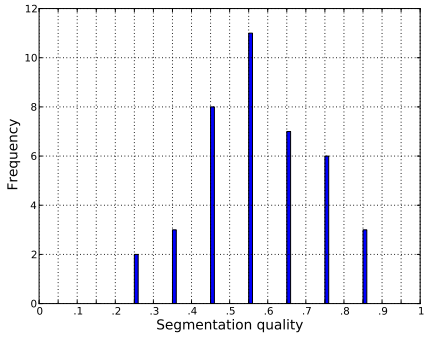
(d) CILAB3



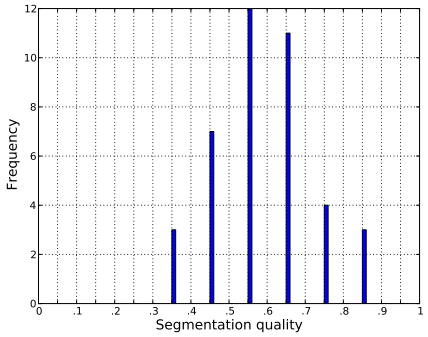
(e) RGB9



(f) HSI9



(g) HSV9



(h) CILAB9

Figura 5.5: Histogramas dos resultados dos descritores de cor.

Figura 5.6: Resultados qualitativos obtidos com o descritor CILAB9. Primeira linha: 10 imagens originais do conjunto BSDS300. Demais linhas, de cima para baixo, respectivamente: resultados para os algoritmos LP, FG, SC1, SC2, ML1 e ML2.

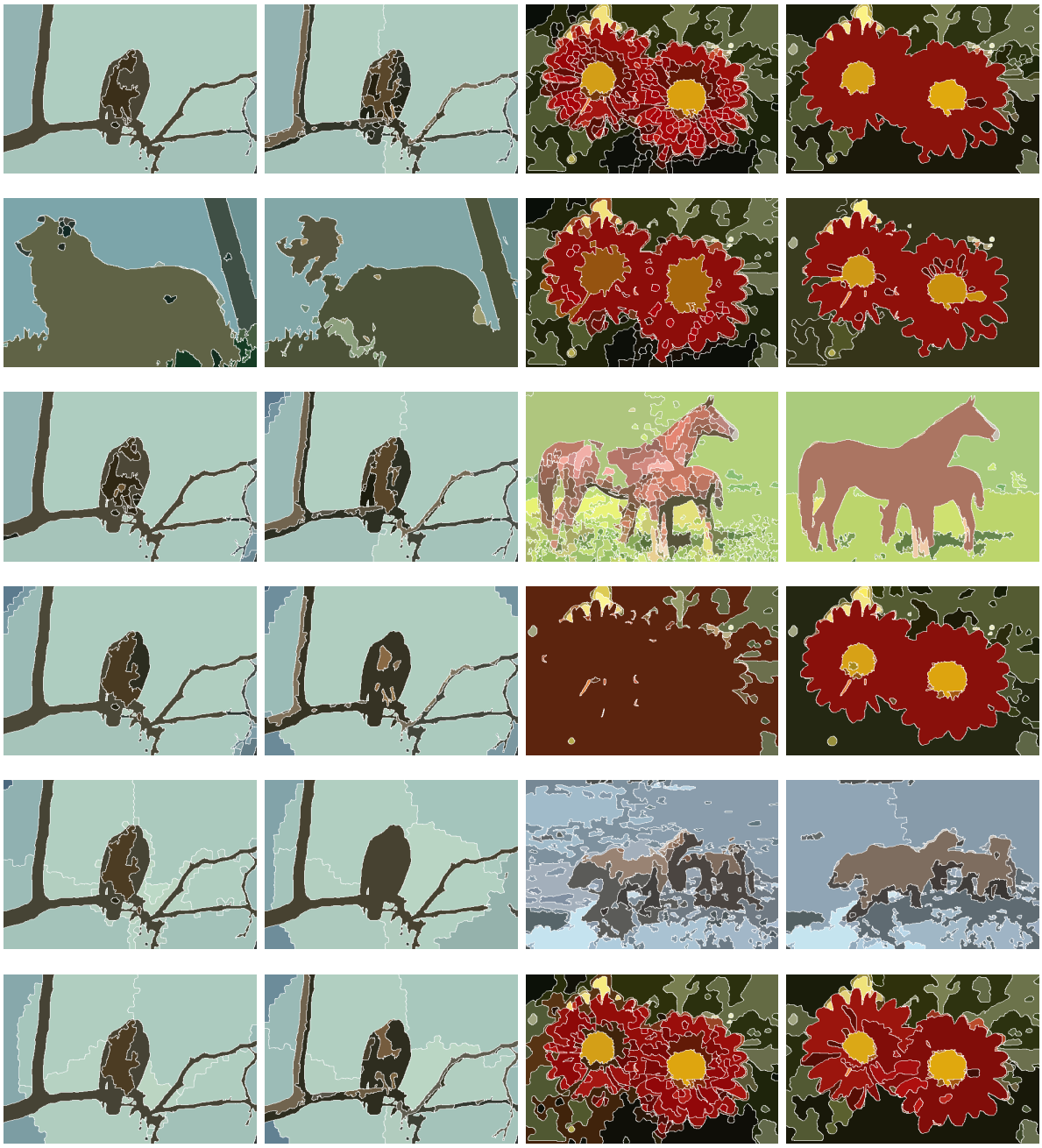


Figura 5.7: Melhores e piores resultados do experimento 1: de cima para baixo melhor (primeira coluna) e pior (terceira coluna) resultado para os algoritmos LP, FG, SC1, SC2, ML1 e ML2. A segunda e quarta coluna apresentam, respectivamente os piores e melhores resultados para cada imagem.

Tabela 5.1: Melhores e piores resultados do experimento 1. Segundo os resultados qualitativos da Figura 5.7, esta tabela apresenta os dados quantitativos dos melhores e piores resultados.

	Melhores					Piores				
Alg.	Imagem	dc	$A_c >$	dc	$A_c <$	Imagem	dc	$A_c <$	dc	$A_c >$
LP	42049.jpg	HSI3	0.8675	HSI9	0.6386	124084.jpg	HSV9	0.2007	RGB9	0.4043
FG	247085.jpg	HSV9	0.7337	RGB9	0.5219	124084.jpg	HSV9	0.2836	RGB9	0.5250
SC1	42049.jpg	HSI9	0.8717	CILAB3	0.6059	113044.jpg	RGB9	0.2515	CILAB9	0.6722
SC2	42049.jpg	HSV3	0.8799	RGB3	0.8175	124084.jpg	RGB9	0.2951	CILAB9	0.4745
ML1	42049.jpg	HSV9	0.6057	HSI3	0.4748	94079.jpg	RGB9	0.2562	HSV9	0.3442
ML2	42049.jpg	HSV3	0.6065	CILAB3	0.5071	124084.jpg	HSI9	0.2165	RGB3	0.4665

5.2 Experimento 2

Para este experimento foram empregadas 10 imagens do conjunto BSDS300 (Figuras A.1 e A.2 do Apêndice A). Para todos os 6 algoritmos formulados foram utilizados os seguintes parâmetros: $r = 15$, $\tau = 0.46$, $s = 10$, $k = 7$, $\sigma = 15\%$ e $dc = CILAB9$. Cada um dos 6 algoritmos foi avaliado para cada uma das funções de similaridade, $fs = \{gEU, gMH, gCH, gCO, gTA, gFU, gMB\}$. O valor de k foi estimado por meio da observação direta das imagens do conjunto BSDS300, em que o número máximo de regiões se limitava a 7. A escolha de $dc = CILAB9$ deve-se ao fato de que esta, na média, produziu um dos melhores resultados, como é visto no experimento 1. Os resultados quantitativos são ilustrados nas Figuras 5.8 e 5.9, e a segmentação para as 10 imagens empregadas é ilustrada na Figura 5.10.

A Figura 5.8 ilustra o desempenho de cada uma das 7 funções de similaridade fs para cada um dos 6 algoritmos de agrupamento, totalizando 42 medidas. Os histogramas da Figura 5.9 (computados sobre os valores apresentados no *boxplot* da Figura 5.8) indicam a taxa de precisão para cada função de similaridade fs. Cada barra do histograma representa, para uma determinada qualidade de segmentação com variação de 10 pontos percentuais, a quantidade total de imagens segmentadas para cada um dos 4 algoritmos de agrupamento (os algoritmos ML1 e ML2 não foram incluídos no cálculo).

As Figuras revelam que não há uma diferença marcante entre as várias funções de similaridade empregadas, embora possa ser observada uma marcada desvantagem no caso da função gMB (Mahalanobis). Note que o histograma gerado a partir desta função é levemente deslocado para a esquerda (o que indica os piores resultados) quando comparado aos demais. Observa-se também que o desempenho de cada função varia conforme o algoritmo empregado, razão que explica a ausência de uma função que se destaque na tarefa de segmentação. No entanto também é notável que as combinações das funções de similaridade com SC2 apresentam melhoras na qualidade de segmentação em comparação às combinações feitas com SC1. Isto se explica pela presença do limiar (SC2), que gera grafos mais adequados agrupamento, que resultarão em regiões mais precisas nas imagens segmentadas.

Analogamente ao experimento anterior, é apresentada uma comparação entre os resultados quantitativos e qualitativos, ilustrando as melhores e piores segmentações para cada um dos algoritmos, segundo cada uma das medidas de similaridade, como ilustrado na Figura 5.11 e na Tabela 5.2. De cima para baixo, são ilustrados os pares obtidos para os algoritmos LP, FG, SC1,

SC2, ML1 e ML2, respectivamente. É notável que a maioria dos piores resultados correspondem à função de similaridade gMB (Mahalanobis).

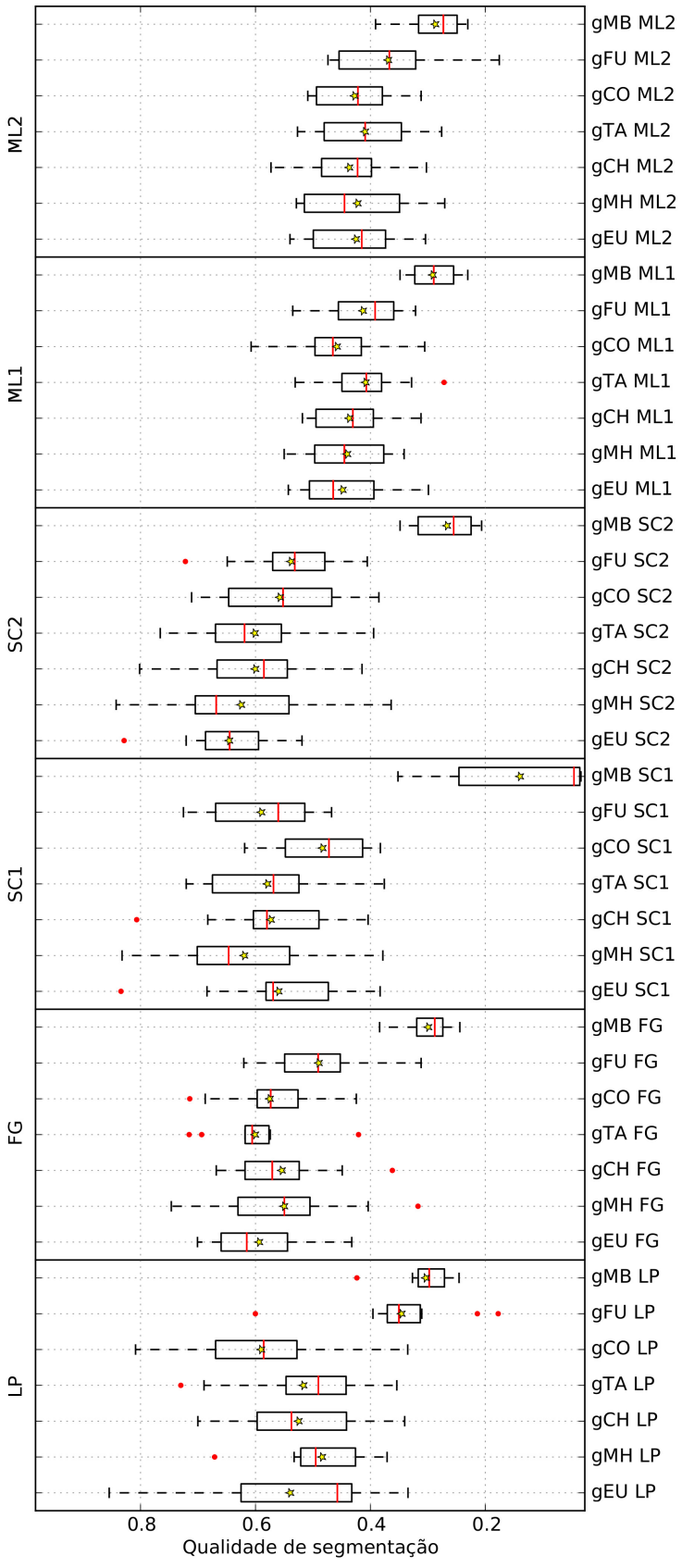
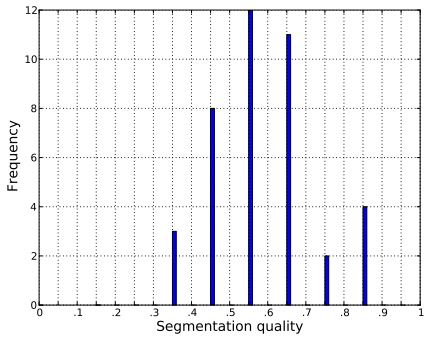
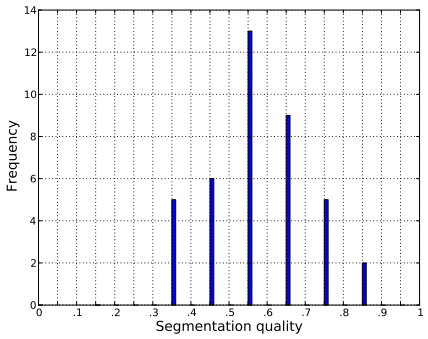


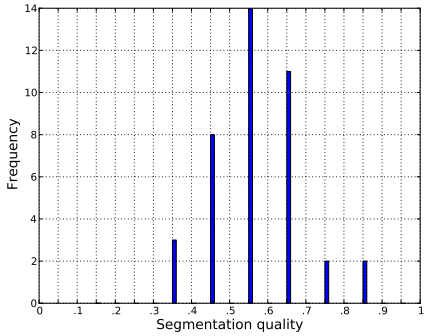
Figura 5.8: Avaliação das funções de similaridade: em cada retângulo, a linha vermelha representa a mediana, o asterisco a média e os pontos vermelhos representam os *outliers*. Cada um dos 42 elementos do *boxplot* indica o resultado da segmentação de fs para cada um dos 6 algoritmos de agrupamento, considerando as 10 imagens do conjunto BSDS300.



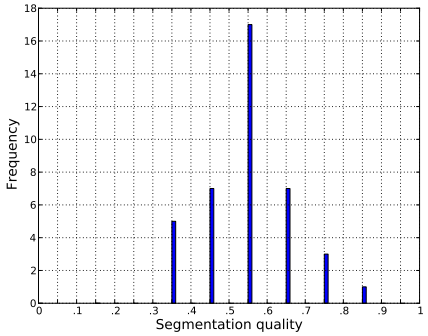
(a) gEU



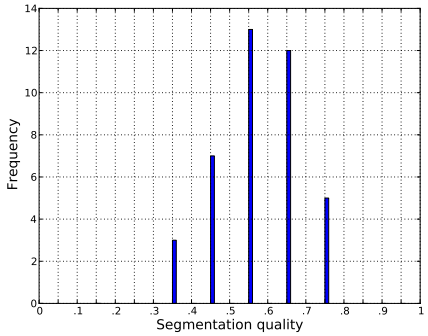
(b) gMH



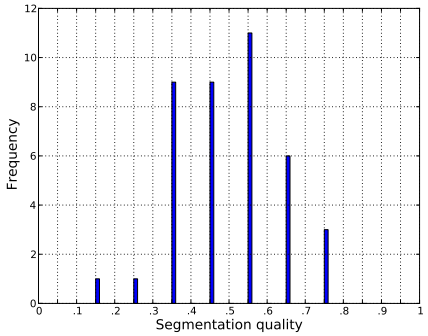
(c) gCH



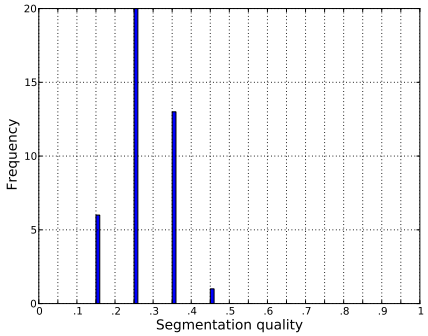
(d) gCO



(e) gTA



(f) gFU



(g) gMB

Figura 5.9: Histogramas das 7 funções de similaridade.

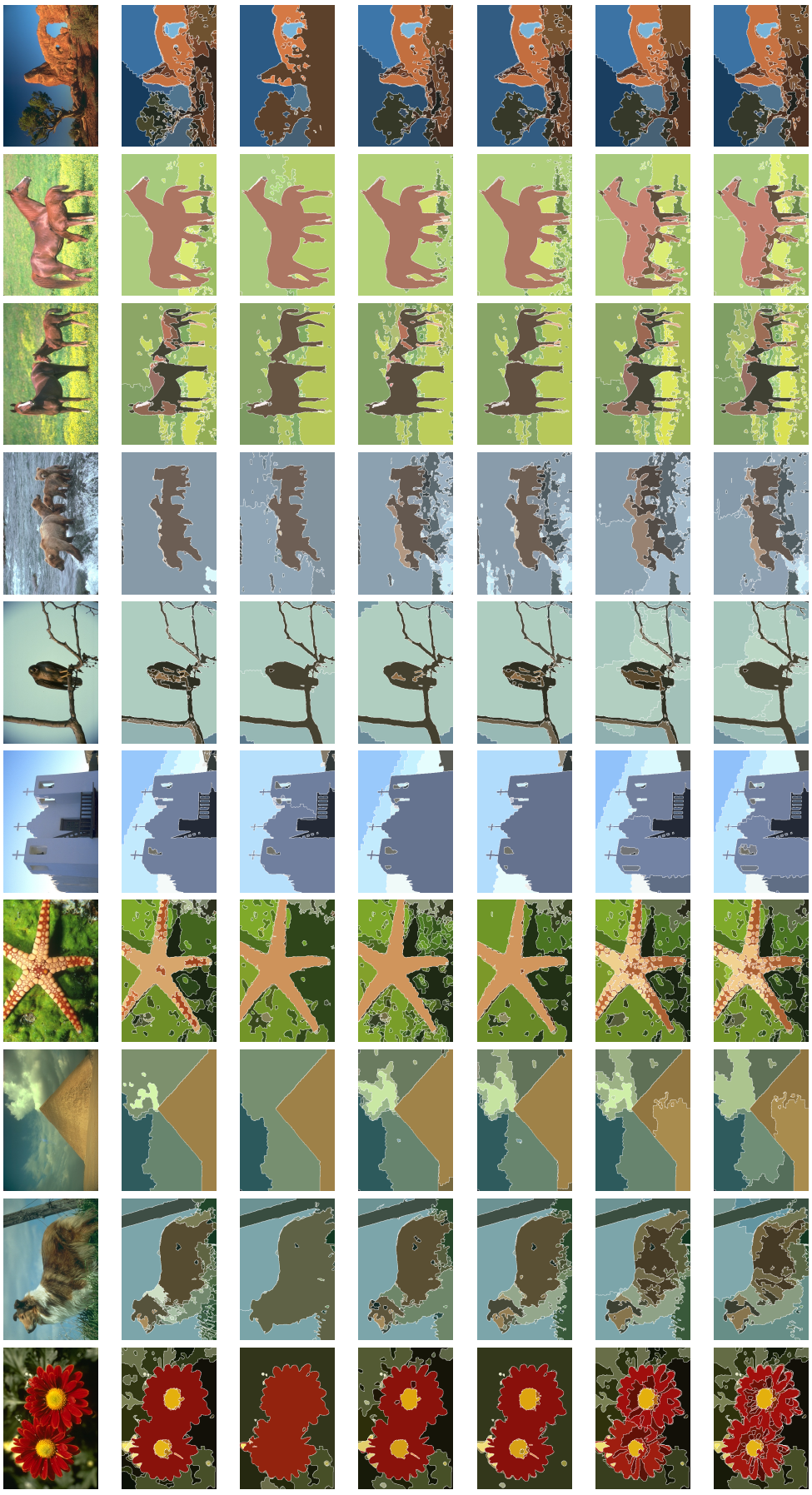


Figura 5.10: Resultados qualitativos obtidos com a função de similaridade gEU. Primeira linha: 10 imagens originais do conjunto BSDS300. Demais linhas, de cima para baixo, respectivamente: resultados para os algoritmos LP, FG, SC1, SC2, ML1 e ML2.

Figura 5.11: Melhores e piores resultados do experimento 2: de cima para baixo melhor (primeira coluna) e pior (terceira coluna) resultado para os algoritmos LP, FG, SC1, SC2, ML1 e ML2. A segunda e quarta coluna apresentam, respectivamente os piores e melhores resultados para cada imagem.

Tabela 5.2: Melhores e piores resultados do experimento 2. Segundo os resultados qualitativos da Figura 5.11, esta tabela apresenta os dados quantitativos dos melhores e piores resultados.

	Melhores					Piores				
Alg.	Imagem	fs	$A_c >$	fs	$A_c <$	Imagem	fs	$A_c <$	fs	$A_c >$
LP	94079.jpg	gEU	0.8546	gEU	0.8546	113016.jpg	gFU	0.1777	gFU	0.1777
FG	247085.jpg	gMH	0.7465	gMH	0.7465	42049.jpg	gMB	0.2443	gMB	0.2443
SC1	42049.jpg	gEU	0.8340	gEU	0.8340	113016.jpg	gMB	0.0336	gMB	0.0336
SC2	42049.jpg	gMH	0.8425	gMH	0.8425	295087.jpg	gMB	0.2065	gMB	0.2065
ML1	247085.jpg	gTA	0.6076	gTA	0.6076	94079.jpg	gMB	0.2306	gMB	0.2306
ML2	42049.jpg	gCH	0.5730	gCH	0.5730	94079.jpg	gFU	0.1757	gFU	0.1757

5.3 Experimento 3

Para este experimento foram empregadas as 300 imagens do conjunto BSDS300. Para todos os 6 algoritmos formulados foram utilizados os seguintes parâmetros: $r = 15$, $\tau = 0.46$, $s = 10$, $k = 7$, $\sigma = 15\%$, $dc = CILAB9$ e $fs = gEU$. O valor de k foi determinado estimando uma aproximação ao número de segmentações das imagens utilizadas. A escolha de $dc = CILAB9$ deve-se ao fato de que esta, na média, produziu um dos melhores resultados, como é visto no experimento 1. A escolha de $fs = gEU$ deve-se ao fato de que esta, na média, produziu um dos melhores resultados, como é visto no experimento 2. Os resultados quantitativos são ilustrados na Tabela 5.3 e a segmentação, para 10 das 300 imagens empregadas, são ilustrados na Figura 5.12.

Os resultados apresentados na Tabela 5.3 indicam que todos os algoritmos de agrupamento em grafos, que seguem a metodologia proposta, obtiveram maior correlação entre a qualidade de agrupamento (índice de silhueta) e a qualidade de segmentação. Isto poderia acontecer em razão a que o cálculo da silhueta é realizado com a informação dos vetores de características dos *superpixels*, além do resultado do agrupamento. Para o caso da modularidade o cálculo da qualidade do agrupamento é realizado somente com a informação do grafo, não tendo desta forma relação direta com a qualidade da segmentação.

Para este experimento é concluível que a qualidade de agrupamento da silhueta tem maior correlação com a qualidade de segmentação, em comparação à qualidade de agrupamento da modularidade.

Tabela 5.3: Resultados da qualidade de segmentação (eixo x) (vs) a qualidade de agrupamento em grafos (eixo y): os pontos nas figuras representam as 300 imagens do banco de imagens de BSDS300. Na primeira linha da tabela, no eixo y das figuras são apresentados os valores de agrupamento de *silhouette index*. Na segunda linha, no eixo y das figuras são apresentados os valores de agrupamento de *modularity*. A qualidade de segmentação, *silhouette index* e *modularity*, respectivamente são calculados seguindo as Equações 4.9, 2.23 e 2.24. p representa o cálculo da Equação 4.11.

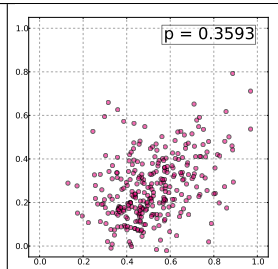
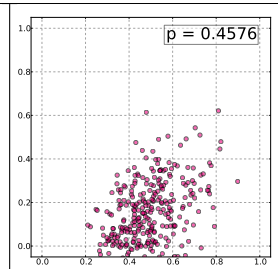
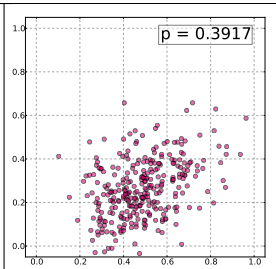
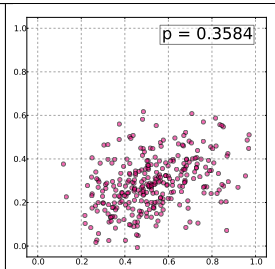
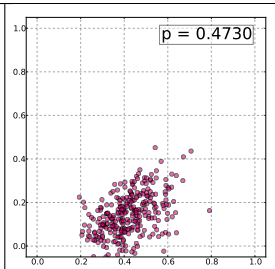
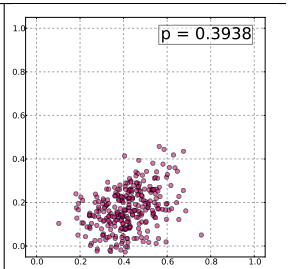
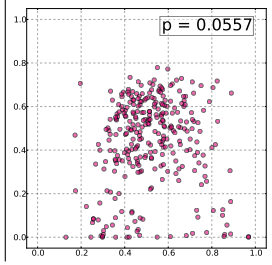
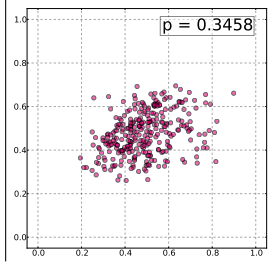
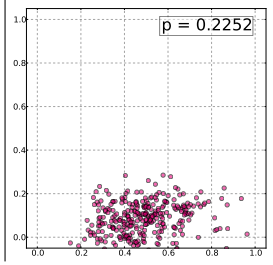
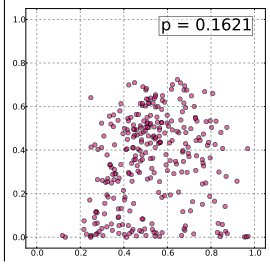
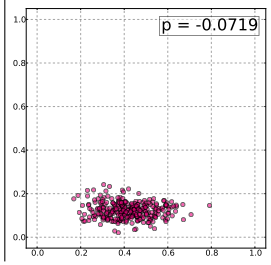
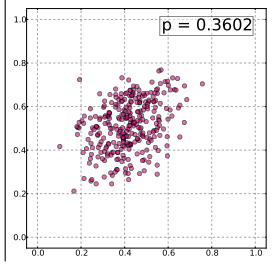
	LP	FG	SC1	SC2	ML1	ML2
S_{index}						
Q						

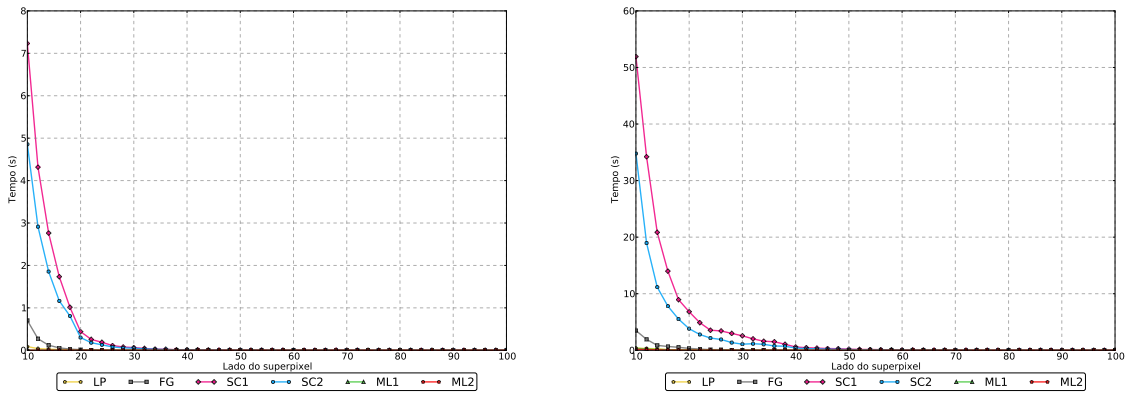
Figura 5.12: Resultados qualitativos. Primeira linha: 10 de 300 imagens originais do conjunto BSDS300: Demais linhas, de cima para baixo, respectivamente: resultados para os algoritmos LP, FG, SC1, SC2, ML1 e ML2.

5.4 Experimento 4

No início deste capítulo foram apresentadas algumas evidências de que existe escalabilidade nos métodos propostos, com exceção dos algoritmos de agrupamento espectral (SC). Particularmente, o gráfico da Figura 5.3**(b)**, mostra como, para valores de $s = 10$ para os quais a qualidade de segmentação é maior, os algoritmos SCs consomem mais tempo de processamento que os demais. No entanto, tais experimentos foram realizados com imagens de tamanho $a\times$ apenas. Este experimento emprega uma imagem de tamanho 481×321 e descreve como o tempo de processamento é afetado à medida que cresce o valor do lado s do *superpixel*. É importante recapitular que o tempo de computação do *superpixel* SLIC não apresenta variação considerável à medida que s cresce, uma vez que utiliza um número fixo de iterações.

A Figura 5.13**(a)** e 5.13**(b)** ilustram o tempo total de processamento dos 6 algoritmos de agrupamentos testados para as imagens de tamanho 400×300 e 1000×667, respectivamente. Pode-se verificar que os algoritmos SC tem um aumento considerável no tempo de processamento, e portanto, não apresentam escalabilidade. Já o algoritmo FG leva cerca de 15 segundos para segmentar a imagem maior, enquanto o tempo de processamento para a imagem menor foi de cerca de 1 segundo. Os algoritmos LP, por um outro lado, pouco foram afetados pela variação de tamanho na imagem e produzem segmentações em menos de 1 segundo.

A Figura 5.14 ilustra a imagem empregada neste experimento sendo 5.14**(a)**, 5.14**(b)** e 5.14**(c)** os *superpixels* de lado 10, 50 e 100, respectivamente.



(a) **(b)**

Figura 5.13: Tempo total de processamento do agrupamento para os 6 algoritmos testados: (a) 10 imagens (400×300), (b) uma imagem (1000×667).

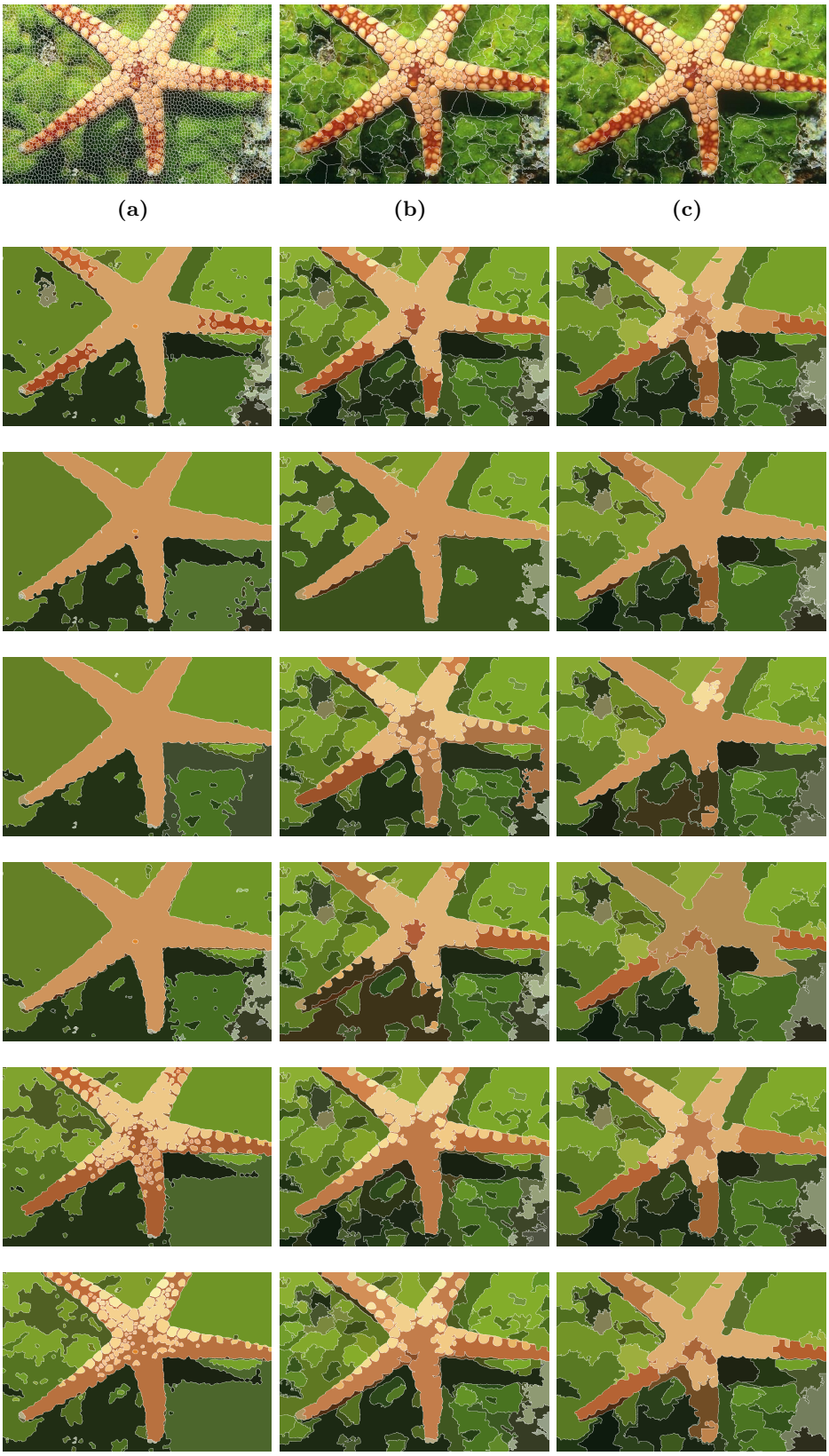


Figura 5.14: Resultados da variação do lado do *superpixel* empregando uma imagem com dimensão 1000×667. As 3 colunas, respectivamente, apresentam valores do lado do *superpixel*: $s = 10$, $s = 50$ e $s = 100$. Primeira linha (a-c), resultados de *superpixels*. Demais linhas, resultados de segmentação para os algoritmos LP, FG, SC1, SC2, ML1 e ML2.

Capítulo

6

Conclusões

Neste trabalho avaliou-se a adequação de seis algoritmos de agrupamentos em grafos para a tarefa de segmentação de imagens. São estes: *Label Propagation* (LP), *Fast Greedy* (FG), *Spectral Clustering* (SC1 e SC2) e *Multi-Level Partition* (ML1 e ML2). Esta adequação pode ser entendida como parte de um estudo que visava responder as seguintes perguntas, como colocadas na introdução desta dissertação que, por clareza, são reproduzidas aqui:

- Quais das categorias de métodos descritas acima são mais adequadas para a segmentação de imagens, ou seja, quais algoritmos produzem segmentações com melhor qualidade?
- Quais métodos apresentam melhor escalabilidade e podem ser aplicados a imagens de grandes dimensões?
- Quais funções de similaridade, empregadas na criação dos grafos, são mais adequadas para a segmentação?
- Existe correlação entre as métricas que avaliam a qualidade do agrupamento em grafos e a qualidade da segmentação?

Pôde-se observar pelos experimentos que os algoritmos de agrupamento espectral (SC1 e SC2) e os de detecção de comunidades (LP e FG) foram os que apresentaram melhores resultados na qualidade de segmentação. Para o caso do SC2, FG e LP, foi empregado um novo parâmetro, o limiar (τ). Mais especificamente, este parâmetro introduzido no algoritmo SC2, quando comparado à implementação encontrada na literatura que não o emprega, e aqui denominado SC1, conduziu a uma leve melhora nos resultados. Esta influência deve-se ao fato de que

o limiar (τ) contribui para uma melhor definição dos agrupamentos, porque ele é responsável por definir a existência de arestas.

A grande quantidade de nós e arestas do grafo é o principal desafio no agrupamento em grafos, quando aplicado ao processamento de imagens. Isso compromete a aplicação de muitos algoritmos para imagens de dimensões maiores e, portanto, suscita questões de escalabilidade para cada um dos algoritmos estudados. Nos experimentos realizados, verificou-se que os métodos mais escaláveis são os baseados em multinível (ML1 e ML2), seguidos de perto pelos algoritmos de identificação de comunidades, FG e LP. Já os algoritmos baseados em agrupamento espectral (SC1 e SC2), embora produzam segmentação de qualidade, não são escaláveis e não se aplicam a imagens de grandes dimensões.

Um dos processos mais importantes na criação do grafo é a determinação do peso das arestas, computado sobre alguma propriedade da imagem. Neste trabalho a propriedade avaliada foi a cor, na forma de vetores de características computados para cada *superpixel*. O peso final é calculado utilizando-se uma função de similaridade. Nesta dissertação foram formuladas 7 funções de similaridade (gEU, gMH, gCH, gCO, gTA, gFU, gMB), testadas para as 6 formulações de algoritmos de agrupamento. Os resultados mostram que não foi possível determinar uma melhor função de similaridade dentre as todas analisadas. No entanto, é possível determinar localmente quais funções de similaridade são mais adequadas para cada um dos 6 algoritmos. Exceção feita à função de similaridade gMB, que não apresentou bons resultados na qualidade de segmentação para nenhum dos 6 algoritmos implementados.

Grande parte dos algoritmos de agrupamento em grafo não foram projetados para lidar com imagens. Parte fundamental no estudo de tais algoritmos, no entanto, é a avaliação da qualidade dos agrupamentos. Neste trabalho procurou-se também investigar a correlação entre tais medidas, particularmente a Modularidade e Silhueta, e a qualidade da segmentação. Estas duas métricas foram comparadas com os resultados da qualidade da segmentação, empregando para isto o Coeficiente de correlação de *Pearson*. Os resultados apontam o índice de silhueta como a métrica que apresentou a maior correlação com a qualidade da segmentação, se comparado com a modularidade. A existência desta correlação pode ser explicada pelo fato de que o índice de silhueta utiliza no cálculo da qualidade do agrupamento, a informação dos vetores de características (informação dos *superpixels*), além do resultado do agrupamento. Por outro lado, a modularidade somente utiliza a informação do resultado do agrupamento do grafo.

Os *superpixels*, como era de se esperar, contribuíram para a redução da cardinalidade do grafo, reduzindo o tempo de processamento do agrupamento. Também foi possível perceber que o tamanho dos *superpixels* é inversamente proporcional à qualidade da segmentação. Para *superpixels* com grandes tamanhos os resultados da qualidade da segmentação tendem a diminuir. Neste sentido, para manter bons resultados da qualidade de segmentação foi possível determinar o lado do *superpixel* em $s = 10$.

Uma última observação importante que se faz necessária é quanto ao processo quantitativo de avaliação da qualidade de segmentação. As segmentações manuais existentes no banco de imagens BSDS300 apresentam uma grande variação de usuário para usuário. Isto é, de certa

forma, esperado dado o carater subjetivo de qualquer processo de segmentação baseado na percepção humana. Neste trabalho todas as segmentações foram consideradas. Provavelmente, se fossem extraídos os *outliers*, os valores de qualidade poderiam ter sido melhor dos que os obtidos neste trabalho.

6.1 Contribuições

Dentre as principais contribuições deste trabalho de mestrado podemos citar:

Proposta de uma abordagem adequada para a segmentação de imagens que combina algoritmos de agrupamento em grafos (*Label Propagation* (LP), *Fast Greedy* (FG), *Spectral Clustering* (SC1 e SC2) e *Multi-Level Partition* (ML1 e ML2)) com *superpixels* em tempo de execução baixo e com qualidade.

Formulação de uma proposta para a criação de grafos em segmentação de imagens. Esta formulação está baseada na existência de um parâmetro denominado limiar (τ). Os resultados demostram que existe uma melhora da qualidade da segmentação de imagens para os algoritmos LP, FG, SC2.

Implementação de descritores de características sobre os *superpixels*, baseados nos 3 primeiros momentos de cor. Em contraste às primeiras aplicações que apenas consideram o nível de cinza ou a tripla RGB de cada pixel.

Propostas de novas funções de similaridade para o cálculo dos pesos das arestas na montagem do grafo. Estas funções foram propostas em base à função de similaridade Gaussiana. Nela foram empregadas as seguintes funções de distância além da Euclidiana: *Manhattan*, *Chebyshev*, *1-Coseno*, *1-Tanimoto*, *1-Fu* e *Mahalanobis*.

Estudo da correlação entre a qualidade de segmentação e a qualidade de agrupamento. Esta informação pode ser de grande importância para pesquisadores que realizam trabalhos com agrupamento em grafos aplicado à segmentação de imagens. Os resultados da dissertação revelaram que a qualidade de agrupamento calculada com a métrica de índice de silhueta apresenta maior correlação com a qualidade da segmentação. Desta forma a medida de silhueta é um indicativo correto da qualidade de segmentação.

6.2 Limitações

Os resultados em uma grande parte das imagens não apresentam excelentes resultados porque os descritores de características são baseados somente na cor. A extração de atributos de textura poderia contribuir para aumentar a qualidade da segmentação sobre as imagens do banco BSDS300.

A proposta desta dissertação está baseada na criação do grafo mediante o limiar (τ). Isto gera mais um parâmetro, e a dependência deste parâmetro também influi nos resultados da qualidade de segmentação. Na presente dissertação não foi proporcionado um procedimento automático para calcular este valor de limiar (τ).

6.3 Trabalhos Futuros

Uma das tarefas futuras é a implementação de descritores de características baseados em textura. Estes descritores serão implementados após a computação dos *superpixels*, levando em consideração a vizindade que estes possam apresentar.

Também pretende-se trabalhar em uma formulação de um processo adaptativo para determinar automaticamente o valor do limiar, procurando sempre uma boa qualidade nos resultados da segmentação de imagens.

Outro trabalho interessante é a formulação de um algoritmo para avaliar a influência da criação do grafo na qualidade de agrupamento do grafo, portanto na qualidade da segmentação de imagens.

Referências Bibliográficas

Abreu, N. *Teoria espectral dos grafos: Um hibrido entre a algebra linear e a matematica discreta e combinatoria com origens na quimica quantica.* Sociedade Brasileira de Matematica Aplicada e Computacional, 1-10 p., 2005.
Disponível em `http://www.sbmac.org.br/tema/seletas/docs/v6/01-conc-A-XXVII.pdf`

Achanta, R.; Shaji, A.; Smith, K.; Lucchi, A.; Fua, P.; Susstrunk, S. Slic superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, v. PP, n. 99, p. 1, 2012.

Achanta, R.; Shaji, A.; Smith, K.; Lucchi, A.; Fua, P.; Süsstrunk, S. *SLIC Superpixels.* Relatório Técnico, EPFL, 2010.

Aggarwal, C. C.; Wang, H. A survey of clustering algorithms for graph data. In: Elmagarmid, A. K.; Aggarwal, C. C.; Wang, H., eds. *Managing and Mining Graph Data*, v. 40 de *Advances in Database Systems*, Springer US, p. 275–301, 10.1007/978-1-4419-6045-0_9, 2010.
Disponível em `http://dx.doi.org/10.1007/978-1-4419-6045-0_9`

Almeida, H.; Guedes, D.; Meira, W.; Zaki, M. Is there a best quality metric for graph clusters? In: Gunopulos, D.; Hofmann, T.; Malerba, D.; Vazirgiannis, M., eds. *Machine Learning and Knowledge Discovery in Databases*, v. 6911 de *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, p. 44–59, 10.1007/978-3-642-23780-5_13, 2011.
Disponível em `http://dx.doi.org/10.1007/978-3-642-23780-5_13`

Arbelaez, P.; Maire, M.; Fowlkes, C.; Malik, J. From contours to regions: An empirical evaluation. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 2009, p. 2294 –2301.

Chung, F. R. K. *Spectral graph theory*, v. 92 de *CBMS Regional Conference Series in Mathematics.* Published for the Conference Board of the Mathematical Sciences, Washington, DC, xii+207 p., 1997.

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

CLAUSET, A.; NEWMAN, M. E. J.; MOORE, C. Finding community structure in very large networks. *Phys. Rev. E*, v. 70, p. 066111, 2004.
Disponível em `http://link.aps.org/doi/10.1103/PhysRevE.70.066111`

COMANICIU, D.; MEER, P. Mean shift: a robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, v. 24, n. 5, p. 603 –619, 2002.

COUR, T.; BENEZIT, F.; SHI, J. Spectral segmentation with multiscale graph decomposition. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2005, p. 1124 – 1131 vol. 2.

CREVIER, D. Image segmentation algorithm development using ground truth image data sets. *Comput. Vis. Image Underst.*, v. 112, n. 2, p. 143–159, 2008.
Disponível em `http://dx.doi.org/10.1016/j.cviu.2008.02.002`

CVETKOVIC, D. M.; DOOB, M.; SACHS, H. *Spectra of graphs: Theory and applications*. Academic Press, New York, 1-10 p., 1980.

DHILLON, I. S.; GUAN, Y.; KULIS, B. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, v. 29, p. 1944–1957, 2007.
Disponível em `http://portal.acm.org/citation.cfm?id=1313055.1313291`

FAN, N.; PARDALOS, P. Multi-way clustering and biclustering by the ratio cut and normalized cut in graphs. *Journal of Combinatorial Optimization*, p. 1–28, 10.1007/s10878-010-9351-5, 2010.
Disponível em `http://dx.doi.org/10.1007/s10878-010-9351-5`

FELZENSZWALB, P.; HUTTENLOCHER, D. Efficient graph-based image segmentation. *International Journal of Computer Vision*, v. 59, p. 167–181, 10.1023/B:VISI.0000022288.19776.77, 2004.
Disponível em `http://dx.doi.org/10.1023/B:VISI.0000022288.19776.77`

FOWLKES, C.; BELONGIE, S.; CHUNG, F.; MALIK, J. Spectral grouping using the nystrom method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, v. 26, n. 2, p. 214 –225, 2004.

FULKERSON, B.; VEDALDI, A.; SOATTO, S. Class segmentation and object localization with superpixel neighborhoods. In: *Computer Vision, 2009 IEEE 12th International Conference on*, 2009, p. 670 –677.

GE, F.; WANG, S.; LIU, T. Image-segmentation evaluation from the perspective of salient object extraction. In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 2006, p. 1146 – 1153.

HAGEN, L.; KAHNG, A. New spectral methods for ratio cut partitioning and clustering. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, v. 11, n. 9, p. 1074 –1085, 1992.

HENDRICKSON, B.; LELAND, R. A multi-level algorithm for partitioning graphs. In: *Supercomputing, 1995. Proceedings of the IEEE/ACM SC95 Conference*, 1995.

KARYPIS, G.; KUMAR, V. Multilevel k-way partitioning scheme for irregular graphs. *J. Parallel Distrib. Comput.*, v. 48, p. 96–129, 1998.
Disponível em `http://dx.doi.org/10.1006/jpdc.1997.1404`

KAUFHOLD, J.; COLLINS, R.; HOOGS, A.; RONDOT, P. Recognition and segmentation of scene content using region-based classification. In: *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, 2006, p. 755 –760.

KERNIGHAN, B. W.; LIN, S. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell system technical journal*, v. 49, n. 1, p. 291–307, 1970.

KONG, T. F. *Multilevel spectral clustering : Graph partitions and image segmentation.* Dissertação de Mestrado, Massachusetts Institute of Technology, 2008.
Disponível em `http://dspace.mit.edu/handle/1721.1/45275`

KUNEGIS, J.; LOMMATZSCH, A.; BAUCKHAGE, C. Alternative similarity functions for graph kernels. In: *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, 2008, p. 1 –4.

LEVINSHTEIN, A.; STERE, A.; KUTULAKOS, K. N.; FLEET, D. J.; DICKINSON, S. J.; SIDDIQI, K. Turbopixels: Fast superpixels using geometric flows. *IEEE Trans. Pattern Anal. Mach. Intell.*, v. 31, p. 2290–2297, 2009.
Disponível em `http://portal.acm.org/citation.cfm?id=1638615.1639288`

MAIER, M.; LUXBURG, U.; HEIN, M. How the result of graph clustering methods depends on the construction of the graph. *CoRR*, v. abs/1102.2075, 2011.

MALISIEWICZ, T.; EFROS, A. A. Improving spatial support for objects via multiple segmentations. In: *British Machine Vision Conference (BMVC)*, 2007.

MARTIN, D.; FOWLKES, C.; TAL, D.; MALIK, J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, 2001, p. 416 –423 vol.2.

Moore, A.; Prince, S.; Warrell, J.; Mohammed, U.; Jones, G. Superpixel lattices. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, p. 1 –8.

Nascimento, M. C. V. *Metaheuristicas para o problema de agrupamento de dados em grafo*. Tese de Doutoramento, Universidade de Sao Paulo, 2010.

Newman, M. E. J.; Girvan, M. Finding and evaluating community structure in networks. *Phys. Rev. E*, v. 69, n. 2, p. 026113, 2004.

Nishida, H.; Nguyen, T. Optimal client-server assignment for internet distributed systems. In: *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, 2011, p. 1 –6.

Raghavan, U. N.; Albert, R.; Kumara, S. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E*, v. 76, p. 036106, 2007.
Disponível em `http://link.aps.org/doi/10.1103/PhysRevE.76.036106`

Rasmussen, C. Superpixel analysis for object detection and tracking with application to uav imagery. In: Bebis, G.; Boyle, R.; Parvin, B.; Koracin, D.; Paragios, N.; Tanveer, S.-M.; Ju, T.; Liu, Z.; Coquillart, S.; Cruz-Neira, C.; Müller, T.; Malzbender, T., eds. *Advances in Visual Computing*, v. 4841 de *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, p. 46–55, 2007.
Disponível em `http://dx.doi.org/10.1007/978-3-540-76858-6_5`

Ren, X.; Malik, J. Learning a classification model for segmentation. In: *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, 2003, p. 10 –17 vol.1.

Rodrigues, F. A.; Ferraz de Arruda, G.; da Fontoura Costa, L. A Complex Networks Approach for Data Clustering. *ArXiv e-prints*, 2011.

Sakai, T.; Imiya, A. Fast spectral clustering with random projection and sampling. In: *Proceedings of the 6th International Conference on Machine Learning and Data Mining in Pattern Recognition*, Berlin, Heidelberg: Springer-Verlag, 2009, p. 372–384.
Disponível em `http://dx.doi.org/10.1007/978-3-642-03070-3_28`

Sanders, P.; Schulz, C. Engineering multilevel graph partitioning algorithms. *CoRR*, v. abs/1012.0006, 2010.
Disponível em `http://arxiv.org/abs/1012.0006`

Santo; Fortunato Community detection in graphs. *Physics Reports*, v. 486, n. 3-5, p. 75 – 174, 2010.
Disponível em `http://dx.doi.org/10.1016/j.physrep.2009.11.002`

Schaeffer, S. E. Graph clustering. *Computer Science Review*, v. 1, n. 1, p. 27 – 64, 2007.
Disponível em `http://www.sciencedirect.com/science/article/B8JDG-4PBG1S7-1/2/6537f3d1ffbf391086c60dbeba874b13`

Shi, J.; Malik, J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22, p. 888–905, 1997.

Shi, J.; Malik, J. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, v. 22, n. 8, p. 888 –905, 2000.

Sun, F.; He, J.-P. A normalized cuts based image segmentation method. *Information and Computing Science, International Conference on*, v. 2, p. 333–336, 2009.

Tan, P.-N.; Steinbach, M.; Kumar, V. *Introduction to data mining.* 1 ed. Addison Wesley, 2005.

Tolliver, D.; Miller, G. Graph partitioning by spectral rounding: Applications in image segmentation and clustering. In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 2006, p. 1053 – 1060.

Tung, F.; Wong, A.; Clausi, D. A. Enabling scalable spectral clustering for image segmentation. *Pattern Recogn.*, v. 43, p. 4069–4076, 2010.
Disponível em `http://dx.doi.org/10.1016/j.patcog.2010.06.015`

Vedaldi, A.; Soatto, S. Quick shift and kernel methods for mode seeking. In: *In European Conference on Computer Vision, volume IV*, 2008, p. 705–718.

Veksler, O.; Boykov, Y.; Mehrani, P. Superpixels and supervoxels in an energy optimization framework. In: Daniilidis, K.; Maragos, P.; Paragios, N., eds. *Computer Vision - ECCV 2010*, v. 6315 de *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, p. 211–224, 10.1007/978-3-642-15555-0_16, 2010.
Disponível em `http://dx.doi.org/10.1007/978-3-642-15555-0_16`

Vincent, L.; Soille, P. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, v. 13, n. 6, p. 583 –598, 1991.

Von Luxburg, U. A tutorial on spectral clustering. *Statistics and Computing*, v. 17, p. 395–416, 10.1007/s11222-007-9033-z, 2007.
Disponível em `http://dx.doi.org/10.1007/s11222-007-9033-z`

Wertheimer, M. *Laws of organization in perceptual forms* Routledge and Kegan Paul, p. 71–88, 1938.

Wu, Z.; Leahy, R. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, v. 15, n. 11, p. 1101 –1113, 1993.

Xiang, S.; Nie, F.; Zhang, C. Learning a mahalanobis distance metric for data clustering and classification. *Pattern Recognition*, v. 41, n. 12, p. 3600 – 3612, 2008.

Disponível em http://www.sciencedirect.com/science/article/pii/S0031320308002057

YAN, D.; HUANG, L.; JORDAN, M. I. Fast approximate spectral clustering. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, New York, NY, USA: ACM, 2009, p. 907–916 (*KDD '09*,).
Disponível em http://doi.acm.org/10.1145/1557019.1557118

ZENG, G.; WANG, P.; WANG, J.; GAN, R.; ZHA, H. Structure-sensitive superpixels via geodesic distance. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011, p. 447 –454.

Apêndice

A

Imagens Utilizadas nos Experimentos

As Figuras A.1, A.2, A.3 e A.4 apresentam as imagens utilizadas nesta dissertação e suas respectivas segmentações manuais. As Figuras A.1 e A.2 foram empregadas nos experimentos 1 e 2. As Figuras A.3 e A.4 foram utilizadas no experimento 3.

Figura A.1: Imagens empregadas nos experimentos 1 e 2. Primeira coluna (imagem original) e demais (segmentações manuais). De cima para baixo(como empregado no conjunto BSDS300: 124084.jpg, 247085.jpg, 299091.jpg, 12003.jpg e 24063.jpg.

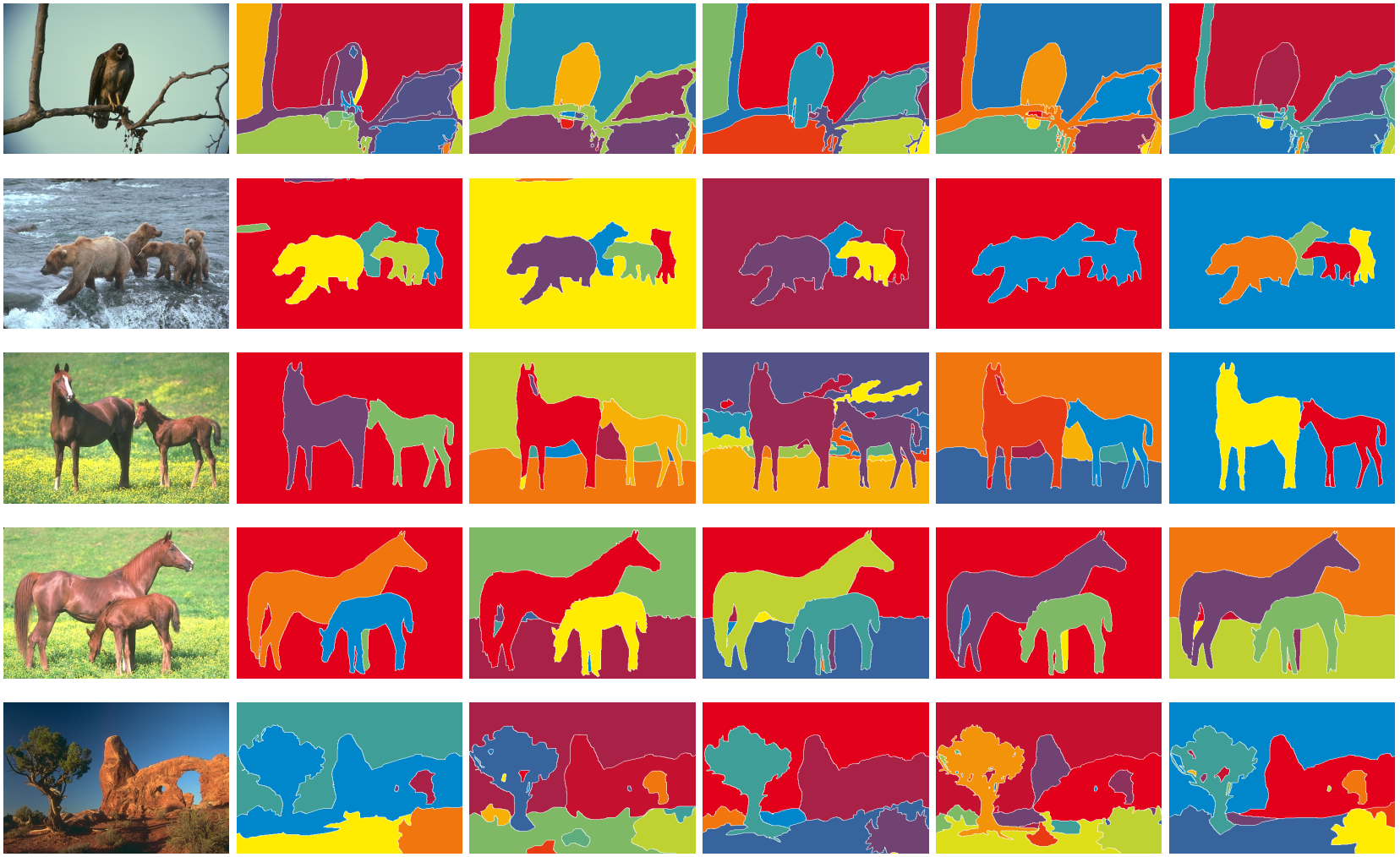


Figura A.2: Imagens empregadas nos experimentos 1 e 2. Primeira coluna (imagem original) e demais (segmentações manuais). De cima para baixo(como empregado no conjunto BSDS300: 42049.jpg, 94079.jpg, 113016.jpg, 113044.jpg e 295087.jpg

Figura A.3: Imagens empregadas no experimento 3. Primeira coluna (imagem original) e demais (segmentações manuais). De cima para baixo(como empregado no conjunto BSDS300: 35010.jpg, 67079.jpg, 353013.jpg, 118035.jpg e 253036.jpg

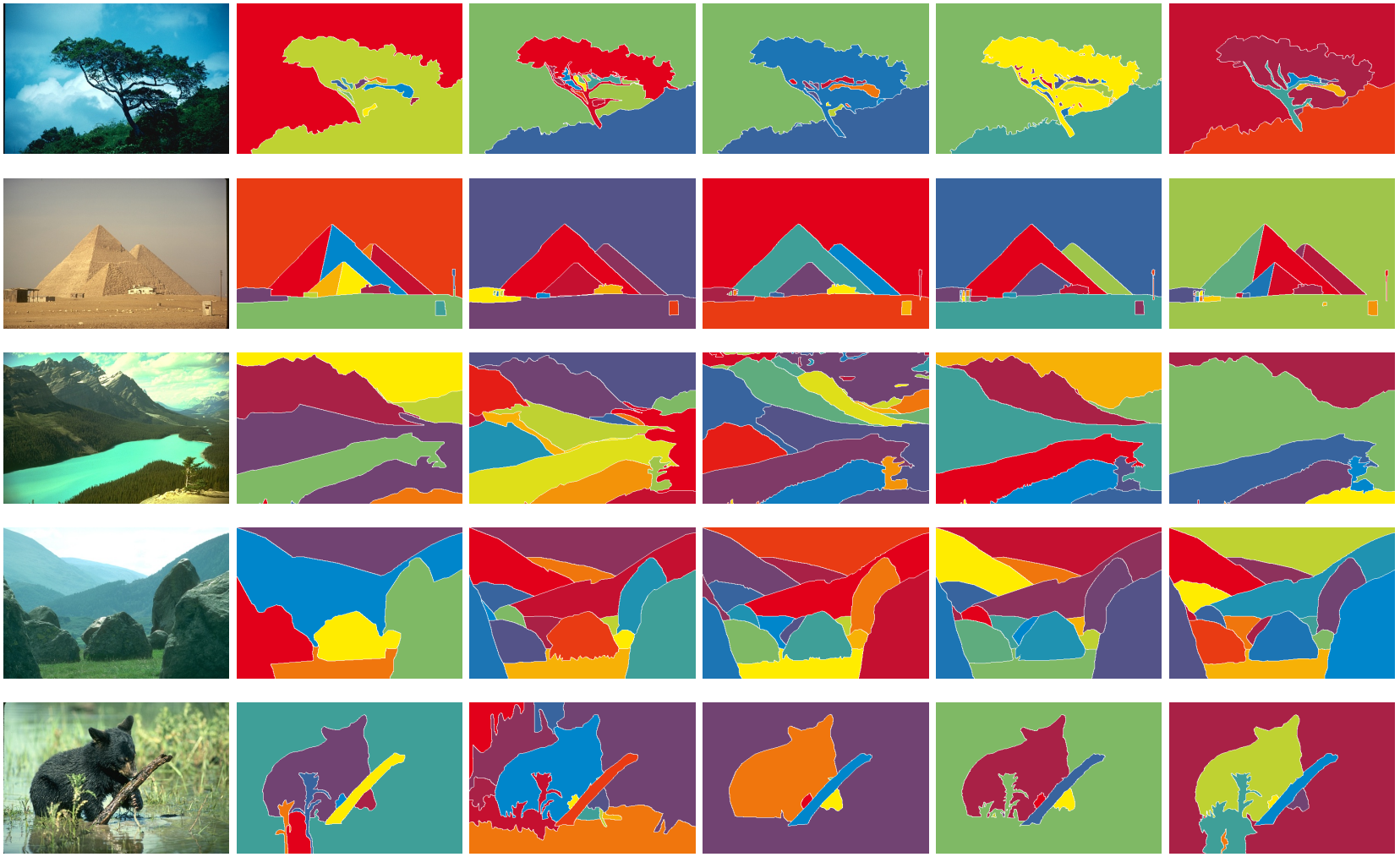


Figura A.4: Imagens empregadas no experimento 3. Primeira coluna (imagem original) e demais (segmentações manuais). De cima para baixo(como empregado no conjunto BSDS300: 147091.jpg, 161062.jpg, 176035.jpg, 241004.jpg e 159091.jpg

Apêndice

B

Algoritmos Auxiliares

Algoritmo para computação de componentes conexas.

Entrada:
I: imagem; $l[i]$, $i \in n$; s: número de segmentos; W: largo da imagem; H: alto da imagem.
Saída:
$l'[i]$, $i \in n$, cs': novo número de segmentos, $|cs'| \geq |s|$.

```
1  início
2      def(m = bool[n](false))                  /* Arranjo para definir a visita de um pixel */
3      def(nx = int[8](1,1,0,−1,−1,−1,0,1))          /* 8 vizinhos de um pixel (eixo x) */
4      def(ny = int[8](0,−1,−1,−1,0,1,1,1))          /* 8 vizinhos de um pixel (eixo y) */
5      def(cs = 0)                                                /* Contador de regiões */
6      def(v = vector())      /* Vetor para salvar os pixel conexos de uma mesma região */
7      def(d = queue())                 /* Fila para salvar os pixel conexos visitados */
8      para y = 0 to H faça
9          para x = 0 to W faça
10             i = y∗W + x
11             e = l[i]
                                                          /* Se o pixel não foi visitado */
12             se m[i] == false então
13                 v.add(i)
14                 d.push(i)
15                 m[i] = true
                                                        /* Enquanto a fila esteia cheia */
16                 enquanto NOT d.isEmpty() faça
17                     id = d.pop()
18                     xi = I.x(id)
19                     yi = I.y(id)
                           /* Para os 8 vizinhos de cada pixel da mesma região conexa */
20                     para k = 0 to 8 faça
21                         ex = xi + nx[k]
22                         ey = yi + ny[k]
23                         j = ey∗W + ex
24                         se ex ∈ {0,W} AND ey ∈ {0,H} AND l[j] == e AND m[j] == false então
25                             m[j] = true
26                             d.push(j)
27                             v.add(j)
28                         fim
29                     fim
30                 fim
                                                /* Para todos os pixels de uma região conexa */
31                 para j = 0 to v.size() faça
32                     id = v[j]
33                     l[id] = cs                          /* Atualizando a etiqueta da região.  */
34                 fim
35                 v.clear()
36                 cs = cs + 1
37             fim
38         fim
39     fim
40 fim
```

Algoritmo B.1: Regiões com *pixels conexos*