Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Thais Lima de Sousa

# Algoritmos de segmentação de imagens baseados em grafos

São Paulo
Novembro de 2017

# Algoritmos de segmentação de imagens baseados em grafos

Monografia final da disciplina
MAC0499 – Trabalho de Formatura Supervisionado.

Supervisor: Prof. Dr. Roberto Hirata Junior

São Paulo
Novembro de 2017

# Agradecimentos

Agradeço aos professores Roberto Hirata e Nina Hirata pela dedicação e disponibilidade para direcionar este trabalho, me orientando ao longo do ano, principalmente nos momentos em que achava que não conseguiria finalizá-lo.

Agradeço ao professor Ronaldo Hashimoto, por ter me orientado durante a Iniciação Científica, encorajando meu interesse na área.

Agradeço a minha família, pelo amor e apoio durante todos esses anos, por me incentivar e permitir que eu corresse atrás dos meus sonhos.

Agradeço aos meus amigos, que sempre estiveram ao meu lado. Em especial, agradeço meus irmãos e irmãs da ABU (Aliança Bíblica Universitária) por momentos semanais de comunhão e oração.

Agradeço especialmente ao Bruno, por encher minha vida de luz e amor, por ser minha fortaleza tanto em tempos de tristeza quanto de alegria.

Agradeço a Deus por tudo.

# Resumo

Segmentação de imagens é o processo de particionar o domínio de uma imagem em múltiplas regiões que contenham informações significativas para um determinado propósito. Embora seja uma tarefa fácil para seres humanos, é bastante difícil para o computador. Outro problema é o elevado custo computacional quando se trabalha em nível de *pixels*, pois uma imagem, mesmo que em resolução moderada, é composta por muitos deles, e eles não são entidades naturais para representar regiões homogêneas da imagem. O agrupamento de *pixels* em *superpixels* (regiões compactas e conexas) visa diminuir a complexidade da tarefa de processamento de imagens sem perda de informações locais desses elementos. O objetivo deste trabalho é estudar dois algoritmos de segmentação de imagens, que particionam seu domínio com base na Árvore Geradora de Custo Mínimo do grafo de *superpixels*, construído na etapa de pré-processamento pelo algoritmo *Simple Linear Iterative Clustering*. Os experimentos realizados visam comparar os métodos de segmentação, os resultados para diferentes valores de parâmetros, a diferença de desempenho entre segmentar a imagem a partir de *pixels* e de *superpixels* e o comportamento desses fatores diante de diferentes imagens.

**Palavras-chave:** processamento de imagens, segmentação, superpixels, agrupamento, algoritmos em grafos.

# Abstract

Image segmentation is the process of partitioning the domain of an image into multiple regions containing information that is meaningful for a particular purpose. Although it is an easy task for human beings, it is quite difficult for the computer. Another problem is the high computational cost when working at pixels level, because an image, even in moderate resolution, has a lot of pixels, and they are not natural entities to represent homogeneous regions of the image. Through clustering pixels into superpixels (compact and connected regions), we aim to reduce the complexity of the image processing task without loss of local information from those elements. The goal of this work is to study two segmentation algorithms that partition the image using the Minimum Spanning Tree of the superpixel graph, built in the preprocessing stage using the *Simple Linear Iterative Clustering* algorithm. The performed experiments aims to compare the segmentation methods, the results for different parameters values, the performance difference between segmentation based on pixels and on superpixels and the behavior of all these factors considering different images.

**Keywords:** image processing, segmentation, superpixels, clustering, graph algorithms.

# Sumário

# Lista de Abreviaturas

| | |
|---|---|
| MST | Árvore Geradora Mínima ou de Custo Mínimo (*Minimum Spanning Tree*) |
| RAG | Grafo de Regiões Adjacentes (*Region Adjacency Graph*) |
| SLIC | Agrupamento Iterativo Linear Simples (*Simple Linear Iterative Clustering*) |
| CIE | Comissão Internacional de Iluminação (*International Commission on Illumination*) |
| CIELAB | Espaço de cor definido pela CIE (*CIELab color space*) |
| FH | Algoritmo de Felzenszwalb & Huttenlocher |
| GUI | Algoritmo de Guimarães et al. |

# Lista de Figuras

# Capítulo 1

# Introdução

## 1.1 O problema de segmentação de imagens

O problema de segmentação consiste em particionar o conjunto de pontos de uma imagem de forma que cada parte (ou segmento) na imagem corresponda a um componente de interesse. O processo de segmentação pode ser realizado, por exemplo, por meio de divisões do seu domínio em regiões menores ou de aglomerações dos pontos desse domínio, até que as características dos pontos de cada região sejam similares em relação a cor, textura, forma, tamanho ou alguma outra característica. O nível de detalhes esperado no particionamento depende do problema em questão. Considere, por exemplo, um sistema de inspeção de peças eletrônicas, cujo objetivo é analisar imagens de produtos e identificar a presença ou ausência de anomalias, como falta de algum componente ou uma parte quebrada de algum deles. É suficiente uma segmentação com nível de detalhes que permita identificar essas componentes.

A segmentação de imagens digitais é uma tarefa importante para diversas áreas, como análise de imagens médicas, reconhecimento facial ou de impressões digitais, recuperação de imagem baseada em conteúdo, dentre outras formas de detecção e classificação de objetos. Ela possibilita a extração de informações relevantes para um determinado problema. A figura 1.1 mostra os seguintes exemplos: detecção de peças de vestuário para modificação e composição da imagem; detecção de rua, calçada, carros, pessoas e objetos em tempo real pelo sistema de um veículo robótico (conduzido sem o auxílio de um ser humano); detecção computadorizada de tumores no cérebro em imagens de exames médicos.

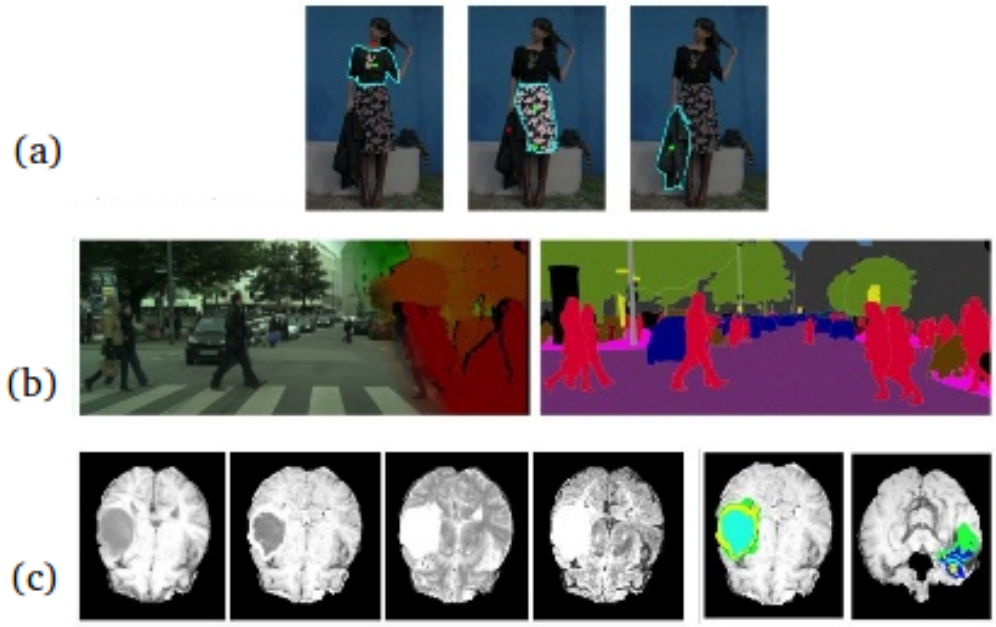



**Figura 1.1:** *Uso de segmentação de imagens em aplicações reais*
(a) edição e composição de imagens; (b) veículos autônomos; (c) análise de imagens médicas.

Entretanto, essa tarefa pode se tornar extremamente complicada devido à ampla variabilidade de cores, texturas, intensidades de iluminação, contornos e outras características presentes em uma imagem. Outro problema é a grande quantidade de operações computacionais quando se trabalha em nível de *pixels* da imagem.

Visto que, geralmente, uma imagem contém pequenas regiões apresentando mesmas características, uma forma de reduzir o custo computacional de operações é agrupar *pixels* em *superpixels* através de supersegmentação (*over-segmentation*). Veja o exemplo na figura 1.2. As motivações para realizar a segmentação a partir de *superpixels* são: (1) *pixels* não são entidades naturais, são apenas uma consequência da representação discreta da imagem; (2) o número de *pixels* é grande até mesmo em imagens com resolução moderada, fazendo com que otimizações em nível de *pixels* seja intratável.

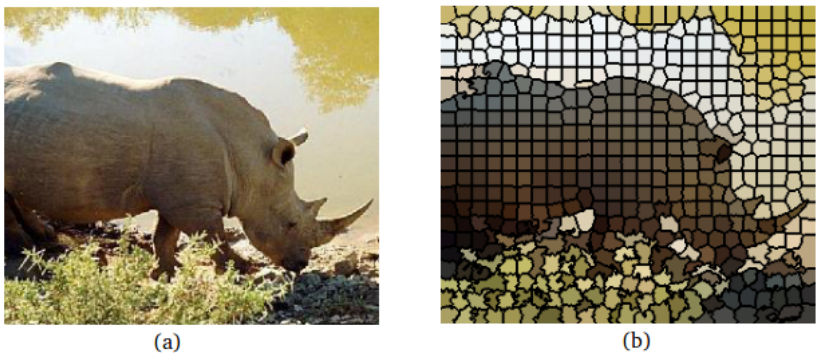



**Figura 1.2:** *Imagem e seus superpixels*
(a) imagem original, composta por 76500 *pixels*; (b) imagem formada a partir de (a), com 500 *superpixels* (1/153 do número de *pixels*).

Para gerar resultados com qualidade, queremos trabalhar com *superpixels* que preservem informações locais, coerentes e estruturais necessárias à segmentação na escala de interesse,

pois sua representação pode afetar positivamente ou negativamente o desempenho dos algoritmos de segmentação, caso não respeitem os limites dos objetos na imagem.

Assim como um modelo de segmentação desenvolvido por Ren e Malik (2003), queremos utilizar a supersegmentação como um pré-processamento para segmentar imagens. Neste trabalho, os algoritmos de segmentação estudados utilizam um arcabouço de grafos com pesos nas arestas. Cada *pixel* ou *superpixel* é representado por um vértice no grafo, elementos vizinhos são conectados por arestas e os pesos nas arestas são definidos por uma função de diferença de similaridade entre eles (obtida a partir da imagem). Os algoritmos estudados segmentam as imagens com base na Árvore Geradora de custo Mínimo (MST) do grafo de entrada, através da remoção de arestas.

## 1.2 Objetivos

Neste trabalho, foram estudados dois métodos de segmentação de imagens: o método desenvolvido por Felzenszwalb e Huttenlocher (2004) e o método hierárquico de Guimaraes *et al.* (2012). Eles foram aplicados em imagens a partir de *pixels* e de *superpixels*, estes gerados pelo algoritmo *Simple Linear Iterative Clustering* (SLIC) de Achanta *et al.* (2010).

Para realizar as segmentações, esses dois métodos baseiam-se em um parâmetro chamado *Escala de Observação (k)*. Altos valores de $k$ fazem com que a segmentação produza componentes de grandes tamanhos, ou seja, esse parâmetro refere-se à quantidade de regiões que serão observadas.

A diferença entre o método de Felzenszwalb e o de Guimarães é que este obtém a segmentação a partir de uma coleção de segmentações sob diferentes níveis de detalhes. Segmentações em níveis de detalhes mais grossos são obtidas pela fusão de regiões de segmentações em níveis de detalhes mais finos, de forma a não violar o *Princípio de Causalidade* e o *Princípio de Localização*, diferentemente do método de Felzenszwalb, que pode violar esses princípios.

O objetivo deste trabalho é analisar e comparar as segmentações obtidas pelo método de Felzenszwalb e pelo método de Guimarães, a partir de *pixels* e de *superpixels*, bem como os resultados dos métodos para variação de parâmetros, aplicados a diferentes imagens.

## 1.3 Organização do trabalho

No capítulo 2 são definidos conceitos básicos sobre imagens e grafos, necessários à compreensão dos algoritmos estudados. A construção de *superpixels* é apresentada no capítulo 3 e os algoritmos de segmentação, no capítulo 4. Os experimentos, discussões e conclusões são abordados nos capítulos 5 e 6.

# Capítulo 2

# Fundamentos

Este capítulo apresenta conceitos básicos relacionados a grafos e imagens.

## 2.1 Conceitos básicos sobre grafos

### 2.1.1 Vértices e arestas

Um grafo $G(V, E)$ é uma estrutura de dados definida por um par de conjuntos: um conjunto $V$ de vértices ou nós e um conjunto $E$ de arestas. Cada aresta $\{v_i, v_j\}$ é formada por um par não ordenado de vértices $v_i$ e $v_j$, com $v_i \neq v_j$.

Podemos associar a cada aresta $\{v_i, v_j\} \in E$ um peso correspondente $w(\{v_i, v_j\}) \geq 0$ relacionado a um custo ou a alguma medida de dissimilaridade entre os elementos vizinhos $v_i$ e $v_j$.

### 2.1.2 Subgrafo

Um subgrafo de um grafo $G(V, E)$ é qualquer grafo $H(V', E')$ tal que $V' \subset V$ e $E' \subset E$.

Um subgrafo $H$ de um grafo $G(V, E)$ é induzido se toda aresta $e \in E$ que tem ambas as pontas em $H$ também é aresta de $H$.

### 2.1.3 Caminho, circuito e conexidade

Um caminho é um grafo da forma $G(V, E)$ onde $V = \{v_1, v_2, ..., v_n\}$ e $E = \{\{v_i, v_{i+1}\} : 1 \leq i \leq n-1\}$.

Dado $n \geq 3$, um circuito de $n$ vértices é um grafo da forma $G(V, E)$ onde $V = \{v_1, v_2, ..., v_n\}$ e $E = \{\{v_i, v_{i+1}\} : 1 \leq i \leq n-1\} \cup \{\{v_n, v_1\}\}$. Dizemos que um grafo é acíclico quando ele não tem nenhum subgrafo que é um circuito.

Um grafo é conexo se, para quaisquer dois vértices $v_i, v_j \in V$, existe um caminho cujos extremos são $v_i$ e $v_j$.

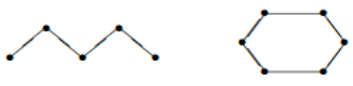


**Figura 2.1:** *Um caminho e um circuito*

### 2.1.4 Árvore Geradora de Custo Mínimo (MST)

Uma árvore é um grafo acíclico conexo. Uma Árvore Geradora de Custo Mínimo (*Minimum Spanning Tree* ou MST) de um grafo $G(V, E)$ é uma árvore $T(V, E_t)$, onde $E_t \subset E$ contém aretas que conectam todos os vértices em $V$ com o menor custo possível, ou seja, a soma dos pesos de todas as arestas tem valor mínimo.

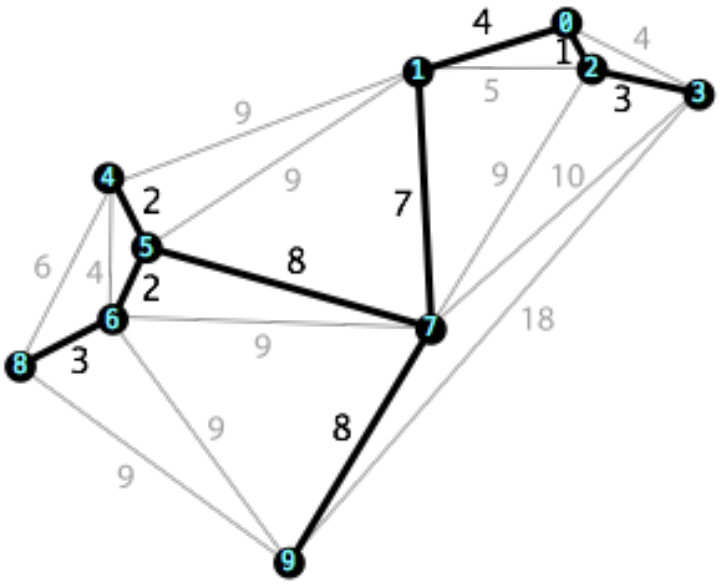



**Figura 2.2:** *Exemplo de Árvore Geradora de custo mínimo (MST)*
Em negrito, estão marcadas as arestas da MST do grafo.

### 2.1.5 Corte

Um grafo $G = (V, E)$ pode ser particionado em dois conjuntos disjuntos $A$ e $B$, tais que $A \cup B = V$, por meio da remoção de arestas ligando os dois conjuntos. O valor do custo de um corte é definido por:

$$cut(A, B) = \sum_{v_i \in A, v_j \in B} w(v_i, v_j)$$

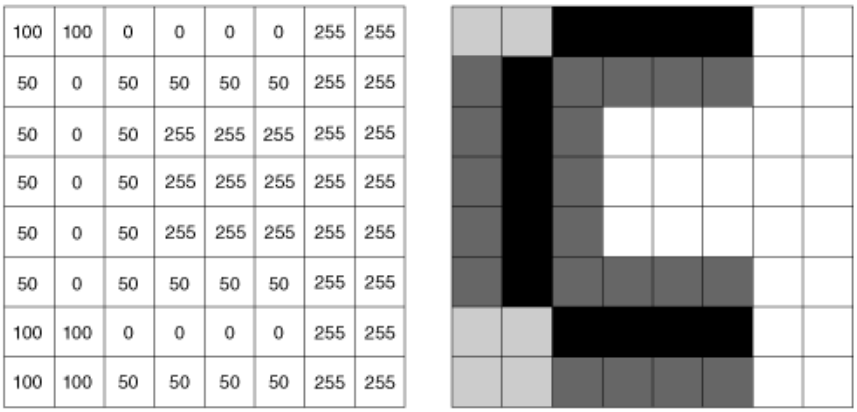
## 2.2 Representação de imagens digitais

Uma imagem digital é formada por elementos chamados *pixels*[1], que representam uma intensidade de energia captada por um elemento específico do sensor que captura a imagem. Por causa da disposição espacial dos *pixels*, uma imagem digital é representada por uma matriz numérica de duas dimensões quando o sensor é monocromático, ou por uma matriz numérica de $d$ dimensões quando a imagem não é monocromática. Neste trabalho, denotaremos uma imagem digital por $I$, seja ela monocromática ou não, e um *pixel* específico da imagem por $I(x, y)$, onde o par $(x, y)$ denota os índices da representação do pixel na matriz.

Em imagens monocromáticas cada *pixel* é representado por um valor inteiro finito dentro de um intervalo $[0, L]$. Neste trabalho, é usado $L = 255$, então a imagem é representado em 256 níveis de cinza.

[1] Termo derivado da contração da expressão *picture element*.

No caso de imagens coloridas, cada dimensão da matriz corresponde a um canal de cor representado em 256 níveis de intensidade. Em imagens RGB, por exemplo, a representação é feita por uma matriz tridimensional (com tons de vermelho, verde e azul).



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 100 | 100 | 0 | 0 | 0 | 0 | 255 | 255 |
| 50 | 0 | 50 | 50 | 50 | 50 | 255 | 255 |
| 50 | 0 | 50 | 255 | 255 | 255 | 255 | 255 |
| 50 | 0 | 50 | 255 | 255 | 255 | 255 | 255 |
| 50 | 0 | 50 | 255 | 255 | 255 | 255 | 255 |
| 50 | 0 | 50 | 50 | 50 | 50 | 255 | 255 |
| 100 | 100 | 0 | 0 | 0 | 0 | 255 | 255 |
| 100 | 100 | 50 | 50 | 50 | 50 | 255 | 255 |



**Figura 2.3:** *Imagem digital*
Imagem em escala de cinza; (a) é a representação matricial de (b), na qual cada entrada corresponde ao nível de cinza do *pixel* na posição correspondente na imagem.

## 2.3 Representação de imagens digitais como grafos

Na representação de uma imagem digital por um grafo com pesos nas arestas, os elementos em $V$ podem ser *pixels* ou *superpixels* (a construção desses é explicada no capítulo 3) e o peso associado a cada aresta é uma medida de dissimilaridade (*e.g.* diferença de intensidade, cor, localização, etc.) entre os elementos conectados por essa aresta.

Há várias formas de se definir a vizinhança de um grafo em imagens. No caso de *pixels*, é comum considerar uma das seguintes vizinhanças:

- **4-conectados**: um *pixel* na posição $(x, y)$ é vizinho dos *pixels* cujas coordenadas são $(x \pm 1, y)$ ou $(x, y \pm 1)$;
- **8-conectados**: um *pixel* na posição $(x, y)$ é vizinho dos *pixels* cujas coordenadas são $(x \pm 1, y \pm 1)$.

No caso de *superpixels*, é comum utilizar o conceito de Grafo de Regiões Adjacentes (*Region Adjacency Graph* ou RAG), no qual dois vértices são conectados se os respectivos *superpixels* são adjacentes na imagem. Note que o RAG pode ser construído a partir de qualquer partição da imagem, não necessariamente a partir de *superpixels*.

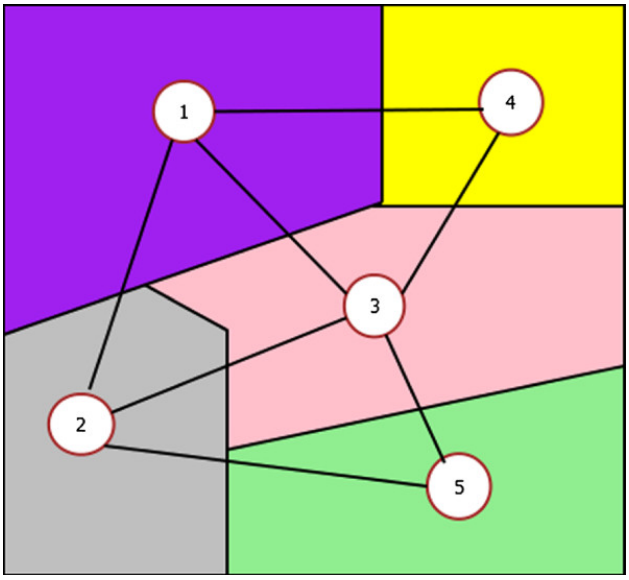



**Figura 2.4:** *Exemplo de RAG*
Cada região colorida corresponde a um *superpixel*, que é representado por um vértice no grafo. Para regiões adjacentes, há arestas conectando seus respectivos vértices.

## 2.4 Segmentação de imagens baseada em grafos

Uma vez obtida a representação da imagem por grafo, podemos pensar na segmentação de imagem como um problema de particionamento de grafo: cada componente do grafo definirá um componente (ou segmento) de interesse na imagem. Note que cada componente do grafo irá corresponder a um conjunto de *pixels* (ou *superpixels*) conexos, como mostra a figura 2.5.

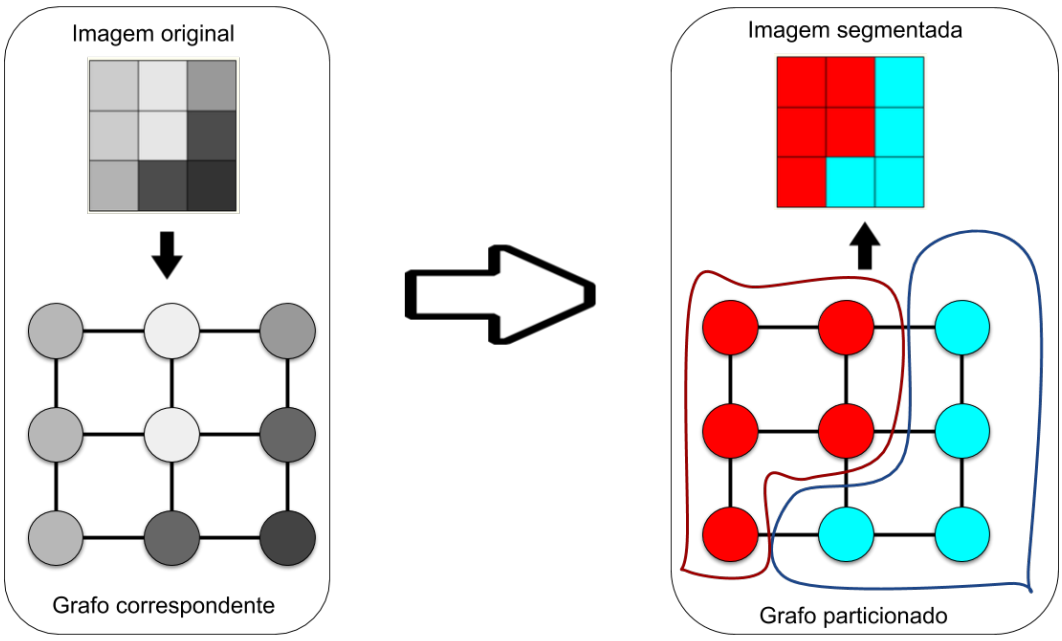



**Figura 2.5:** *Segmentação baseada em grafo*
A segmentação da imagem é obtida pelo particionamento do grafo (à esquerda) em dois componentes, indicados no grafo à direita (os *pixels* contidos em cada um deles estão indicados em cores diferentes). Cada componente do grafo corresponde a um segmento no resultado do processo.

O particionamento do grafo pode ser feito através de um corte ótimo. No caso de imagens, um corte no grafo corresponde a dividir a imagem (ou subimagem) em duas regiões. Há diversos critérios para determinar um corte ótimo, entre eles estão o Corte Mínimo (Wu e Leahy (1993)) e o Corte Normalizado (Shi e Malik (2000)).

## 2.5 Propriedades da segmentação baseada na MST

Assim como os métodos que serão apresentados no capítulo 4, Morris *et al.* (1986) desenvolve no artigo *Graph Theory for Image Analysis* uma técnica para segmentação de imagens baseada na MST do grafo construído a partir da imagem. Ele menciona as seguintes propriedades de MST relevantes à segmentação de imagens:

- Uso de informação espacial a respeito de *pixels* (ou *superpixels*) vizinhos;
- contorno de regiões são definidos com acurácia;
- as regiões produzidas são próximas;
- a MST contém toda a informação necessária para separar a imagem em qualquer número de regiões, conforme a ordem hierárquica de contraste;
- quando uma região é criada, somente sua região de origem é modificada. Ou seja, os contornos estabelecidos não se movem quando a segmentação inclui mais detalhes (em outras palavras, uma segmentação mais fina).

No capítulo 5 vamos verificar, através dos experimentos, como os algoritmos estudados se comportam em relação a essas propriedades.

# Capítulo 3

# Da imagem aos *superpixels*

Neste capítulo, apresentamos o algoritmo usado na fase de pré-processamento, cujo objetivo é particionar o domínio da imagem em entidades que representam homogeneamente pequenas regiões da imagem: os *superpixels*. O resultado dessa supersegmentação é a base para construir um grafo contendo informações significativas da imagem e, ao mesmo tempo, com um tamanho reduzido quando comparado ao grafo gerado tendo *pixels* como vértices.

## 3.1 O método SLIC

Proposto por Achanta *et al.* (2010), o método *Simple Linear Iterative Clustering* (SLIC) consiste em agrupar localmente *pixels* em um espaço pentadimensional definido pelos valores de intensidade no espaço de cor *Lab* (ou CIELAB) e pelas coordenadas espaciais na imagem.

O espaço de cor CIELAB descreve matematicamente em três dimensões as cores perceptíveis por seres humanos. As dimensões são $L$, $a$ e $b$, que se referem a luminosidade, coordenada vermelho/verde e coordenada amarelo/azul, respectivamente. Esse espaço é amplamente considerado como perceptualmente uniforme para pequenas distâncias entre cores, pois correlaciona consistentemente [1] as intensidades de cor com a percepção visual do olho humano. Cada cor é tratada como um ponto nesse espaço tridimensional, e a diferença entre duas cores é calculada através da distância Euclidiana entre suas coordenadas.

A distância máxima entre duas cores no espaço CIELAB é limitada de maneira diferente da distância espacial entre dois *pixels*, porque esta depende do tamanho da imagem. Dessa forma, não podemos usar simplesmente a distância entre *pixels* no espaço *5D* sem que as distâncias espaciais sejam normalizadas.

### 3.1.1 Medida de distância

Um dos parâmetros do SLIC é o número $K$, correspondente a quantidade desejada de *superpixels*. Para uma imagem com $N$ *pixels*, o tamanho de cada um é aproximadamente $N/K$ *pixels*, e o centro de cada um deles se encontra a intervalos espaçados de tamanho $S = \sqrt{N/K}$.

[1] Ver Jain (1989).

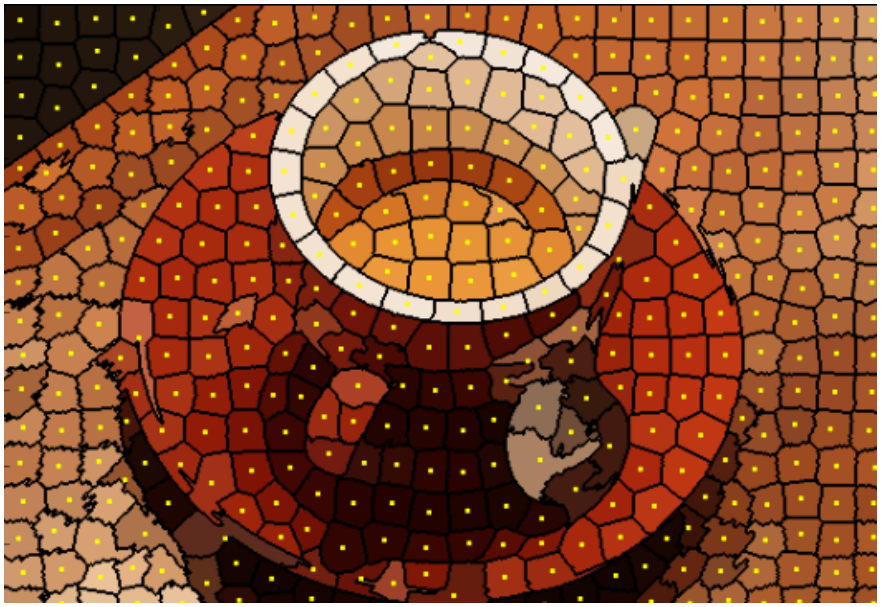


**Figura 3.1:** *Superpixels SLIC*
Construção de *superpixels* usando o algoritmo SLIC (com K = 400). As fronteiras estão delimitadas em preto e os pontos amarelos correspondem aos centros.

No algoritmo, esses $K$ centros (*cluster centers*) são definidos por $C_k = (L_k, a_k, b_k, x_k, y_k)$, com $k \in [1, K]$. Como a área aproximada de cada *superpixel* é $S^2$, podemos assumir que os *pixels* presentes na área de dimensão $2S \times 2S$ em torno de cada centro são associados a esse *cluster center* no plano espacial da imagem. É nessa área que a busca pelos *pixels* mais próximos de cada centro é realizada.

Como dito anteriormente, a distância Euclidiana no espaço CIELAB é perceptualmente significante em pequenas distâncias. Quando a distância espacial entre os *pixels* excede o limite da distância de cor, ela passa a se sobrepor em relação às diferenças de cor dos *pixels*. Isso resulta em *superpixels* que respeitam apenas a proximidade espacial entre eles, desconsiderando os limites entre regiões de baixa similaridade. Por esse motivo, ao invés de usar a norma Euclidiana no espaço $5D$, definimos a distância $D_s$, entre dois *pixels* $i = (x_i, y_i)$ e $j = (x_j, y_j)$, como a combinação das distâncias $d_{lab}$ e $d_{xy}$:

$$
\begin{aligned}
d_{lab}(i,j) &= \sqrt{(L_i - L_j)^2 + (a_i - a_j)^2 + (b_i - b_j)^2} \\
d_{xy}(i,j) &= \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \\
D_s(i,j) &= d_{lab}(i,j) + \frac{m}{S} d_{xy}(i,j)
\end{aligned}
$$

A distância $D_s$ é a soma da distância no espaço CIELAB e da distância no plano espacial normalizada pelo intervalo $S$ fixado. Nessa fórmula, é introduzida uma variável $m$ que permite controlar a compacidade do *superpixel*, isto é, altos valores de $m$ enfatizam a distância espacial na fórmula e tornam o *superpixel* mais compacto.

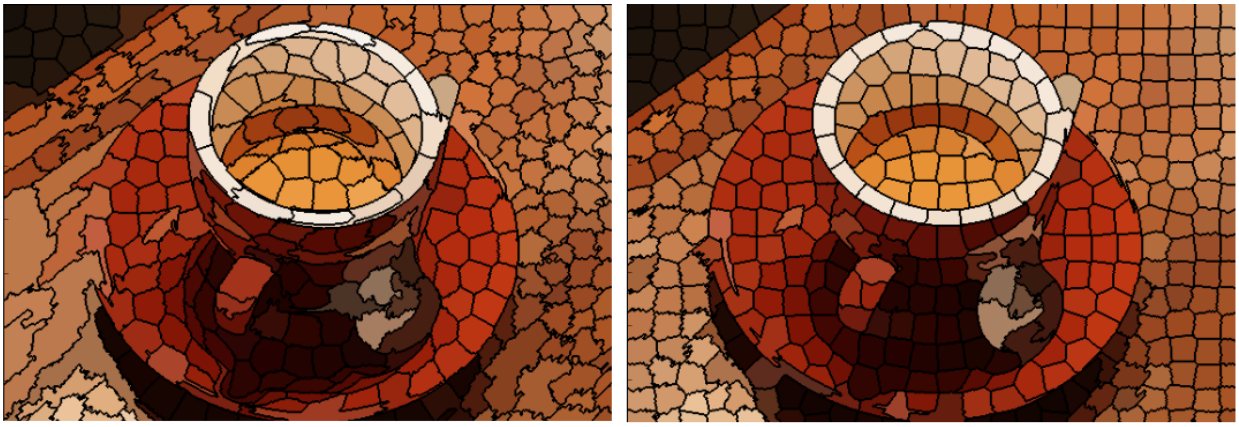



**Figura 3.2:** *Superpixels com diferentes valores de compacidade m*
*Superpixels* com maior valor de compacidade apresentam bordas mais regulares.

### 3.1.2 Algoritmo

O algoritmo SLIC começa definindo as posições uniformemente espaçadas dos $K$ *cluster centers*, e os realoca para a posição com o menor gradiente em uma vizinhança $3 \times 3$. O gradiente de um *pixel* da imagem é definido por:

$$G(x, y) = \|I(x+1, y) - I(x-1, y)\|^2 + \|I(x, y+1) - I(x, y-1)\|^2 \tag{3.1}$$

onde I(x, y) é o vetor $(L, a, b)$ correspondente ao *pixel* da posição $(x, y)$ e $\|.\|$ é a norma $L_2$. Isso é feito para evitar que o *cluster center* seja posicionado na borda de algum componente da imagem ou em um *pixel* de ruído.

Em seguida, cada *pixel* é associado ao *cluster center* mais próximo. Após essa etapa, redefinimos a posição de cada *cluster center* como a média dos vetores $(L, a, b, x, y)$ de cada *pixel* contido no *superpixel/cluster* correspondente. Repete-se o processo iterativamente até a convergência (baseada na distância entre a posição anterior do *cluster center* e a que é recalculada).

**Algoritmo 1:** Algoritmo SLIC

**Entrada:** Imagem RGB, parâmetros $K$ e $m$
**Saída :** Matriz com rótulos da imagem

converter a imagem RGB para CIELAB
inicializar os *cluster centers* $C_k = (L_k, a_k, b_k, x_k, y_k)$ como sendo os *pixels* com espaçamento regular de tamanho $S$, atribuindo um rótulo único para cada um deles
realocar os centros para a posição de menor gradiente
**repita**
    **para** *cada* $C_k$ **faça**
        atribua seu rótulo aos *pixels* na vizinhança $2S \times 2S$ que estão mais próximos desse centro
        calcular o erro residual $E$ (distância entre posição anterior dos *cluster centers* e a recalculada)
    **fim**
**até** $E \leq limite$;
reforçar a conectividade

No fim do algoritmo, reforçamos a conectividade dos *superpixels* atualizando os rótulos

de pequenas regiões próximas de regiões maiores, mas não ligadas a ela.

O algoritmo calcula as distâncias de cada *pixel* a, no máximo, 8 *cluster centers* (limitação pela vizinhança). Os autores afirmam no artigo que 10 iterações é uma quantidade suficiente para obter resultados satisfatórios, pois o erro de convergência diminui drasticamente em poucas iterações. Por essa razão, o método tem consumo de tempo linear no número de *pixels* da imagem.

# Capítulo 4

# Segmentação de imagens

Neste capítulo, são abordados os métodos de segmentação de Felzenszwalb e Guimarães, que particionam a imagem a partir da MST do grafo associado a ela, produzido a partir de *pixels* ou *superpixels*.

O método de Guimarães et al. é baseado no método de Felzenszwalb & Huttenlocher, mas produz a segmentação utilizando o conceito de hierarquias.

## 4.1 O método de Felzenszwalb & Huttenlocher

A técnica desenvolvida por Felzenszwalb e Huttenlocher (2004) seleciona arestas de um grafo através do ajuste de um critério de segmentação baseado no grau de variabilidade entre regiões vizinhas da imagem. Isso resulta em um método que, enquanto faz decisões gulosas, mostra-se de acordo com certas propriedades globais.

O método mede o nível de evidência para haver um contorno entre duas regiões a partir da comparação de duas quantidades: uma baseada na dissimilaridade entre regiões e outra baseada na dissimilaridade entre componentes dentro de uma mesma região. Intuitivamente, as diferenças entre as regiões (cruzamento de um contorno) são perceptualmente importantes quando são grandes em comparação às diferenças dentro de alguma das regiões.

### 4.1.1 Predicado de Comparação de Regiões Similares

Vamos definir o predicado $D$ para avaliar se há evidência de existência de contorno entre duas regiões da imagem. $D$ compara as diferenças inter-componentes às diferenças intra-componentes e, portanto, é adaptativa em relação às características locais da imagem. Dado um grafo $G = (V, E)$ da imagem:

**Definição 4.1.1 (Diferença interna)** *A diferença interna de um componente $C \subseteq V$ é o maior peso associado a alguma aresta na $MST(C, E')$, $E' \subseteq E$*

$$Int(C) = \max_{e \in MST(C,E')} w(e)$$

**Definição 4.1.2 (Diferença entre dois componentes)** *A diferença entre dois componentes $C_1, C_2 \subseteq V$ é a aresta de peso mínimo conectando duas componentes:*

$$Diff(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, \{v_i, v_j\} \in E} w(\{v_i, v_j\})$$

O predicado verifica se $Diff(C_1, C_2)$ é grande em relação à diferença interna dentro de pelo menos uma das componentes.

Uma função de limiarização é usada para controlar o grau a partir do qual a diferença entre componentes deve ser maior que a diferença interna mínima.

**Definição 4.1.3 (Predicado de Comparação de Regiões Similares)**

$$D(C_1, C_2) = \begin{cases} VERDADEIRO & se\ Diff(C_1, C_2) > MInt(C_1, C_2) \\ FALSO & caso\ contrário \end{cases}$$

onde a diferença interna mínima vale:

**Definição 4.1.4 (Diferença interna mínima)**

$$MInt(C_1, C_2) = \left\{ Int(C_1) + \frac{k}{|C_1|}, Int(C_2) + \frac{k}{|C_2|} \right\}$$

$|C_i|$ corresponde à quantidade de unidades (*pixels* ou *superpixels*) que a região $i$ possui, e $k$ é um parâmetro que previne a fusão de regiões grandes: valores grandes de $k$ "forçam" a fusão de regiões menores, pois exigem uma evidência maior para existência de contorno. Na prática, $k$ define a escala de observação, na qual altos valores de $k$ fazem com que o algoritmo tenda a produzir componentes maiores.

O critério para fusão de regiões se baseia em uma *Escala de Observação*, que depende de $k$. Para duas regiões $X$ e $Y$, a escala de observação $S_y(X)$, de $X$ relativa a $Y$, é definida por

$$S_Y(X) = (Diff(X, Y) - Int(X)) * |X| \tag{4.1}$$

No qual $|X|$ é o tamanho do componente $X$. Com isso, definimos a Escala $S(X, Y)$

**Definição 4.1.5 (Escala de observação $S(X, Y)$)** *:*

$$S(X, Y) = \max \{S_Y(X), S_X(Y)\}$$

$S_X(Y)$ é definida analogamente a $S_Y(X)$. Através dessa escala, determinamos a condição para as regiões vizinhas $X$ e $Y$ serem fundidas. Isso acontece quando a escala delas é menor que o parâmetro de limiarização $k$, ou seja, quando

$$k \geq S(X, Y) \tag{4.2}$$

### 4.1.2 Algoritmo e propriedades

O algoritmo produz a segmentação utilizando o critério de decisão $D$ definido anteriormente.

**Algoritmo 2:** Método de FH

**Entrada:** grafo $G = (V, E)$ com n vértices e m arestas
**Saída :** uma segmentação de V em componentes $S = \{C_1, ..., C_n\}$

ordena E em $\pi = (e_1, ..., e_m)$ em ordem não decrescente do peso das arestas
comece com uma segmentação $S^0$ , onde cada vértice $v_i$ está em seu próprio componente
**para** $q \leftarrow 1, ..., m$ **faça**
  // constrói $S^q$, dado $S^{q-1}$
  $v_i, v_j \leftarrow vertex(e_q)$ ▷ vértices conectados pela aresta $e_q$
  $C_i^{q-1} \leftarrow comp(S^{q-1}, v_i)$ ▷ componente de $S^{q-1}$ contendo $v_i$
  $C_j^{q-1} \leftarrow comp(S^{q-1}, v_j)$
  **se** $C_i^{q-1} \neq C_j^{q-1}$ ***and*** $w(e_q) \leq MInt(C_i^{q-1}, C_j^{q-1})$ **então**
    // $S^q$ é obtida a partir da fusão entre as componentes
    $S^q \leftarrow union(C_i^{q-1}, C_j^{q-1})$
  **fim**
  **senão**
    $S^q = S^{q-1}$
  **fim**
**fim**
**retorna** $S = S^m$

O algoritmo é computacionalmente eficiente, pois roda em tempo $O(nlogn)$, para $n$ *pixels* na imagem.

Em relação as suas propriedades, considere as seguintes definições:

- **Definição 1**: Uma segmentação $S$ é *muito fina* se existe um par de regiões $C_1, C_2 \in S$ para o qual não há evidência de contorno entre elas;
- **Definição 2**: Uma segmentação $S$ é *muito grossa* se existe um refinamento apropriado de $S$ que não é muito fino.

Os autores provam no artigo, a partir das definições acima, que o algoritmo tem a seguinte propriedade:

***Propriedade****: para qualquer grafo (finito) $G = (V, E)$ existe uma segmentação $S$ que não é tão grossa nem tão fina.*

ou seja, a segmentação produzida obedece propriedades globais de não ser tão fina nem tão grossa quando se usa o predicado de comparação $D$, mesmo que o algoritmo faça decisões gulosas.

### 4.1.3 Problemas do método

1. O número de regiões pode aumentar quando o parâmetro $k$ aumenta, o que não deveria ocorrer se $k$ for uma verdadeira escala de observação. Isso viola o *Princípio de Causalidade*: um contorno presente em uma escala $k_1$ deveria estar presente em toda escala $k_2 < k_1$ (o número de regiões deveria diminuir quando a *escala de observação* aumenta).

2. Até mesmo quando o número de regiões decresce, os contornos não são estáveis: eles podem se mover quando o parâmetro $k$ varia, violando o *Princípio de Localização*.


**Figura 4.1:** *Violação do Princípio de Localização*
Contornos presentes na segmentação com valor $k$ mais alto não estão presentes na segmentação com valor mais baixo.

## 4.2 O método de Guimarães et al.

Métodos hierárquicos têm como propriedade a preservação de informações de espaço e vizinhança entre as regiões segmentadas. No método de Guimaraes *et al.* (2012), cada hierarquia é representada por uma MST e a segmentação é feita a partir de uma hierarquia de partições baseada em diferentes escalas de observação.

### 4.2.1 Método hierárquico

Considere um mapa $w : E \to \mathbb{N}$, que associa pesos a arestas, e um parâmetro $\lambda \in \mathbb{N}$ de limiarização.

Para toda árvore $T$ geradora do conjunto $V$ de unidades da imagem, todo mapa $w$ e toda limiarização $\lambda$, podemos associar a partição $P^w_\lambda$ de $V$ induzida pelos componentes conexas do grafo criado por $V$ e as arestas associadas a pesos $w < \lambda$.

Para dois valores quaisquer de $\lambda_1$ e $\lambda_2$ de forma que $\lambda_1 \geq \lambda_2$, as partições $P^w_{\lambda_1}$ e $P^w_{\lambda_2}$ são *aninhadas* se $P^w_{\lambda_1}$ é *mais grossa* que $P^w_{\lambda_2}$. Dessa forma, o conjunto $\mathcal{H}^w = \{P^w_\lambda | \lambda \in \mathbb{N}\}$ é uma *hierarquia de partições induzidas pelo mapa de peso w.*

### 4.2.2 Escala de Observação

Para $k$ ser uma verdadeira escala de observação, o método deve satisfazer o princípio de causalidade e o de localização. A segmentação hierárquica produz todas as escalas de observação, ao invés de uma só segmentação. Dessa forma, os dois princípios citados são respeitados.

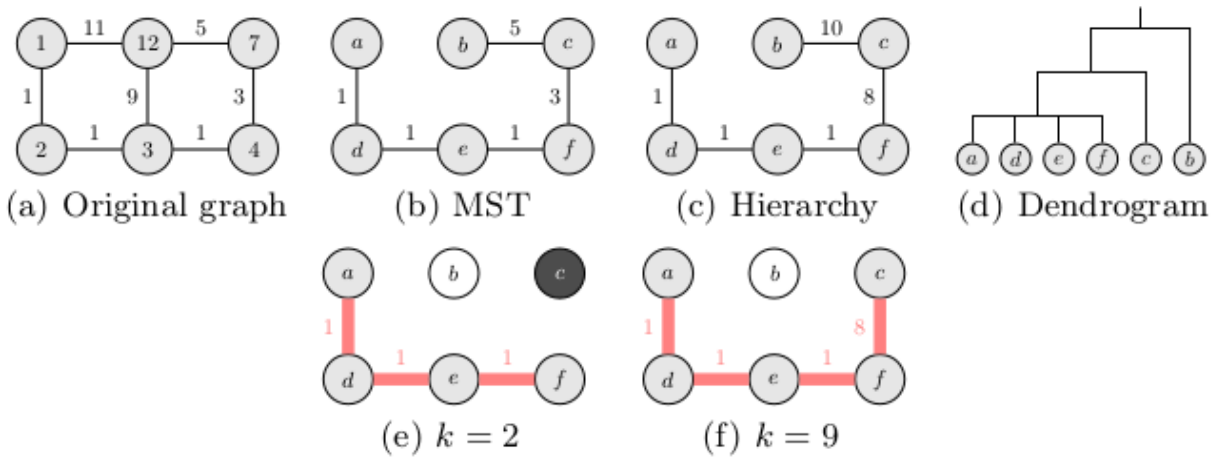



**Figura 4.2:** *Grafos de hierarquias*
A partir do grafo (a), o método de Guimarães et al. produz a escala de observação hierárquica (c). Assim como o método de Felzenszwalb, esse método utiliza somente as arestas da árvore geradora mínima (b). (d) mostra a hierarquia completa gerada pelo algoritmo. (e) e (f) são dois níveis da hierarquia, nas escalas 2 e 9, respectivamente.

### 4.2.3 Algoritmo

O algoritmo de Guimaraes *et al.* (2012) produz um mapa de pesos $L$ (escala de observação) a partir do qual a hierarquia desejada $\mathcal{H}^L$ pode ser inferida com base em uma árvore geradora $T$ fornecida.

O algoritmo parte de uma árvore geradora de custo mínimo $T$ do grafo da imagem. Para calcular o $L(e)$ associado a cada aresta de $T$, o método funciona da seguinte maneira:

**Algoritmo 3:** Método de Guimarães

**Entrada:** grafo $G = (V, E)$ com n vértices e m arestas
**Saída** : grafo $G'$ cujos pesos da aresta são os valores do mapeamento $L$

1 ordena E em $\pi = (e_1, ..., e_m)$ em ordem não decrescente do peso das arestas
2 **para** $e \in E$ **faça**
3 | $L(e) \leftarrow +\infty$
4 **fim**
5 **para** $e \in E$ **faça**
6 | $x, y \leftarrow vertex(e)$ ▷ vértices conectados pela aresta $e$
7 | Encontre a região $X$ de $P^w_{w(e)}$ que contém $x$
8 | Encontre a região $Y$ de $P^w_{w(e)}$ que contém $y$
9 | Calcule a escala de obervação hierárquica $L(e)$
10 **fim**

Na linha 9, precisamos da **escala hierárquica** $S'_Y(X)$ de $X$ relativa a $Y$ para obter o valor de $L(e)$.

Intuitivamente, dizemos que $S'_Y(X)$ é a *menor escala de observação* na qual alguma subregião de $X$ (chamamos de $X^*$) será fundida a $Y$.

**Algoritmo 4:** Escala Hierárquica

```
// Usando um parâmetro interno v, a escala é calculada da
   seguinte forma:
```

1 Inicialize o valor de $v$ com $+\infty$
2 Decremente o valor de $v$ em 1 unidade
3 Encontre a região $X^*$ de $P_v^L$ que contém $x$
4 Repita 2 e 3 enquanto $S_Y(X^*) < v$
5 Obtenha $S'_X(Y)$ de forma análoga.

ou seja, a escala hierárquica $L(e)$ pode ser definida como:

**Definição 4.2.1 (Escala Hierárquica)**

$$L(e) = \max\{S'_Y(X), S'_X(Y)\}$$

Para obter a segmentação da imagem a partir do método de Guimarães, o parâmetro $k$ seleciona a segmentação equivalente a essa escala de observação.

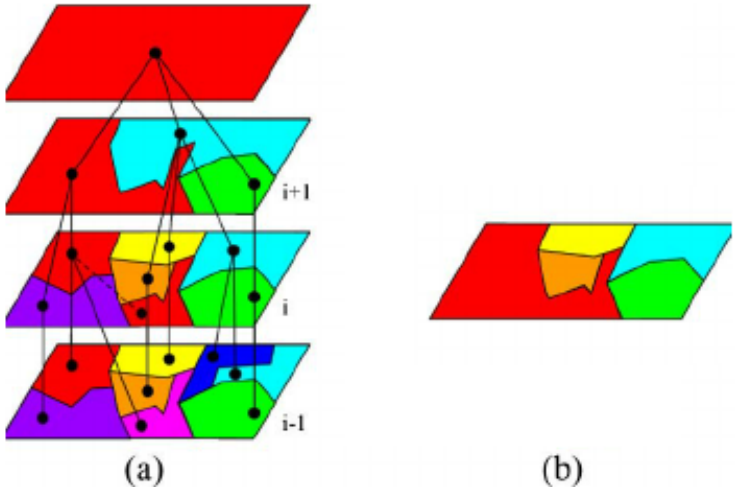



**Figura 4.3:** *Segmentação hierárquica*
O resultado (b) é a segmentação obtida pela escala $i$ da hierarquia (a).

# Capítulo 5

# Experimentos

Neste capítulo são apresentados os resultados dos algoritmos estudados aplicados a diferentes imagens, com variações de parâmetros e partindo de diferentes níveis: *pixel* e *superpixel*.

Para os experimentos realizados neste trabalho, foram utilizadas imagens da base de dados *The Berkeley Segmentation Dataset and Benchmark* (Martin *et al.* (2001)) e da *Airplanes Database* do Instituto de Tecnologia da Califórnia[1].

Os algoritmos de segmentação foram implementados na linguagem *Python*, no ambiente interativo *IPython Notebook* (Pérez e Granger (2007)). Demais funções e algoritmos foram utilizados das bibliotecas *scikit-image*, *NumPy* e *NetworkX*.

Abaixo constam as informações da máquina utilizada:

```
Sistema Operacional Debian 8.9 (x86-64)
Processador Intel Core i3-3217U CPU @ 1.80 GHz x 2
Memória: 3.8 GB
```

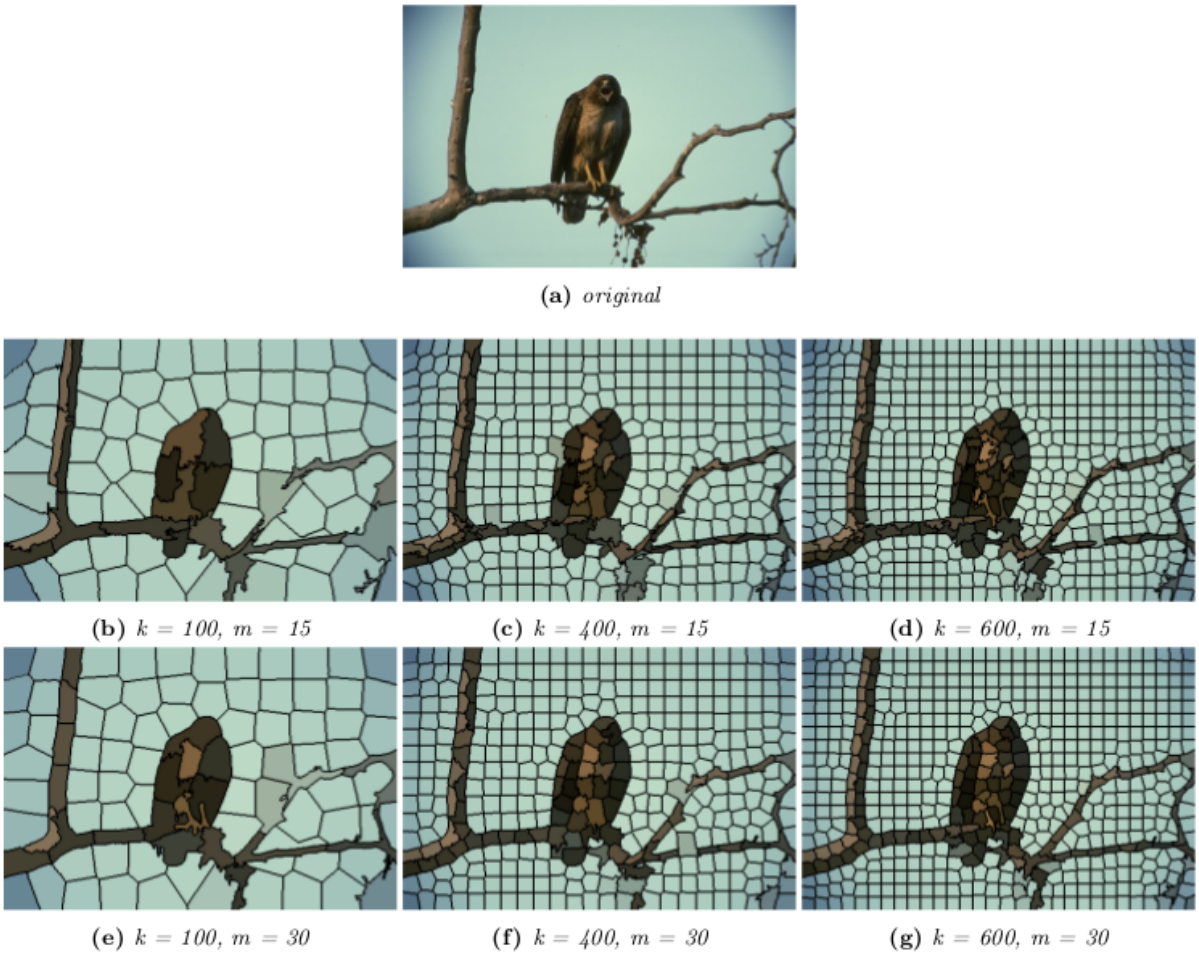
A estrutura de dados usada para gerenciar os particionamentos construídos pelos algoritmos de segmentação foi a *Union Find* (Cormen *et al.* (2009)). A medida de dissimilaridade usada para a construção das arestas dos grafos da imagem foi a a média da intensidade das cores.

## 5.1 *Superpixels*

Vamos analisar os resultados obtidos pelo *SLIC*. Na figura 5.1 podemos observar a imagem do pássaro supersegmentada em diferentes níveis de detalhes, determinados pela quantidade $K$ escolhida de *superpixels*. Conforme $K$ aumenta, a imagem é supersegmentada mais vezes, de forma que regiões menores conseguem ser formadas respeitando os contornos de áreas estreitas da imagem. Observe que parte do galho à direita do pássaro não tem suas propriedades locais respeitadas para valores baixos de $K$ (a estrutura dos *superpixels* nessa região não respeita o contorno).

Experimentos realizados por Achanta *et al.* (2010) mostram que a compacidade $m$ dos *superpixels* influencia diretamente na aderência da região aos contornos da imagem. No entanto, podemos observar que, mesmo para um valor mais alto de $m$, imagens com regiões estreitas apresentam problemas para a geração de *superpixels* que preservem completamente os contornos.

[1] Disponível em http://www.vision.caltech.edu/archive.html

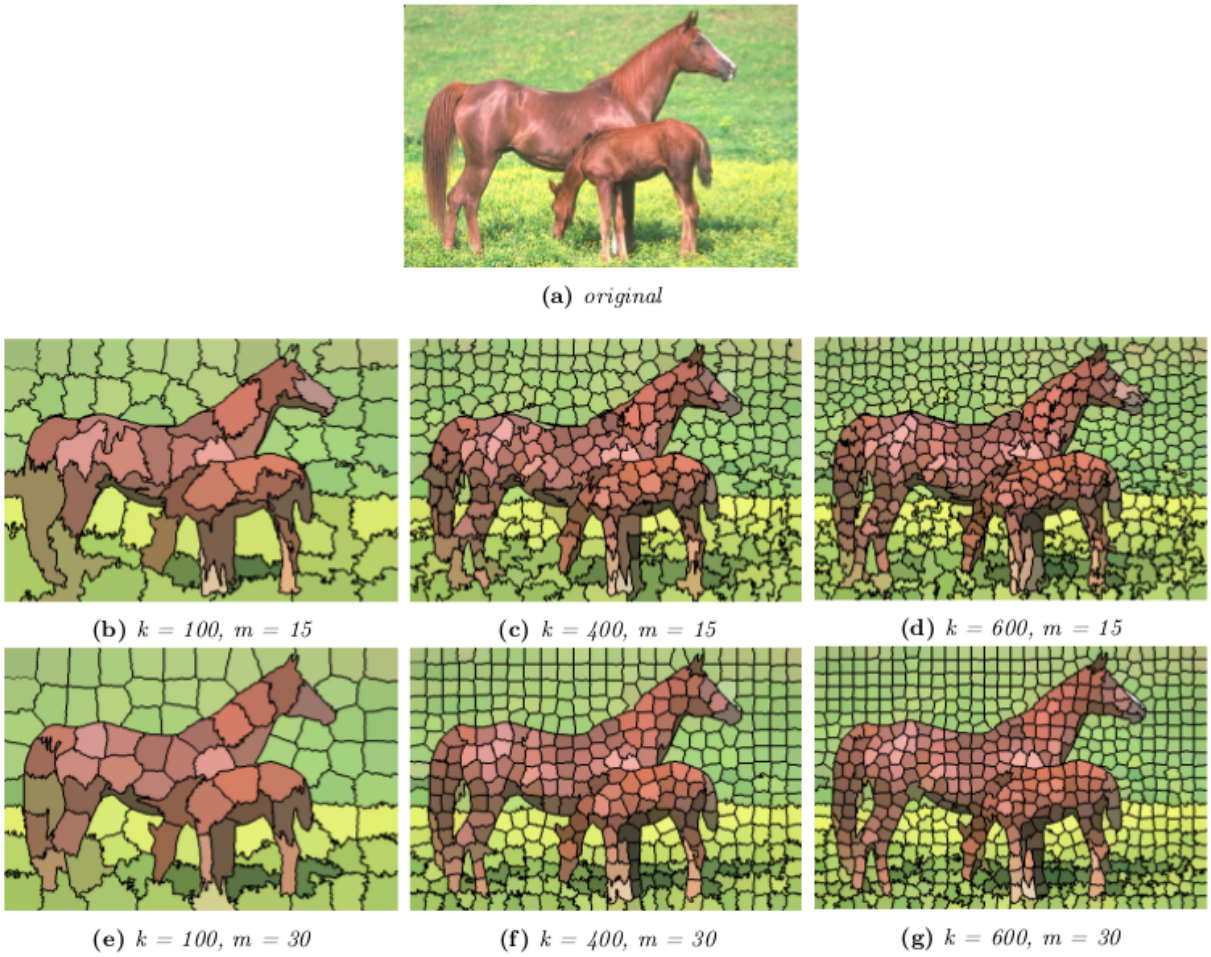


**(a)** *original*

**(b)** $k = 100,\ m = 15$ **(c)** $k = 400,\ m = 15$ **(d)** $k = 600,\ m = 15$

**(e)** $k = 100,\ m = 30$ **(f)** $k = 400,\ m = 30$ **(g)** $k = 600,\ m = 30$

**Figura 5.1:** *Variações dos parâmetros do SLIC*
Imagens da mesma linha foram criadas com mesmos valores de $m$. Imagens de colunas diferentes variam no número de *superpixels*.



**(a)** *original*

**(b)** $k = 100,\ m = 15$ **(c)** $k = 400,\ m = 15$ **(d)** $k = 600,\ m = 15$

**(e)** $k = 100,\ m = 30$ **(f)** $k = 400,\ m = 30$ **(g)** $k = 600,\ m = 30$

**Figura 5.2:** *Variações dos parâmetros do SLIC: imagem com objetos que se sobrepõem*
As escolhas dos parâmetros na figura (g) são as que mais preservam os contornos dos animais na imagem, dentre os valores testados.

## 5.2 Segmentação a partir de *superpixels*

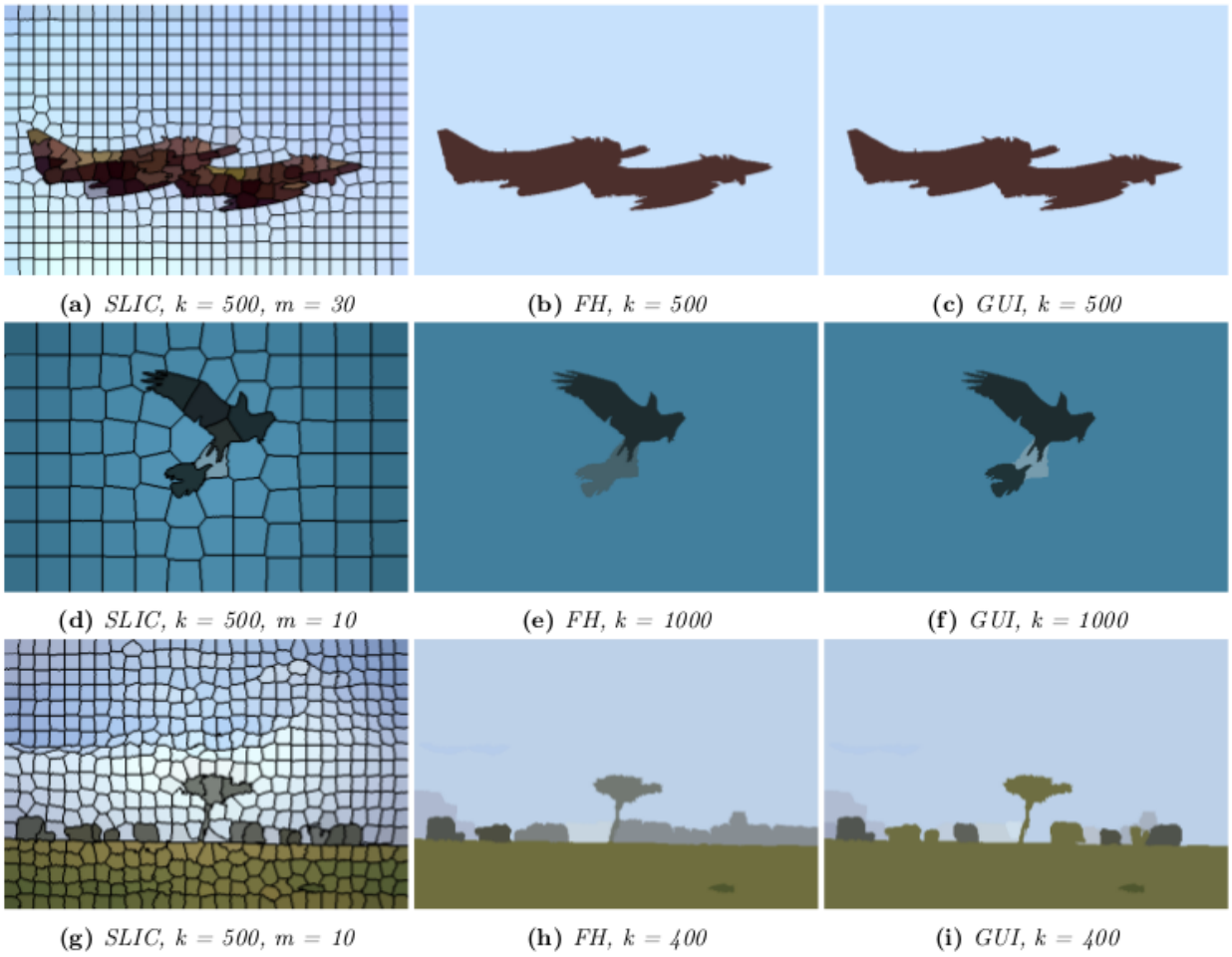



**Figura 5.3:** *Imagens segmentadas a partir de superpixels SLIC (I)*

A figura 5.3 mostra resultados de segmentação a partir dos *superpixels*. Na primeira imagem, ambos os resultados são satisfatórios: não há grande variação de cores nos objetos da imagem; isso facilita a segmentação que separa objetos (primeiro plano) e o plano de fundo (*foreground-background segmentation*). Já nas outras duas, o resultado para o algoritmo Felzenszwalb aglutina objetos distintos da imagem: o pássaro menor com a penagem branca do pássaro maior na segunda imagem, e a maior parte das pedras com regiões do céu/horizonte, na terceira.

Visto que o custo das arestas dos grafos são construídas somente com informação de intensidade de cor, é natural encontrar dificuldades para obter bons resultados para imagens cujos objetos tenham regiões com tonalidades próximas às do plano de fundo, como por exemplo na imagem *(d)* da figura 5.4. Algo similar ocorre com a imagem *(g)*, devido a região escura de sombra, na parte inferior da imagem. As demais variações ocorrem porque o método de Guimarães não segmenta o mesmo grafo que o método de Felzenszwalb, mas uma variação dele (os custos nas arestas são um mapeamento dos valores dos custos das arestas da MST).

Conforme o artigo de Guimaraes *et al.* (2012), seu método utiliza uma verdadeira escala de observação, pois respeita o princípio de localização, como mostra a figura 5.5.

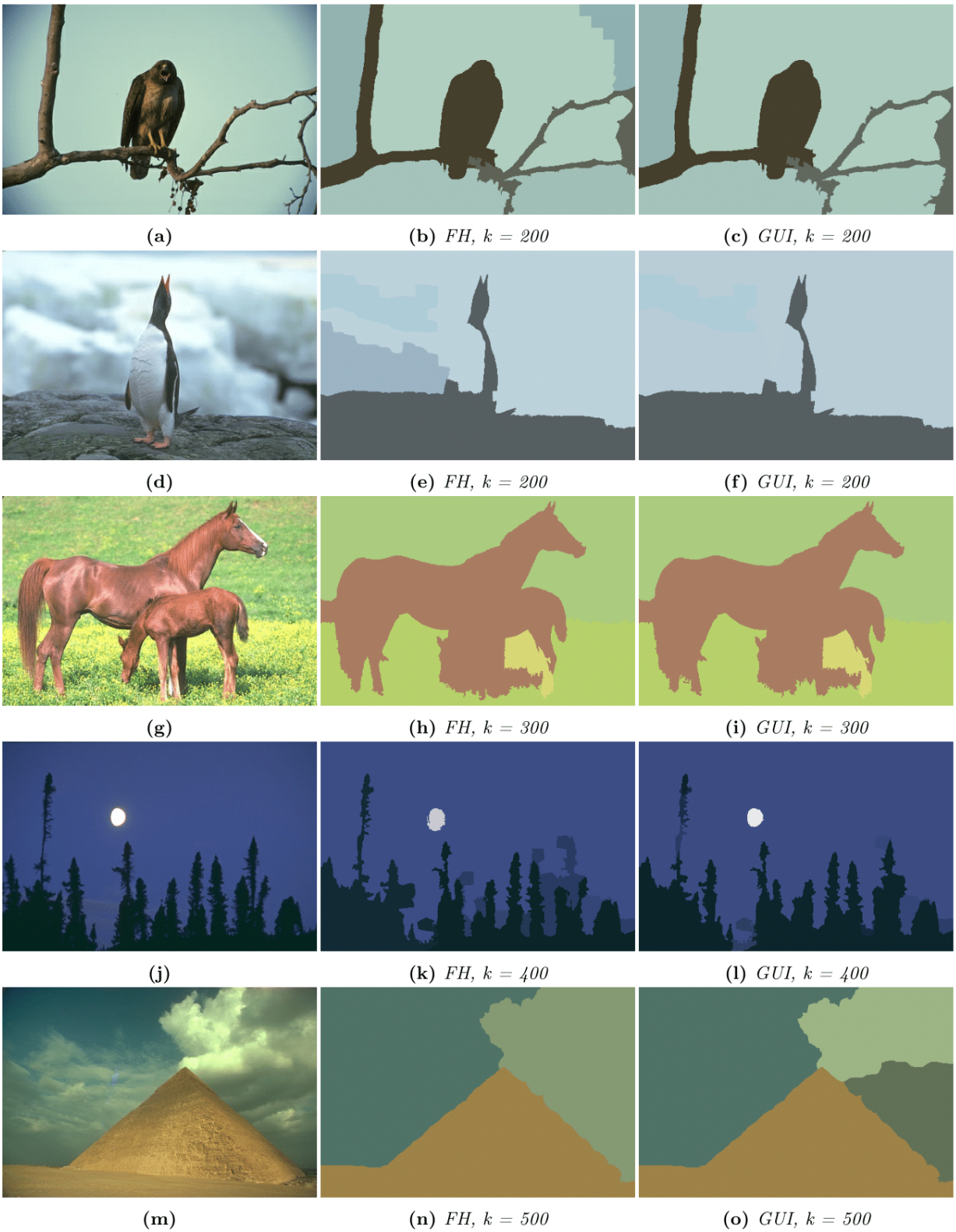
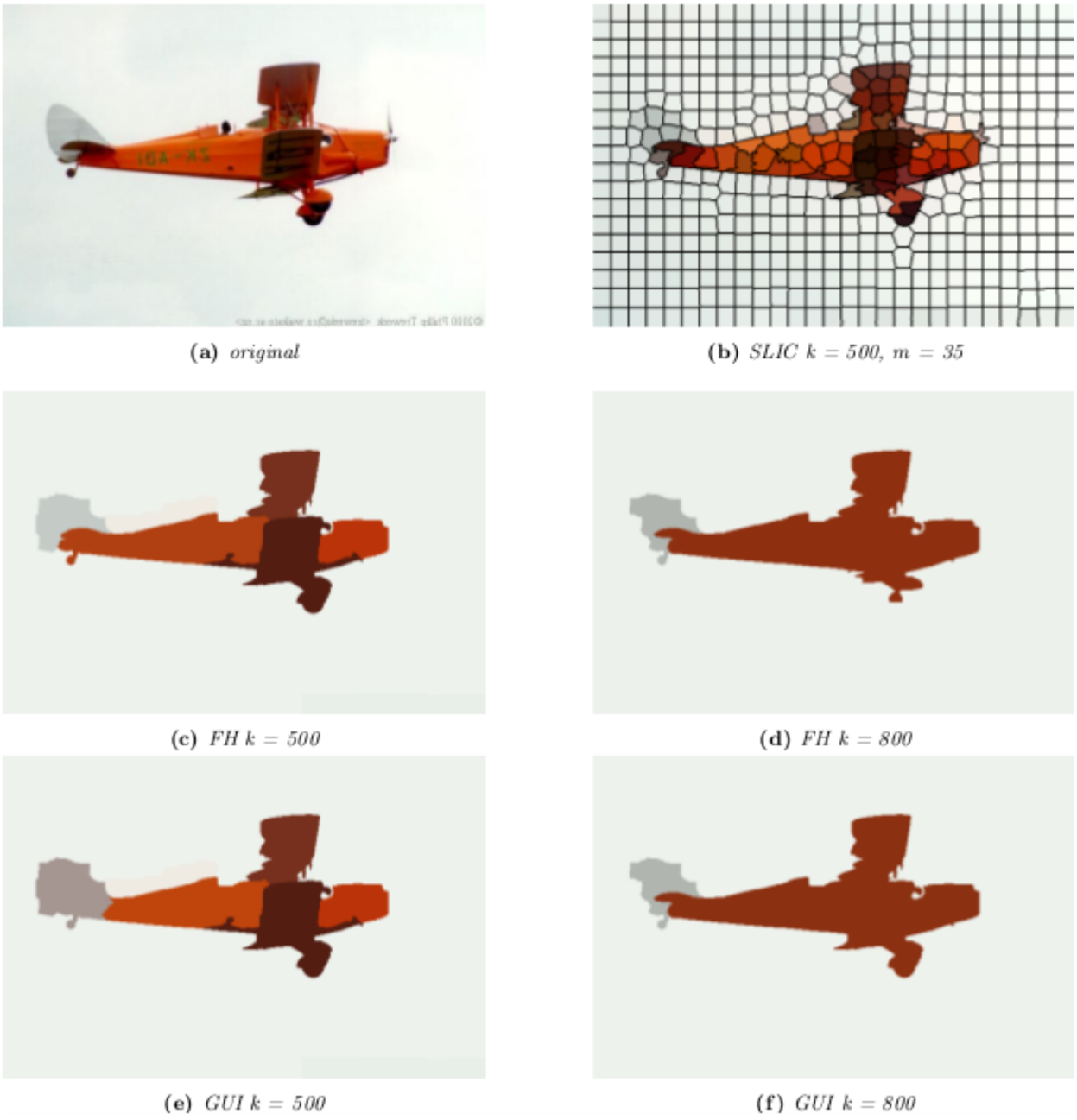



**Figura 5.4:** *Imagens segmentadas a partir de superpixels SLIC (II)*
Imagens originais na primeira coluna, resultados para o método de Felzenszwalb na segunda, e resultados para o método de Guimarães na terceira.



**(a)** *original* **(b)** *SLIC* $k = 500$, $m = 35$

**(c)** *FH* $k = 500$ **(d)** *FH* $k = 800$

**(e)** *GUI* $k = 500$ **(f)** *GUI* $k = 800$

**Figura 5.5:** *Violação do Princípio de Localização pelo método FH*
As segmentações (c)-(f) foram geradas a partir de *superpixels*. Parte do contorno presente na parte inferior do avião na figura (d), a um valor maior de $k$, não está na figura (c).

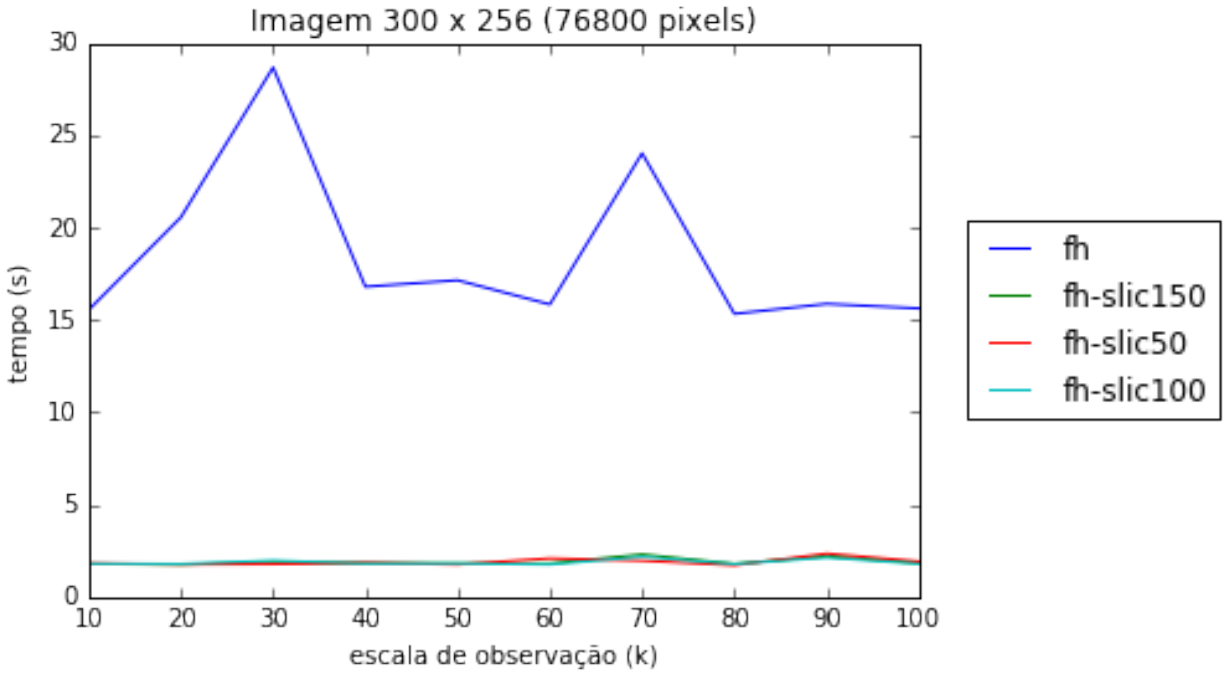
## 5.3 Impacto do pré-processamento

O uso de *superpixels* diminui o número de vértices no grafo da imagem na mesma proporção que o número de regiões criadas com o *SLIC*. Na figura 5.6, são mostrados os tempos de execução dos algoritmos para uma imagem de dimensão $400 \times 256$.

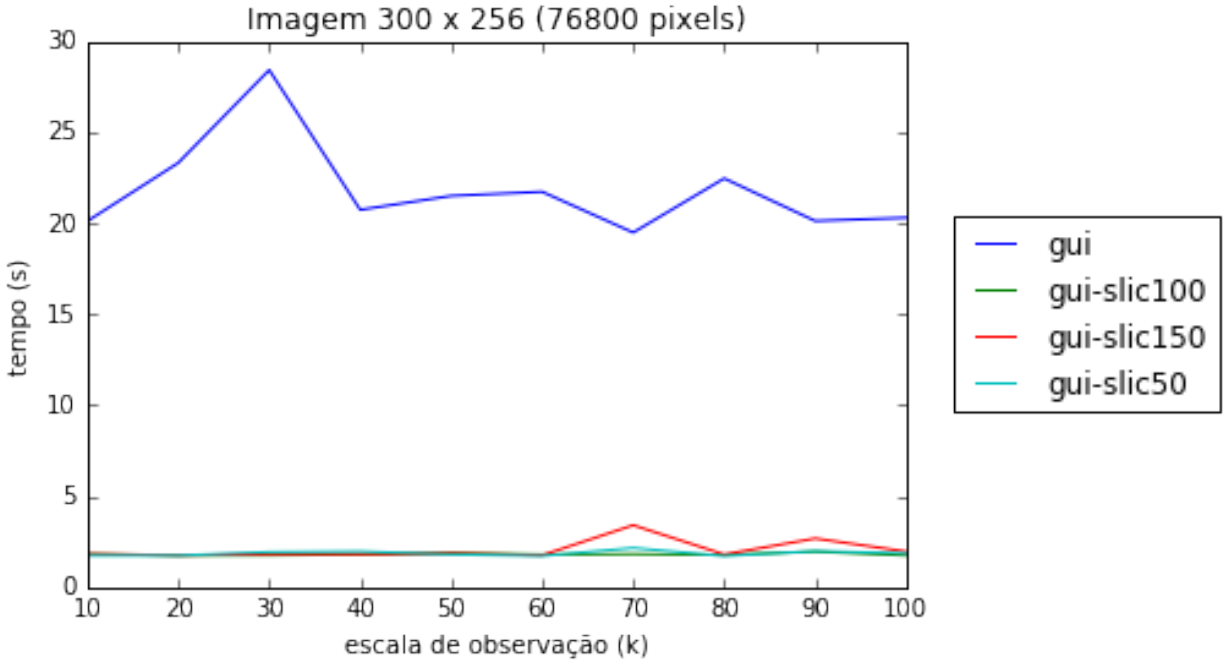
O tempo de execução cai drasticamente com o uso de *superpixels*. No gráfico, o tempo de execução do *SLIC* (nos casos em que o pré-processamento foi feito) também está incluso. A segmentação com o SLIC chega a ser aproximadamente 20 vezes mais rápida do que a feita diretamente em nível de *pixels*. Os picos observados no gráfico estão relacionados a valores de escala de observação que acarretam fusões entre regiões grandes, sendo necessário alterar os rótulos dos componentes de uma delas através da operação *union* na estrutura *Union Find*.

Na figura 5.7, vemos que, apesar de ser mais rápido, o uso do pré-processamento em imagens pequenas não diminui tanto o tempo de execução da segmentação. Entretanto, podemos considerar que, ao segmentar um grande volume de imagens, o impacto dessa redução seja considerável. Visto que o tempo de processamento dos algoritmos de segmentação são proporcionais ao número de *pixels* da imagem, o tempo de execução aumenta conforme aumentam as dimensões da imagem, logo aumentam também com a quantidade de imagens.

Além do impacto no desempenho, verifica-se que os *superpixels* são bons representantes unitários da imagem. Na figura 5.8, observamos uma segmentação mais direta, ou seja, com partições de regiões significativas, para separar o rio, a floresta e o céu. A segmentação em nível de *pixels* requer maior trabalho no ajuste dos parâmetros para particionar regiões relevantes da imagem, pois são mais propensas a gerar regiões disjuntas que pertencem a um mesmo objeto da imagem.



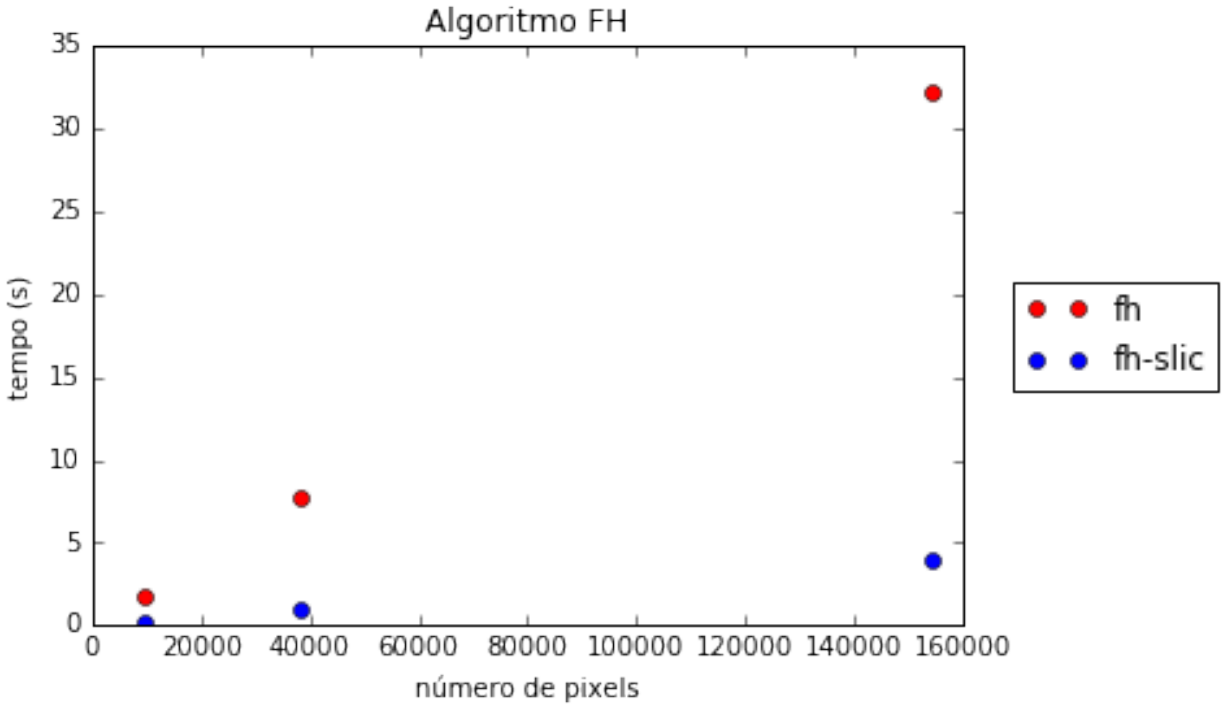

**(a)** *Tempo consumido pelo algoritmo FH*



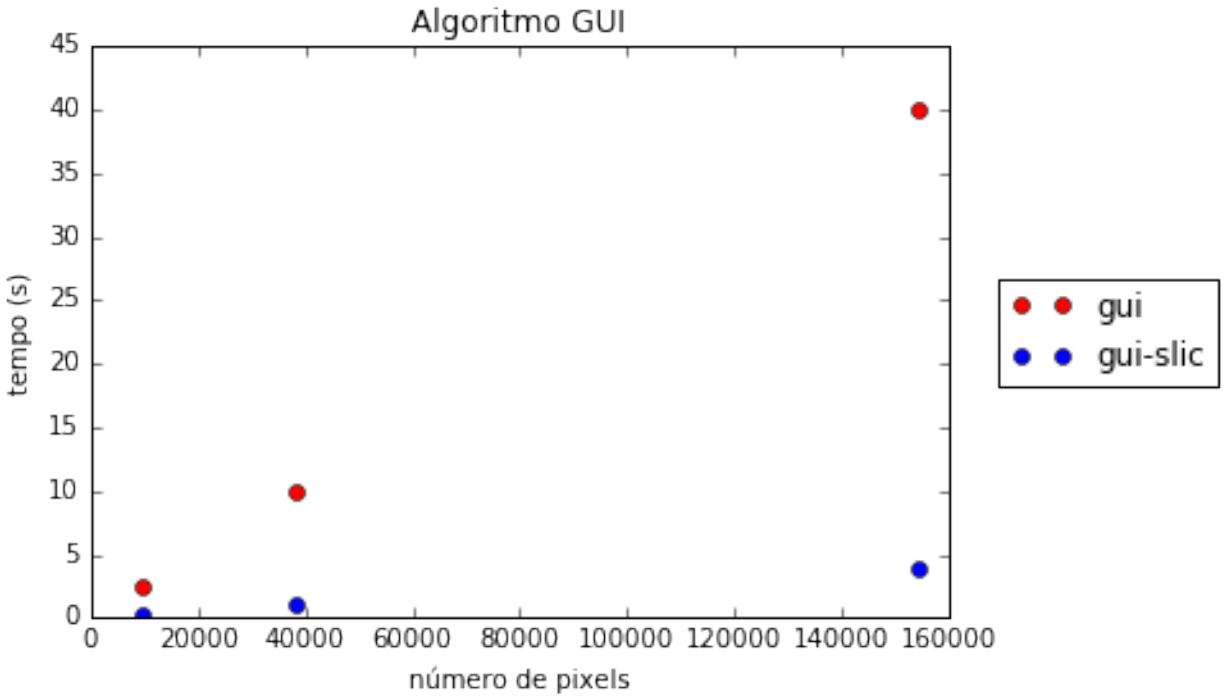

**(b)** *Tempo consumido pelo algoritmo GUI*

**Figura 5.6:** *Duração do processamento dos algoritmos de segmentação conforme a variação da escala de observação*

Os gráficos mostram o desempenho dos algoritmos a partir de *pixels* (*fh* e *gui*) e a partir de *superpixels* - *slic50*, *slic100*, *slic150* - gerados com 50, 100 e 150 regiões, respectivamente. O tempo consumido para o cálculo dos *superpixels* está incluso.

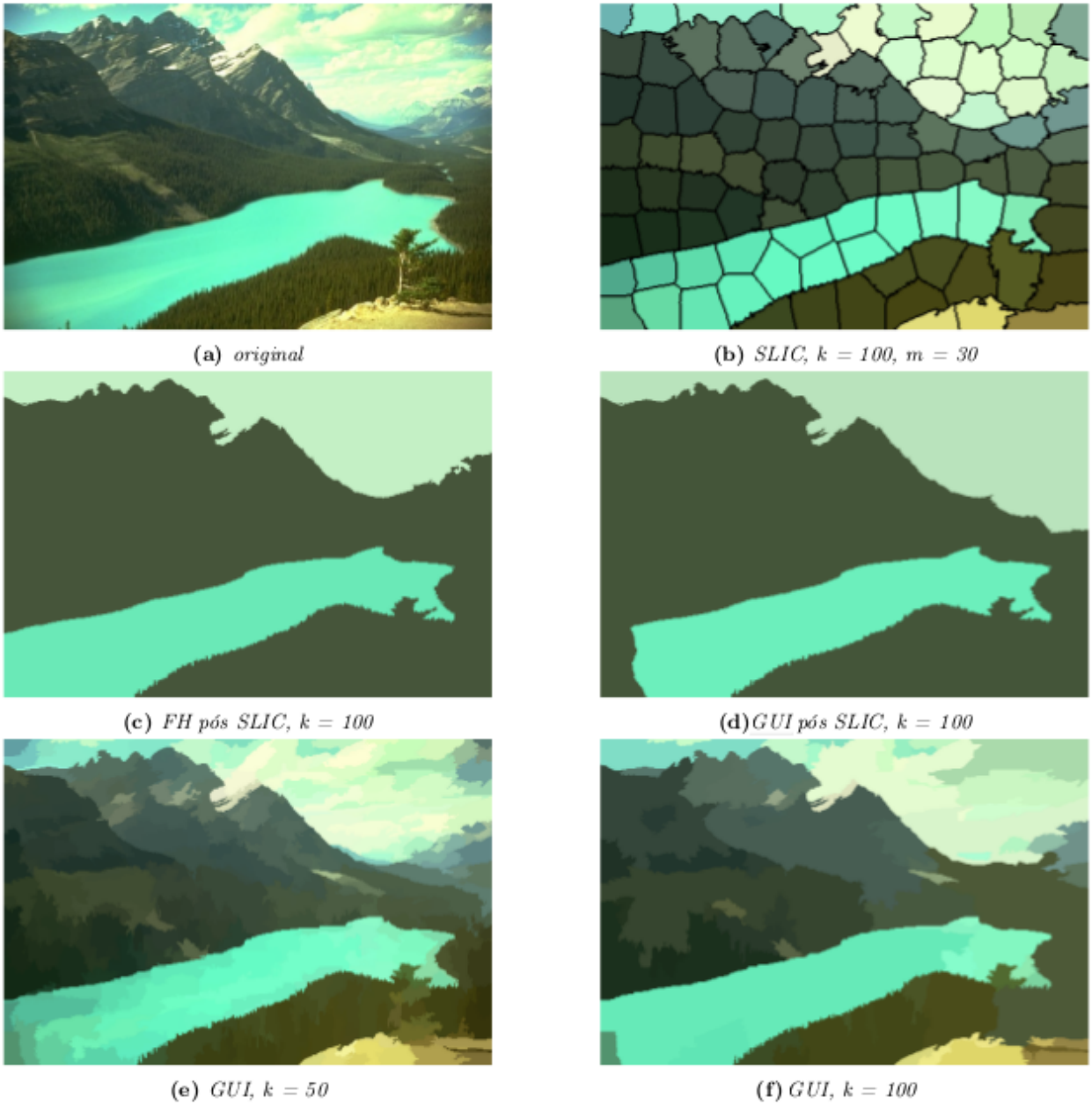



**(a)** *Tempo consumido pelo algoritmo FH*




**(b)** *Tempo consumido pelo algoritmo GUI*

**Figura 5.7:** *Duração do processamento dos algoritmos de segmentação conforme o número de pixels da imagem*



**(a)** *original*

**(b)** *SLIC, k = 100, m = 30*

**(c)** *FH pós SLIC, k = 100*

**(d)** *GUI pós SLIC, k = 100*

**(e)** *GUI, k = 50*

**(f)** *GUI, k = 100*

**Figura 5.8:** *Comparação entre segmentação a partir dos pixels e a partir de superpixels*

# Capítulo 6

# Conclusão

Neste trabalho foi realizada a análise de alguns algoritmos de segmentação que utilizam o arcabouço de grafo com pesos nas arestas, criado a partir da imagem, para particioná-la em regiões menores. Entre eles, foi estudado um algoritmo para supersegmentação - o *SLIC* - capaz de reduzir consideravelmente o tempo de processamento para segmentar regiões significativas da imagem.

A segmentação de imagens é parte essencial da visão computacional, cujo objetivo é automatizar tarefas realizadas pelo sistema visual humano. O particionamento do domínio de imagens está diretamente relacionado às tarefas de extração, compreensão e análise de informações capturadas do mundo real e transformadas em números e símbolos para compreensão das máquinas.

Devido a sua importância, algoritmos de segmentação devem ser computacionalmente eficientes. Os resultados obtidos foram satisfatórios, no geral, mesmo com a utilização de pouca informação a respeito da imagem. O uso de medidas de evidência de contorno, análise de continuidade e de textura seria uma maneira de melhorar a qualidade da segmentação, principalmente em imagens com regiões sobrepostas e objetos distintos com mesmas tonalidades de cores.

Uma das limitações dos métodos estudados é a necessidade de ajustar parâmetros para refinar os resultados. Com o avanço da tecnologia, a produção de dados (principalmente imagens) é cada vez maior, dessa forma, se torna intratável determinar individualmente os valores adequados para cada uma.

Um possível trabalho futuro seria aliar o uso de pré-processamento (*superpixels*) a algoritmos de aprendizagem computacional, que reduzem consideravelmente a participação humana na tarefa e impactam no desempenho desse tipo de trabalho.

# Referências Bibliográficas


**Achanta** *et al.***(2010)** Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurélien Lucchi, Pascal Fua e Sabine Süsstrunk. SLIC Superpixels. Relatório técnico, EPFL. Citado na pág. 3, 11, 21

**Cormen** *et al.***(2009)** Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest e Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edição. ISBN 0262033844, 9780262033848. Citado na pág. 21

**Felzenszwalb e Huttenlocher(2004)** Pedro F. Felzenszwalb e Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181. ISSN 1573-1405. doi: 10.1023/B:VISI.0000022288.19776.77. URL http://dx.doi.org/10.1023/B:VISI.0000022288.19776.77. Citado na pág. 3, 15

**Guimaraes** *et al.***(2012)** Silvio Jamil F. Guimaraes, Jean Cousty, Yukiko Kenmochi e Laurent Najman. *A Hierarchical Image Segmentation Algorithm Based on an Observation Scale*, páginas 116–125. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-34166-3. doi: 10.1007/978-3-642-34166-3_13. URL http://dx.doi.org/10.1007/978-3-642-34166-3_13. Citado na pág. 3, 18, 19, 24

**Jain(1989)** Anil K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. ISBN 0-13-336165-9. Citado na pág. 11

**Martin** *et al.***(2001)** D. Martin, C. Fowlkes, D. Tal e J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. Em *Proc. 8th Int'l Conf. Computer Vision*, volume 2, páginas 416–423. Citado na pág. 21

**Morris** *et al.***(1986)** OJ Morris, M de J Lee e AG Constantinides. Graph theory for image analysis: an approach based on the shortest spanning tree. Em *IEE Proceedings F-Communications, Radar and Signal Processing*, volume 2, páginas 146–152. Citado na pág. 9

**Pérez e Granger(2007)** Fernando Pérez e Brian E. Granger. IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29. ISSN 1521-9615. doi: 10.1109/MCSE.2007.53. URL http://ipython.org. Citado na pág. 21

**Ren e Malik(2003)** Xiaofeng Ren e Jitendra Malik. Learning a classification model for segmentation. Em *ICCV*, volume 1, páginas 10–17. Citado na pág. 3

**Shi e Malik(2000)** Jianbo Shi e Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905. Citado na pág. 9

**Wu e Leahy(1993)** Z. Wu e R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(11):1101–1113. ISSN 0162-8828. doi: 10.1109/34.244673. URL http://dx.doi.org/10.1109/34.244673. Citado na pág. 9