

# Segmentação de Imagens por Grafos: Árvores Geradoras Mínimas e Arborescências

Vitor de Meira Gomes   [ PUC Minas | [vitormeiragomes@outlook.com](mailto:vitormeiragomes@outlook.com) ]

Antônio Drumond Cota de Sousa   [ PUC Minas | [antonio.drumondcs@gmail.com](mailto:antonio.drumondcs@gmail.com) ]

Achille Guérard   [ PUC Minas/EPITA | [achille.guerard15@gmail.com](mailto:achille.guerard15@gmail.com) ]

Laura Menezes Heráclito Alves   [ PUC Minas | [laura.heraclito@gmail.com](mailto:laura.heraclito@gmail.com) ]

Davi Ferreira Puddo   [ PUC Minas | [davifpuddo@gmail.com](mailto:davifpuddo@gmail.com) ]

 Pontifícia Universidade Católica de Minas Gerais, R. Dom José Gaspar, 500, Coração Eucarístico, 30535-901, Belo Horizonte, MG, Brazil.

## Resumo

**Abstract.** Este trabalho apresenta um estudo prático de segmentação de imagens baseado em modelagem por grafos, com ênfase na comparação entre métodos fundamentados em árvores geradoras mínimas e arborescências direcionadas. A imagem é representada como um grafo ponderado cujos vértices correspondem a pixels e cujas arestas capturam dissimilaridades derivadas de cor e gradiente após etapas de pré-processamento (conversão para tons de cinza, suavização Gaussiana e operador de Sobel).

**Keywords:** Segmentação de Imagens; Grafos; Árvores Geradoras Mínimas; Arborescências; Algoritmo de Felzenszwalb; Algoritmo de Chu–Liu/Edmonds; Processamento Digital de Imagens; C++; Pré-processamento

## 1 Introdução

A segmentação de imagens pode ser formulada de maneira natural por meio de grafos, nos quais cada pixel corresponde a um vértice e suas vizinhanças definem arestas ponderadas por medidas de dissimilaridade. Essa modelagem permite explorar algoritmos clássicos de particionamento que utilizam estruturas de árvores geradoras mínimas e arborescências direcionadas, conforme previsto no escopo deste trabalho. Dada uma imagem de entrada, aplica-se um conjunto de transformações preliminares, composto por conversão para tons de cinza, suavização e cálculo de gradientes, com o objetivo de extrair atributos relevantes à construção do grafo. Em seguida, o grafo ponderado resultante combina diferenças de cor e intensidade de gradiente para definir pesos que capturam transições estruturais significativas na imagem. Sobre essa estrutura são implementados dois métodos complementares de segmentação: (i) um procedimento baseado em Kruskal e no critério de fusão proposto por Felzenszwalb e Huttenlocher, que utiliza propriedades da árvore geradora mínima para agrupar pixels de acordo com coerência interna de cor e fronteiras definidas por pesos; e (ii) uma abordagem fundamentada em arborescências, inspirada no algoritmo de Edmonds [1967], que resolve ciclos por meio de contrações sucessivas para obter uma árvore mínima enraizada e, a partir dela, componentes segmentados. As duas técnicas permitem comparar, de forma experimental, diferenças estruturais entre métodos tradicionais para grafos.

## 2 Referencial Teórico

Esta seção apresenta os fundamentos que sustentam a modelagem por grafos aplicada à segmentação de imagens. Inicialmente descrevemos como a imagem é representada como grafo e como são definidos os pesos das arestas. Em seguida,

discutimos dois paradigmas centrais neste trabalho: os métodos baseados em árvores geradoras mínimas, com ênfase no critério de Felzenszwalb e Huttenlocher, e as abordagens por arborescências mínimas segundo Edmonds.

### 2.1 Modelagem por Grafos

Em uma formulação baseada em grafos, cada pixel da imagem é representado por um vértice do grafo  $G = (V, E)$ , e cada vértice (exceto aqueles nas beiradas da imagem) será conectado a oito vizinhos, sendo quatro arestas para os pixels à direita, esquerda, cima e baixo do vértice, e mais quatro para as diagonais entre essas direções principais. O peso  $w(u, v)$  associado a cada aresta quantifica a dissimilaridade entre dois pixels e pode ser definida pela seguinte fórmula:

$$w = \sqrt{(v_{1r} - v_{2r})^2 + (v_{1g} - v_{2g})^2 + (v_{1b} - v_{2b})^2}$$

Sendo  $v1$  e  $v2$  os pixels (vértices) comparados e  $r, g, b$  seus canais de cor *red*, *green* e *blue*.

Etapas de pré-processamento, como suavização Gaussiana, cálculo do gradiente e conversão para tons de cinza quando adequado, reduzem o ruído da imagem e tornam os pesos mais discriminativos, levando à criação de segmentos mais uniformes e bem demarcados.

### 2.2 Método de Felzenszwalb

O método de Felzenszwalb e Huttenlocher [2004] utiliza uma estratégia de união de componentes semelhante ao algoritmo de Kruskal, processando as arestas em ordem crescente de peso. Para cada componente  $C$ , define-se a *diferença interna*, denotada por  $\text{Int}(C)$ , que corresponde ao maior peso da árvore geradora mínima restrita a  $C$ .

A condição de fusão entre dois componentes  $C_1$  e  $C_2$  dado dois vértice  $(u, v)$ , um em cada componente, com a aresta de menor de peso  $w(u, v)$  entre  $C_1$  e  $C_2$  é dada por:

$$w(u, v) \leq \text{MInt}(C_1, C_2),$$

$$\text{MInt}(C_1, C_2) = \min\left(\text{Int}(C_1) + \frac{k}{|C_1|}, \text{Int}(C_2) + \frac{k}{|C_2|}\right), \quad (1)$$

onde  $k > 0$  é um parâmetro de escala que controla a tendência à fusão, favorecendo componentes maiores para valores mais elevados.

O algoritmo mantém estrutura Union–Find para operações de união, resultando em custo quase linear após a ordenação das arestas ( $O(|E| \log |E|)$ ). Entre as limitações práticas destacam-se a sensibilidade ao parâmetro  $k$  e a dificuldade em preservar contornos finos em regiões de baixo contraste, o que motiva combinar diferenças de cor e gradiente e aplicar pós-processamentos para eliminar componentes muito pequenos.

## 2.3 Arborescências Mínimas: Edmonds

Arborescências mínimas tratam grafos direcionados e buscam uma árvore enraizada de custo mínimo que alcance todos os vértices. O algoritmo de Edmonds constrói essa estrutura selecionando, para cada vértice (exceto a raiz), a aresta de menor custo incidente. A presença de ciclos é resolvida por contrações sucessivas, seguidas da reconstrução da estrutura final. A forma clássica apresenta complexidade  $O(|V| \cdot |E|)$ , concentrada nas etapas de detecção e contração de ciclos. Variações posteriores, como as de Gabow, reduzem o custo prático e tornam o método utilizável em grafos moderados.

Para aplicação em segmentação, o grafo originalmente não direcionado pode ser convertido em grafo direcionado, por exemplo atribuindo a cada par  $u-v$  a direção correspondente ao menor peso. Essa representação permite construir hierarquias de segmentação baseadas em estruturas enraizadas. As vantagens incluem a capacidade de modelar dependências direcionais e relações hierárquicas entre regiões; entretanto, o custo computacional das etapas de contração exige otimizações, restrição do número de arestas ou uso de aproximações em imagens muito grandes.

## 3 Implementação

O pipeline implementado é composto pelas seguintes etapas: leitura da imagem em formato PPM; pré-processamento (conversão opcional para tons de cinza, suavização Gaussiana e cálculo do gradiente via Sobel); construção do grafo ponderado; aplicação dos métodos de segmentação (MST/Felzenszwalb e Arborescência/Edmonds); pós-processamento e recoloração; e escrita das imagens segmentadas. A implementação utiliza representação local de vizinhança (arestas 8-adjacentes) e pesos que combinam diferença de cor e magnitude do gradiente.

### 3.1 Detalhes da Implementação

O projeto foi desenvolvido inteiramente em C++, com foco em desempenho e gestão direta de memória, requisito importante para aplicações de processamento de imagens. O código está organizado em módulos que separam claramente: entrada/saída e pré-processamento, construção de grafos ponderados, algoritmos de segmentação (baseados em MST-/Kruskal e em arborescências) e funções utilitárias.

#### 3.1.1 Estrutura do Projeto

A estrutura modular do código inclui as seguintes responsabilidades principais:

- **ppm.h**: leitura e escrita do formato PPM, filtros de pré-processamento (grayscale, Gaussian blur, Sobel), utilitários para salvar imagens segmentadas por rótulo e rotina auxiliar de segmentação baseada em lista de arestas.
- **Graph.h**: definição das classes de grafo (não direcionado e ponderado), representação de pixels em RGB, construção de grafos a partir de matrizes PPM (`from_ppm_matrix`) ou de cor + gradiente (`from_color_and_gradient`) e métodos para converter o grafo de volta para matriz PPM (`to_ppm_matrix`).
- **felzenszwalb.h**: implementação do procedimento de Kruskal adaptado para segmentação (critério de Felzenszwalb), com estrutura de heap mínimo para ordenação de arestas e uma estrutura Union–Find especializada para gerir componentes e diferenças internas.
- **arborescence.h** e **edmonds.h**: definição de `DirectedGraph`, estruturas de arestas direcionadas, representação do resultado de arborescência e classe `EdmondsAlgorithm`, que implementa a versão utilizada para hierarquias e segmentação baseada em arborescências.
- **main.cc**: orquestra o pipeline experimental, aplicando pré-processamento, criando os grafos, executando `kruskal_segmentation` e o algoritmo de Edmond, recolorindo componentes e gravando as imagens de saída.

#### 3.1.2 Especificações da Implementação MST (Felzenszwalb)

A implementação do método baseado em Felzenszwalb e Huttenlocher Felzenszwalb and Huttenlocher [2004] segue a ideia clássica de ordenar arestas e unir componentes segundo um critério adaptativo.

Fluxo principal:

1. Construção das arestas: para cada pixel  $i$  são consideradas adjacências locais (incluindo diagonais quando aplicável). Para cada vizinho  $j$  calcula-se a diferença de cor (ou combinação cor+gradiente) e gera-se uma aresta  $(i, j, w)$ .
2. Ordenação: todas as arestas são inseridas em uma `priority_queue` (min-heap) e processadas em ordem crescente de peso.
3. Estrutura Union–Find: utiliza-se uma tabela interna com vetores paralelos para ancestral, rank/tamanho e diferença interna para cada vértice, permitindo encontrar

a raiz, unir por rank e manter  $\text{Int}(C)$  (diferença interna) de cada componente.

4. Regra de fusão: para uma aresta de peso  $w$  que conecta componentes  $C_1$  e  $C_2$ , calcula-se

$$\text{MInt}(C_1, C_2) = \min(\text{Int}(C_1) + k/|C_1|, \text{Int}(C_2) + k/|C_2|), \quad (2)$$

e efetua-se a fusão se  $w \leq \text{MInt}(C_1, C_2)$ . Na fusão, atualizam-se ancestral, rank e Int do componente resultante.

5. Construção do resultado: as arestas aceitas compõem a estrutura de saída. Em seguida, cada vértice é recolorido de acordo com o representante de sua componente, preservando as cores originais para facilitar a visualização.

Notas de implementação:

- A construção das arestas explora a topologia regular da imagem (indexação linear com uso da width para calcular vizinhanças), reduzindo custo e complexidade de iteração.
- A ordenação de arestas é a etapa dominante em custo, com complexidade prática  $O(|E| \log |E|)$ ; as operações de união e busca em Union–Find têm custo amortizado quase linear.
- A informação de cor original é preservada para a fase final de recoloração dos componentes, produzindo resultados visualmente interpretáveis.

### 3.1.3 Especificação da implementação do Algoritmo de Edmonds

O algoritmo de Edmonds [1967] foi utilizado para gerar uma Arborescência Geradora Mínima, garantindo conectividade a partir de uma raiz com o menor custo acumulado, antes de segmentá-la.

Principais aspectos:

- Seleção Gulosa: a etapa inicial itera sobre todos os vértices (exceto a raiz) para selecionar a aresta incidente de menor peso. Funções auxiliares validam se esse subgrafo preliminar é acíclico; caso positivo, a solução ótima é retornada imediatamente.
- Contração de Ciclos: diferentemente de abordagens para grafos não direcionados, quando um ciclo é detectado, o algoritmo o condensa em um *super-vértice*. As arestas externas que incidem no ciclo são reponderadas (*reweighting*) para refletir o custo marginal de substituir uma aresta interna do ciclo por uma externa.
- Expansão e Reconstrução: após a resolução recursiva no grafo contraído, o módulo realiza a etapa de expansão (*unrolling*). O super-vértice é desfeito e o ciclo original é “quebrado” removendo-se a aresta interna que conflita com a conexão externa escolhida, restaurando a topologia válida.
- Integridade Estrutural: o método assegura que cada nó possua exatamente um pai (exceto a raiz), sendo fundamental para aplicações que exigem hierarquias estritas de dependência ou fluxo direcionado sem retornos.

### 3.1.4 Métodos de Pré-processamento

Antes da construção do grafo ponderado, a imagem passa por uma sequência de transformações que visam reduzir ruído, destacar características estruturais e melhorar a discriminação entre regiões durante a segmentação.

**Conversão para Escala de Cinza** A conversão para tons de cinza elimina a informação de cromaticidade, reduzindo a dimensionalidade dos dados de três canais RGB para um único canal de intensidade. A conversão implementada utiliza a média aritmética dos três canais:

$$I_{\text{gray}}(x, y) = \frac{R(x, y) + G(x, y) + B(x, y)}{3}, \quad (3)$$

onde  $R$ ,  $G$  e  $B$  representam os valores dos canais vermelho, verde e azul, respectivamente.

**Suavização Gaussiana** A suavização Gaussiana of Edinburgh [2024] reduz flutuações de alta frequência causadas por ruído de aquisição ou compressão. A implementação utiliza um kernel discreto de convolução  $3 \times 3$  com distribuição de pesos proporcional à função Gaussiana:

$$G(x, y) = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}. \quad (4)$$

A aplicação iterativa (controlada pelo parâmetro *passes*) permite ajustar o grau de suavização: valores maiores produzem imagens mais homogêneas, adequadas para segmentações em larga escala, enquanto valores menores preservam detalhes finos.

**Detecção de Bordas: Operador de Sobel** O operador de Sobel estima o gradiente local de intensidade, evidenciando transições abruptas que correspondem a fronteiras entre regiões. O método calcula aproximações das derivadas parciais nas direções horizontal e vertical utilizando dois kernels de convolução  $3 \times 3$ :

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}. \quad (5)$$

A magnitude do gradiente em cada pixel  $(x, y)$  é obtida por:

$$|\nabla I(x, y)| = \sqrt{G_x^2 + G_y^2}. \quad (6)$$

Regiões com gradiente elevado indicam fronteiras potenciais, enquanto regiões de gradiente baixo sugerem homogeneidade interna.

**Construção do Grafo com Ponderação Combinada** Após o pré-processamento, dois grafos são construídos: um baseado exclusivamente em diferenças de cor (a partir da imagem original suavizada) e outro combinando cor e informação de gradiente. A função `from_color_and_gradient` implementa a ponderação híbrida:

$$w(u, v) = \alpha \cdot \|C(u) - C(v)\|_2 + \beta \cdot \frac{|\nabla I(u)| + |\nabla I(v)|}{2}, \quad (7)$$

onde  $C(u)$  e  $C(v)$  representam os vetores RGB dos pixels  $u$  e  $v$ ,  $\|\cdot\|_2$  denota a distância euclidiana no espaço de cor,  $|\nabla I|$  é a magnitude do gradiente e  $\alpha, \beta$  são coeficientes de escala.

## 4 Resultados dos Testes



(a) Original Image, no filters or changes applied.



(b) Sobel operator applied.



(c) Felzenswalb's Segmentation.



(d) Edmond's Segmentation.

**Figura 1.** Comparison of segmentation results. Image (a) is the original. Image (b) shows the result of Felzenswalb's algorithm. Image (c) is the result of Edmond's algorithm, and (d) the image with the Sobel Operator applied, which was used to highlight edges in the image.

O algoritmo de Felzenswalb Felzenswalb and Huttenlocher [2004], que opera sobre grafos não direcionados e o algoritmo de Edmonds, que opera sobre grafos direcionados, foram ambos utilizados para segmentar a imagem, e os resultados alcançados por eles são visualmente diferentes. A imagem gerada pelo algoritmo de Felzenswalb, por exemplo, preserva melhor as sombras da imagem, e gera mais segmentos. A imagem gerada pelo algoritmo de Edmonds Edmonds [1967], por outro lado, é visualmente mais "uniforme", possivelmente devido pela segmentação pela média de cores dos

vizinhos da área analizada. Isso implica também uma menor "saturação" o que torna a imagem menos colorida. Por fim, o Felzenswalb acabou por ter uma imagem melhor segmentada, mesmo os resultados da arborescência sendo mais consisos e homogêneos.

## 5 Conclusão

Como visto durante o artigo, e durante a análise das imagens, é possível notar que a principal diferença visual entre os métodos é a escolha das cores de cada componente, devido ao critério de seleção de cada algoritmo, com o método de Felzenswalb sendo levemente mais segmentado, quando fornecidos com os mesmos parâmetros. O uso de grafos direcionados para a segmentação de imagens apresenta resultados mais significativos, quando comparado com grafos não-direcionados, principalmente em situações onde há a necessidade de destacar uma direção ou gradiente, fazendo com que o caso apresentado durante o artigo, a segmentação de componentes de cores próximas, não apresente diferenças relevantes entre os algoritmos.

## Declarações

### Authors' Contributions

- **Laura Menezes heráclito Alves:** Implementação segmentação de imagens por Arborescência.
- **Vitor de Meira Comes:** Descrição do artigo e pesquisa de artigos e bases teóricas.
- **Antônio Drumond Cota de Sousa:** Implementação da estrutura de grafos a partir do arquivo de imagem e etapas de pré-processamento.
- **Achille Guérard:** Descrição do artigo e apoio técnico
- **Davi Ferreira Puddo:** Implementação segmentação por Árvore Geradora Mínima.

### Availability of data and materials

O código desenvolvido está disponibilizado abertamente no seguinte repositório do GitHub:

[https://github.com/AntonioDrumond/grafos/tree/main/Trabalho\\_2](https://github.com/AntonioDrumond/grafos/tree/main/Trabalho_2)

## Referências

- Edmonds, J. (1967). Optimum branchings. *Journal of Research of the National Bureau of Standards Section B*, 71B(4):233–240.
- Felzenswalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181. DOI: 10.1023/B:VISI.0000022288.19776.77.
- of Edinburgh, U. U. (2024). Gaussian smoothing. <https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm>. Acessado em: 2024.