# Análise Comparativa de Algoritmos de Segmentação de Imagens Baseados em Grafos

**Gabriel Cunha, Kaiky França, Rickerson Antônio, Samuel Horta**

[1]Instituto de Ciências Exatas e Informática –
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
Belo Horizonte – MG – Brazil

***Abstract.*** *Image segmentation is a fundamental process in computer vision, where an image is partitioned into multiple segments or regions. Graph-based approaches model this problem by representing pixels as vertices and the relationship between them as weighted edges, turning segmentation into a graph partitioning problem. This paper presents a comparative analysis of two prominent graph-based segmentation methods: the Minimum Spanning Tree (MST) based approach by Felzenszwalb and Huttenlocher, and the Minimum Path (Image Foresting Transform) based approach by Falcão et al. We describe the implementation of both algorithms and evaluate their behavior on grayscale and color images, highlighting their theoretical similarities, practical differences, and overall performance. This work was developed as a final project for the "Graph Theory and Computability"course at PUC Minas.*

***Resumo.*** *A segmentação de imagens é um processo fundamental em visão computacional, no qual uma imagem é particionada em múltiplos segmentos ou regiões. Abordagens baseadas em grafos modelam esse problema representando pixels como vértices e a relação entre eles como arestas ponderadas, tornando a segmentação um problema de particionamento de grafos. Este artigo apresenta uma análise comparativa de dois proeminentes métodos de segmentação baseados em grafos: a abordagem baseada em Árvore Geradora Mínima (AGM) de Felzenszwalb e Huttenlocher, e a abordagem baseada em Caminho Mínimo (Image Foresting Transform) de Falcão et al. Descrevemos a implementação de ambos os algoritmos e avaliamos seu comportamento em imagens em níveis de cinza e coloridas, destacando suas semelhanças teóricas, diferenças práticas e desempenho geral. Este trabalho foi desenvolvido como projeto final da disciplina de Teoria de Grafos e Computabilidade da PUC Minas.*

## 1. Introdução

A análise e interpretação de imagens digitais são tarefas essenciais em inúmeras áreas da computação, como reconhecimento de padrões, processamento de imagens médicas e visão de robô. Um dos passos mais críticos e desafiadores nesse processo é a segmentação de imagens, que consiste em dividir uma imagem em regiões distintas, onde cada região agrupa pixels com características similares. Este trabalho, desenvolvido no âmbito da disciplina de Teoria de Grafos e Computabilidade, foca em explorar como os fundamentos da teoria dos grafos podem ser aplicados para resolver este problema de forma eficiente.

A modelagem da segmentação de imagens por meio de grafos oferece uma representação poderosa e flexível. Nesse paradigma, a imagem é interpretada como um

grafo $G = (V, E)$, onde o conjunto de vértices $V$ corresponde ao conjunto de pixels da imagem. O conjunto de arestas $E$ representa a relação de vizinhança entre os pixels. A cada aresta é associado um peso, que geralmente quantifica a dissimilaridade entre os pixels conectados, como a diferença de cor, intensidade ou textura. Dessa forma, o problema de segmentar a imagem torna-se equivalente ao problema de particionar os vértices do grafo em conjuntos disjuntos, que corresponderão às regiões segmentadas.

Dentre as diversas estratégias para resolver problemas de particionamento de grafos no contexto da segmentação de imagens, destacam-se três abordagens principais: métodos baseados em Árvore Geradora Mínima (AGM), algoritmos de Fluxo em Rede e técnicas de Caminho Mínimo. Cada uma dessas metodologias explora diferentes propriedades dos grafos para agrupar os pixels de maneira coesa e significativa.

O objetivo deste trabalho é implementar, analisar e comparar duas dessas metodologias. A primeira é a abordagem baseada em AGM, proposta por Felzenszwalb e Huttenlocher, que se destaca por sua eficiência e capacidade de capturar percepções globais na imagem. A segunda é a técnica baseada em Caminho Mínimo, formalizada como a Image Foresting Transform por Falcão, Stolfi e Lotufo, que transforma a segmentação em um problema de competição entre caminhos de custo mínimo a partir de sementes. Ao final, apresentamos uma discussão sobre as semelhanças e diferenças entre as duas técnicas, bem como o comportamento de nossas implementações em imagens em níveis de cinza e coloridas.

## 2. Metodologia

A abordagem metodológica deste trabalho foi dividida em três fases fundamentais: implementação dos algoritmos, delineamento do ambiente experimental e, por fim, a análise comparativa dos resultados. Cada fase foi projetada para cumprir os objetivos de avaliar e contrastar as duas técnicas de segmentação de imagens baseadas em grafos.

### 2.1. Implementação dos Algoritmos

A primeira fase consistiu na implementação dos dois algoritmos de segmentação selecionados, utilizando a linguagem de programação C++. As implementações foram desenvolvidas com base nos fundamentos teóricos e nos pseudocódigos apresentados nos artigos seminais de referência. Para a leitura e escrita de imagens, foi utilizada a biblioteca single-header stb_image.h, que facilita a manipulação dos dados dos pixels.

- **Abordagem por Árvore Geradora Mínima (AGM):** Baseada no trabalho de Felzenszwalb e Huttenlocher, focando no critério de fusão de regiões para particionar o grafo da imagem.
- **Abordagem por Caminho Mínimo:** Fundamentada na *Image Foresting Transform* (IFT) de Falcão, Stolfi e Lotufo, que modela a segmentação como uma competição de caminhos de custo mínimo a partir de sementes.

O desenvolvimento buscou aderir fielmente às definições dos autores para garantir uma base de comparação justa e precisa.

### 2.2. Ambiente Experimental

Para a etapa de testes, foi estabelecido um ambiente experimental para avaliar o comportamento dos algoritmos em diferentes cenários. Foi selecionado um conjunto de imagens de

teste, categorizadas de acordo com sua complexidade visual e cromática, a fim de analisar a robustez e a aplicabilidade de cada método:

- **Imagens Simples:** Imagens com poucas cores e regiões bem definidas e homogêneas.
- **Imagens de Complexidade Mediana:** Imagens com maior variedade de cores, texturas e detalhes moderados.
- **Imagens de Alta Complexidade:** Imagens com gradientes de cor suaves, alto nível de detalhes e texturas complexas.

Adicionalmente, os testes foram realizados tanto em versões coloridas quanto em níveis de cinza de cada imagem, para investigar como cada algoritmo lida com a ausência e a presença de informação de cor.

### 2.3. Análise Comparativa

A fase final da metodologia concentrou-se na análise e comparação dos resultados obtidos. A avaliação foi conduzida sob duas perspectivas:

- **Análise Qualitativa:** Inspeção visual da qualidade das segmentações produzidas por cada algoritmo em cada imagem de teste. Verificou-se a coerência dos segmentos gerados em relação aos objetos e regiões semanticamente relevantes nas imagens.
- **Análise de Comportamento:** Estudo das diferenças e semelhanças práticas entre os métodos, incluindo a sensibilidade aos parâmetros de entrada, a tendência a supersegmentar ou subsegmentar regiões, e o desempenho computacional observado.

## 3. Abordagem por Árvore Geradora Mínima

A primeira abordagem implementada foi a de Felzenszwalb e Huttenlocher, que é notável por sua eficiência e qualidade nos resultados.

### 3.1. Pré-processamento dos Dados

Antes da segmentação, um pré-processamento é aplicado à imagem de entrada através do filtro ”Gaussian Blur”. Este passo, aplicado tanto à versão colorida quanto à de escala de cinza, é fundamental para suavizar a imagem, removendo ruídos e pequenas variações de textura que poderiam levar a uma segmentação excessiva e pouco significativa.

### 3.2. Construção do Grafo

Na construção do grafo, a estrutura representa cada pixel como um vértice. As arestas, por sua vez, são geradas pela classe `Segmenter`, que conecta cada pixel aos seus vizinhos adjacentes. O peso de cada aresta é a medida da dissimilaridade entre os dois pixels que ela conecta. Para imagens em escala de cinza, utiliza-se a diferença absoluta de intensidade; para imagens coloridas, a distância euclidiana no espaço de cores RGB.

### 3.3. Ordenação das Arestas

Um passo central do algoritmo, que o alinha à lógica de construção de uma Árvore Geradora Mínima (MST), é a ordenação de todas as arestas do grafo em ordem crescente de seus pesos. Desta forma, o algoritmo processa as conexões dos pixels mais similares primeiro, avançando para os menos similares.

### 3.4. Fusão de Componentes

O núcleo do algoritmo reside em seu critério para fundir dois componentes distintos. Diferente da construção de uma MST clássica, onde arestas que não formam ciclos são sempre adicionadas, este método utiliza um predicado adaptativo para decidir se dois componentes, $C_1$ e $C_2$, devem ser unidos por uma aresta de peso $w$. A decisão é tomada ao iterar sobre as arestas ordenadas, avaliando se a diferença *entre* os componentes ($w$) é pequena em comparação com a diferença *dentro* de cada um.

A diferença interna de um componente, $Int(C)$, é o maior peso de aresta dentro daquele componente, valor armazenado no vetor `max_internal_edge` da classe `Disjoint`. A fusão ocorre se $w$ satisfaz a seguinte condição:

$$w \leq \min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2)) \tag{1}$$

Onde $\tau(C)$ é um limiar baseado no tamanho do componente:

$$\tau(C) = \frac{k}{|C|} \tag{2}$$

Nesta fórmula, $|C|$ é o tamanho do componente (vetor `component_size`) e $k$ é a constante de escala, ajustada em $k = 500.0$ em nossos testes. Se a condição da Equação 1 é satisfeita, os componentes são fundidos pelo método `unite_sets`, e a gestão eficiente dos conjuntos é realizada pela estrutura de dados *Disjoint Set Union (DSU)*.

### 3.5. O Papel do Parâmetro de Escala *k*

O parâmetro $k$ governa diretamente a granularidade da segmentação. Valores altos de $k$ aumentam o limiar de fusão, resultando em segmentos maiores e em menor quantidade. Inversamente, valores baixos de $k$ tornam o algoritmo mais restritivo, levando a segmentos menores e mais numerosos, o que pode ser útil para capturar detalhes finos.

### 3.6. Geração e Visualização da Saída

Ao final do processo, a estrutura DSU contém a partição final da imagem. O método `segmentationVisualization` traduz essa estrutura em uma imagem colorida, atribuindo uma cor RGB única e aleatória a cada componente (segmento) para visualização.

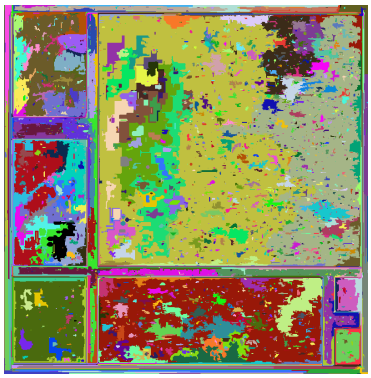
### 3.7. Resultados da AGM

## 4. Abordagem por Caminho Mínimo (Dijkstra)

A segunda metodologia implementada aborda a segmentação como um problema de competição por caminhos de custo mínimo, utilizando uma adaptação do clássico algoritmo de Dijkstra, conforme detalhado nos arquivos `dijkstra.cpp` e `dijkstraSolution.cpp`.

### 4.1. Pré-processamento com Gradiente

Antes da execução, a imagem de entrada é pré-processada pela função `gradient::generateGradient` para gerar uma representação de seu gradiente via operador Sobel. A imagem resultante, em tons de cinza, é fundamental, pois os custos das arestas no grafo serão baseados nas intensidades desta imagem, tornando mais ”caros”os caminhos que cruzam bordas fortes.



*

Imagem Simples

*

Imagem Média

*

Imagem Complexa

### 4.2. Seleção Interativa de Sementes

A implementação permite ao usuário guiar o processo através de uma interface de linha de comando (CRUD) para gerenciar as ”sementes”(pixels iniciais). O resultado da segmentação é inteiramente dependente da posição e quantidade dessas sementes, o que confere ao usuário controle direto sobre o resultado.

### 4.3. Execução do Algoritmo de Dijkstra

O núcleo da segmentação é a classe `CM`, que implementa o algoritmo de Dijkstra.

1. **Inicialização:** As sementes definidas pelo usuário são inseridas em uma fila de prioridade (`std::priority_queue`) com custo zero, enquanto os demais pixels recebem custo infinito.
2. **Processo Iterativo:** O algoritmo remove o pixel de menor custo da fila e, para cada um de seus vizinhos, calcula um novo custo de caminho usando uma função aditiva (*f_sum*). Se um caminho mais barato for encontrado para um vizinho, seu custo e rótulo são atualizados, e ele é inserido na fila.
3. **Término:** O processo termina quando a fila está vazia, e cada pixel foi atribuído à semente do caminho de menor custo.

### 4.4. Geração da Saída Visual

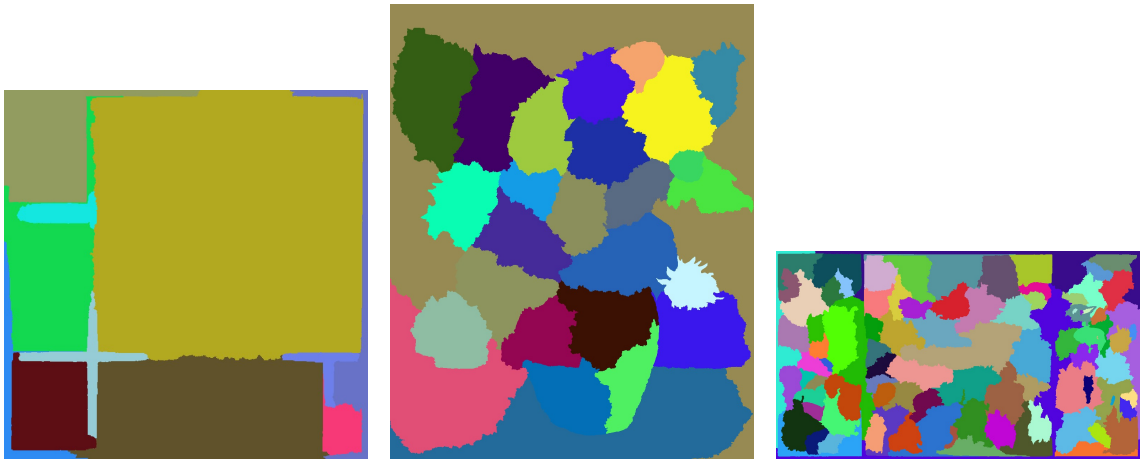
Após a execução, a matriz de rótulos é convertida em uma imagem colorida, onde cada rótulo de semente recebe uma cor RGB aleatória e distinta. O resultado final particiona a imagem de acordo com a ”proximidade”de cada pixel às sementes, respeitando as barreiras impostas pelo gradiente.

### 4.5. Resultados da Abordagem por Menor Caminho

## 5. Conclusão

## 6. Conclusão

Este trabalho realizou a implementação e a análise comparativa de duas abordagens canônicas para a segmentação de imagens baseada em grafos: o método de Árvore Geradora Mínima (AGM) de Felzenszwalb e Huttenlocher e a técnica de Caminho Mínimo,



* Imagem Simples

* Imagem Média

* Imagem Complexa

implementada com o algoritmo de Dijkstra e inspirada na Image Foresting Transform. Ambos os algoritmos foram desenvolvidos em C++ e testados em um conjunto diversificado de imagens, o que permitiu uma avaliação aprofundada de suas características, vantagens e desvantagens, cumprindo assim os objetivos propostos.

A análise revelou diferenças fundamentais entre as duas metodologias. A abordagem por AGM opera de forma não supervisionada, com o resultado sendo controlado globalmente pelo parâmetro de escala $k$. Este método demonstrou ser robusto para a segmentação de propósito geral, sendo capaz de agrupar pixels em regiões perceptualmente coesas sem a necessidade de intervenção do usuário, embora a escolha de um valor ideal para $k$ possa ser desafiadora.

Em contrapartida, o método de Caminho Mínimo se destacou como uma ferramenta poderosa para a segmentação interativa e direcionada. Sua dependência da seleção de sementes pelo usuário confere grande controle sobre o resultado, tornando-o ideal para a extração de objetos de interesse específicos. A utilização da imagem de gradiente como base para o cálculo dos custos, conforme implementado, mostrou-se particularmente eficaz para a delineação de contornos precisos, que servem como ”barreiras”para a expansão dos caminhos.

Conclui-se que não há um algoritmo universalmente superior; a escolha entre as duas abordagens é intrinsecamente dependente da aplicação. Para tarefas de segmentação automática e exploratória, onde não há conhecimento prévio dos objetos de interesse, o método de AGM é mais indicado. Para aplicações que exigem a extração de um objeto específico com alta precisão e onde a intervenção do usuário é desejável, a abordagem interativa de Caminho Mínimo é a mais adequada.

O desenvolvimento deste projeto reforçou a versatilidade e a expressividade da teoria dos grafos como uma ferramenta fundamental para a modelagem e solução de problemas complexos em visão computacional, transformando conceitos teóricos abstratos em resultados visuais concretos e significativos. Como trabalhos futuros, sugere-se a exploração de funções de custo mais complexas, como a função *f_max*, e a implementação de otimizações de desempenho para ambos os algoritmos, visando sua aplicação em ima-

gens de alta resolução em tempo real.