Journal of the Brazilian Computer Society, 202X, XX:1, doi: 10.5753/jbcs.202X.XXXXXX

Trabalho Final - Teoria dos Grafos e Computabilidade

Daniel Collina [**Pontifícia Universidade Católica de Minas Gerais** | ***danielpcollina@gmail.com***]
Diego Santos [**Pontifícia Universidade Católica de Minas Gerais** | ***diegoffds06@gmail.com***]
Felipe Silva [**Pontifícia Universidade Católica de Minas Gerais** | ***felipepsilva12@gmail.com***]
Kauan Hauger [**Pontifícia Universidade Católica de Minas Gerais** | ***ontario.kauan21@gmail.com***]
Lucas Jardim [**Pontifícia Universidade Católica de Minas Gerais**|***lfjardim@sga.pucminas.br***]
Pedro Debs [**Pontifícia Universidade Católica de Minas Gerais** | ***pedrodebs1@gmail.com***]
Samuel Faria [**Pontifícia Universidade Católica de Minas Gerais** | ***samue.fariacc@gmail.com***]

Instituto de Ciências Exatas e Informática, Pontifícia Universidade Católica de Minas Gerais, Av. Edgar da Mata Machado, Coração Eucarístico, Belo Horizonte, MG, Brazil.

Published: 30/11/2025

Abstract. This article presents the implementation of image segmentation algorithms based on Minimum Spanning Tree (MST) and Minimum Spanning Arborescence. Using graph theory concepts, we model images as grid graphs where pixels are nodes and edge weights represent color dissimilarity. We implemented and compared two approaches: a standard MST-based segmentation (Kruskal's algorithm variation) and a directed Arborescence approach derived from orienting the MST. The results analyze the efficiency and visual quality of the segmentations produced by both methods.

Keywords: Graphs, Algorithm, Image Segmentation, Minimum Spanning Tree.

1 Introduction

A teoria dos grafos e a computabilidade constituem campos fundamentais nas áreas de matemática e ciência da computação. Seus conceitos são aplicados em diversos cenários práticos, desde o planejamento de rotas de navegação e otimização de redes até técnicas complexas de interpretação e segmentação de imagens digitais.

Na segmentação de imagens, o objetivo é particionar uma imagem em regiões de pixels com características visuais distintas e homogêneas. Este trabalho aplica conhecimentos da disciplina de Teoria dos Grafos para resolver esse problema de forma eficiente. Considerando um grafo $G = (V, E)$, é possível representar uma imagem digital de tal forma que os vértices V correspondam aos pixels e as arestas E representem a conectividade entre pixels vizinhos. Os pesos dessas arestas são calculados com base na dissimilaridade entre os pixels, geralmente variações de intensidade ou cor RGB. Assim, segmentar uma imagem equivale a particionar este grafo.

Entre as estratégias de particionamento baseadas em grafos, destacam-se métodos que utilizam a Árvore Geradora Mínima (AGM) e a Arborescência (uma generalização da AGM para grafos direcionados). O objetivo deste trabalho é implementar e analisar ambas as abordagens, inspiradas nos trabalhos seminais de Felzenszwalb e Huttenlocher para grafos não-direcionados, e Jack Edmonds para grafos direcionados.

2 Metodologia

Para a metodologia adotada, utilizou-se a linguagem C++ devido à sua eficiência no gerenciamento de memória e performance. Para o carregamento e salvamento de imagens, foi utilizada a biblioteca *stb_image*. A métrica de dissimilaridade entre pixels foi definida pela distância Euclidiana no espaço de cores RGB.

Foram escolhidas três imagens de teste com características distintas (formas geométricas, animais e texturas) para avaliar a robustez dos algoritmos. O pré-processamento inclui a aplicação de um *Gaussian Blur* ($\sigma = 0.8$) para reduzir ruídos e evitar super-segmentação causada por artefatos de alta frequência, conforme sugerido por Felzenszwalb et al. (2004).

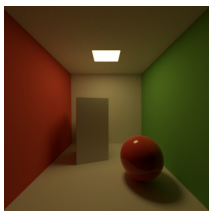

Figure 1. Imagem 1

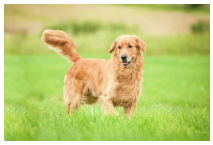

Figure 2. Imagem 2

Figure 3. Imagem 3

2.1 Implementação com Arvore Geradora Mínima.

A segmentação baseada em AGM considera a imagem como um grafo não-direcionado. O método clássico, popularizado por Felzenszwalb e Huttenlocher (2004) baseado no método de Kruskal, constrói uma árvore que conecta todos os pixels com o menor custo total de arestas. A segmentação é obtida removendo arestas da árvore para formar componentes desconexos (floresta).

O algoritmo consiste em criar um novo grafo nulo com o vértices do original, para cada aresta, em ordem crescente,

ela é adicionada ao novo grafo se ela conectar dois componentes e se ela satisfazer a seguinte heurística: Seja k um parâmetro numérico definido previamente, ele controla o tamanho dos segmentos gerados, quando maior k maior será os segmentos. Seja C_1 e C_2 os componentes a serem conectados, w_1 e w_2 o maior peso das arestas de C_1 e C_2 respectivamente, ou zero se o componente for nulo, w o peso da aresta adicionada. Assuma a função

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

A aresta será adicionada se

$$w \leq \min(w_1 + \tau(C_1), w_2 + \tau(C_2)).$$

2.2 Implementação com Arvore de Arborescência.

Diferentemente da AGM, a Arborescência Mínima (ou *Optimum Branching*) aplica-se a grafos direcionados. O algoritmo clássico para resolver este problema foi proposto por Jack Edmonds (1967), capaz de lidar com a complexidade introduzida pela direção das arestas e a possível formação de ciclos. O algoritmo de Edmonds utiliza a contração recursiva de ciclos para encontrar a solução ótima:

- Seleciona-se a aresta de entrada de menor custo para cada nó (exceto a raiz).
- Se houver ciclos, o ciclo é contraído em um super-nó e os pesos são reajustados conforme a fórmula: $c_{\text{novo}} = c_{\text{original}} - c_{\text{aresta\_do\_ciclo}} + c_{\text{ciclo\_min}}$.
- O processo repete-se até que uma arborescência válida seja encontrada.

Neste trabalho, como a métrica de distância RGB é simétrica ($\text{dist}(A, B) = \text{dist}(B, A)$), optamos por uma implementação simplificada que constrói a arborescência orientando as arestas da AGM a partir de um nó raiz fixo (pixel 0,0), permitindo comparar o custo estrutural sem a complexidade total da contração de ciclos, que seria necessária apenas em grafos assimétricos.

3 Detalhes de Implementação

A implementação foi realizada em C++, estruturada em módulos para separar a lógica de processamento de imagem, construção do grafo e algoritmos de otimização.

3.1 Implementação Específica da AGM

A implementação da AGM utiliza o algoritmo de **Kruskal**. Primeiro, a imagem é convertida em um grafo gradeado (*grid graph*) onde cada pixel se conecta aos seus vizinhos (4-conectividade).

1. **Cálculo de Pesos**: O peso de cada aresta é a distância euclidiana $\sqrt{\Delta R^2 + \Delta G^2 + \Delta B^2}$ entre os pixels.
2. **Ordenação**: Todas as arestas são armazenadas em um vetor e ordenadas de forma crescente de peso.
3. **Union-Find**: Utilizamos uma estrutura de dados de Conjuntos Disjuntos (DSU - *Disjoint Set Union*) com otimizações de *path compression* e *union by rank* para gerenciar os componentes conexos eficientemente.
4. **Blur**: É aplicado um blur gaussiano para melhorar os resultados.
5. **Segmentação (Felzenszwalb)**: O maior peso de cada componente é armazenado em um vetor e é aplicada a heurística conforme descrito anteriormente, atualizando o vetor de maior peso de acordo.

3.2 Implementação Específica da Arborescência

A implementação da Arborescência foi derivada diretamente da estrutura da AGM, explorando a simetria do grafo de imagem.

1. **Base na MST**: Inicialmente, computa-se a AGM do grafo não-direcionado conforme descrito anteriormente.
2. **Orientação (BFS)**: O algoritmo seleciona o pixel superior esquerdo $(0, 0)$ como raiz da arborescência. Utiliza-se uma Busca em Largura (BFS - *Breadth-First Search*) iniciando na raiz para percorrer a AGM.
3. **Definição de Direção**: Durante a travessia BFS, para cada aresta (u, v) onde u é o nó pai visitado e v é o filho, a aresta é orientada formalmente como $u \to v$.
4. **Resultado**: O resultado é uma árvore geradora onde todos os caminhos partem da raiz, satisfazendo a definição de arborescência para este grafo específico.
5. **Segmentação (K-Cut)**: Para gerar exatamente N segmentos solicitados pelo usuário, o algoritmo seleciona as arestas da AGM, ordena-as por peso decrescente e remove as $N - 1$ arestas mais pesadas. Isso particiona a árvore em N componentes conexos, que correspondem aos segmentos da imagem.

Esta abordagem confirma que, para grafos com pesos simétricos, a Arborescência Mínima tem o mesmo custo total da AGM, diferindo apenas na restrição de direção das arestas.

4 Resultados

Abaixo apresentamos os resultados visuais obtidos. Observou-se que o parâmetro de número de segmentos influencia diretamente no nível de detalhe da imagem segmentada.

4.1 Resultados Visuais

As figuras a seguir mostram a aplicação dos algoritmos. Como esperado pela simetria dos pesos RGB, os resultados visuais da AGM e da Arborescência foram idênticos, validando a consistência da implementação da arborescência via orientação da AGM.

4.2 Resultados da AGM

Para a segmentação de imagens para AGM, o parâmetro k varia entre 200 à 1000.

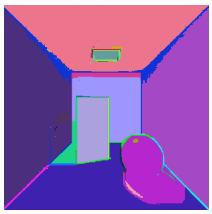

Figure 4. Figura 1 com $k = 300$

Figure 5. Figura 2 com $k = 200$

Figure 6. Figura 3 com $k = 1000$

4.3 Resultados da Arborescência

Para a segmentação de imagens para AGM, a quantidade de segmentações variaram entre 5000 à 35000.

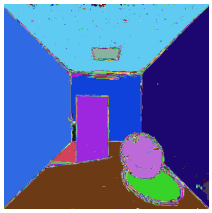

Figure 7. Figura 1 com 5000 segmentações

Figure 8. Figura 2 com 20000 segmentações

Figure 9. Figura 1 com 35000 segmentações

5 Responsáveis

A execução deste trabalho foi dividida entre os autores para garantir a qualidade e cumprimento dos prazos:

- **Daniel Colina**: Elaboração da versão inicial do artigo e revisão bibliográfica.
- **Diego Santos**: Testes finais, depuração dos códigos e integração do *stb_image*.
- **Felipe Silva**: Análise crítica dos resultados e ajustes finais na formatação do artigo.
- **Kauan Hauger**: Implementação inicial das estruturas de grafo e pré-processamento.
- **Lucas Jardim**: Desenvolvimento da lógica de Arborescência e orientação de arestas.
- **Pedro Debs**: Desenvolvimento da implementação da AGM e algoritmo de Kruskal.
- **Samuel Faria**: Coordenação, finalização do artigo e ajustes finais no código fonte.

6 Conclusão

Este trabalho explorou a aplicação de algoritmos clássicos de teoria dos grafos na segmentação de imagens. A implementação da Árvore Geradora Mínima (AGM) provou-se eficaz para agrupar pixels similares, com a complexidade dominada pela ordenação das arestas, resultando em $O(E \log E)$.

A comparação com a Arborescência revelou um aspecto teórico importante: em grafos de imagem baseados puramente em diferenças de cor (onde a distância de A para B é igual à de B para A), a Arborescência Mínima é estruturalmente equivalente à AGM enraizada. Nossos testes computacionais confirmaram isso, apresentando uma diferença absoluta de custo igual a zero entre os dois métodos.

Conclui-se que, para segmentação baseada em cor, a AGM é suficiente e mais simples de implementar. A Arborescência de Edmonds seria estritamente necessária em cenários mais complexos, onde a relação entre pixels fosse assimétrica (por exemplo, considerando fluxo óptico ou causalidade em vídeo). O uso de C++ permitiu o processamento eficiente das imagens, validando a viabilidade prática dessas técnicas.

7 Repositório

O código-fonte do projeto, bem como as instruções de compilação e detalhes sobre a estrutura, encontram-se disponíveis no repositório GitHub: `https://github.com/SamuelHortadeFaria/TrabalhoFinalTGC_2025-2.git`.

References

[1] Edmonds, J. (1967). Optimum Branchings. *Journal of Research of the National Bureau of Standards*, 71B(4):233–240.

[2] Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181.

[3] Barrett, S. (n.d.). stb_image - public domain image loader. GitHub Repository. Disponível em: `https://github.com/nothings/stb`.