Algoritmos de Segmentação de Imagem Baseados em Grafos

Fábio Antônio C. Andrade[1], **Lucas A. Barros**[1], **Pedro M. Lima**[1], **Pedro R. Maia**[1]

[1]Instituto de Ciências Exatas e Informática – Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte – MG – Brasil

{fabio.andrade.1452236, lucas.alkmim.1450684, pedro.lima.1463698,

pedro.maia.1469860}@sga.pucminas.br

Abstract. *This paper presents the implementation of two graph-based image segmentation algorithms proposed by [Felzenszwalb and Huttenlocher 2004] and [Boykov and Funka-Lea 2006]. While Felzenszwalb-Huttenlocher performs segmentation based on regions and internal differences, Boykov–Funka-Lea models segmentation as energy minimization, using max-flow/min-cut algorithms.*

Resumo. *Este artigo apresenta a implementação de dois algoritmos de segmentação de imagem baseados em grafos, propostos por [Felzenszwalb and Huttenlocher 2004] e [Boykov and Funka-Lea 2006]. Enquanto Felzenszwalb-Huttenlocher realiza a segmentação a partir de regiões e diferenças internas, Boykov–Funka-Lea modela a segmentação como minimização de energia, utilizando algoritmos de max-flow/min-cut.*

1. Introdução

O problema de segmentação de imagem é um grande desafio da visão computacional. Por ser um problema com uma alta gama de aplicações, como análise médica, reconhecimento de objetos, sistemas de navegação autônoma e processamento de vídeos, algoritmos que resolvam o problema são de extrema relevância.

Dentre os métodos existentes para a segmentação de imagem, existem aqueles baseados em grafos. Este trabalho foca em dois algoritmos desse tipo: o método de [Felzenszwalb and Huttenlocher 2004], que busca identificar limites entre regiões com base na variabilidade local; o método de [Boykov and Funka-Lea 2006] que formula o problema como uma minimização de energia.

Neste artigo, são apresentadas implementações de ambos os métodos e a aplicação deles em imagens reais e sintéticas.

2. Trabalhos Relacionados

A literatura utilizada como base para esse trabalho foram os artigos de [Felzenszwalb and Huttenlocher 2004] e [Boykov and Funka-Lea 2006].

[Felzenszwalb and Huttenlocher 2004] desenvolveram um algoritmo de segmentação de imagem baseado em um predicado para medir evidência para um limite entre duas regiões, considerando a variabilidade interna e as diferenças entre elas. Utilizando uma representação baseada em grafos para a imagem, cada pixel se torna um vértice, com as diferenças em relação as vizinhos se tornando arestas pesadas.

Apesar de ser baseado em decisões gulosas, o algoritmo produz segmentações que satisfazem propriedades globais. O artigo apresentou aplicação do algoritmo para dois tipos distintos de vizinhanças locais: grafos de grade e grafos de vizinhos próximos.

[Boykov and Funka-Lea 2006] aborda métodos para segmentação de imagens com N dimensões utilizando cortes em grafos. Este trabalho apresenta algoritmos que exploram representações gráficas de imagens, onde os pixels ou voxels são modelados como nós de um grafo, e as relações entre eles (como similaridade de intensidade ou conectividade) são representadas como arestas com pesos.

O método de cortes em grafos otimiza uma função de energia que combina termos de fidelidade aos dados (baseados na aparência dos pixels) e regularização (que incentiva bordas suaves na segmentação). O artigo destaca como essa abordagem pode ser aplicada eficientemente em imagens 2D e 3D, sendo útil em áreas como visão computacional e análise médica.

3. Algoritmos e Implementação

Os algoritmos propostos por [Felzenszwalb and Huttenlocher 2004] e [Boykov and Funka-Lea 2006], mencionados na Seção 2, foram implementados neste trabalho. Nesta seção, essas implementações serão apresentadas e explicadas.

3.1. Felzenszwalb-Huttenlocher

O algoritmo de [Felzenszwalb and Huttenlocher 2004] realiza uma segmentação de imagem baseada na limitação de regiões a partir da diferença interna de componentes. A seguir, são explicados os principais componentes da metodologia.

3.1.1. Representação do Problema com Grafos

Uma imagem I é representada como um grafo $G = (V, E)$ não direcionado, onde:

- V é o conjunto de vértices, representando os pixels da imagem.
- E é o conjunto de arestas, de forma que existe uma aresta $(u, v) \in E$ entre todos os pixels vizinhos.
- $w((u, v))$ é o peso da aresta $(u, v) \in E$, calculado como a distância euclidiana entre os dois vértices, levando em consideração tanto a posição quanto os valores *RGB* das cores.

A implementação do grafo se deu por uma lista de adjacência em que cada vértice tem uma lista de pares (vértice, peso), representando uma aresta do vértice atual para um vértice vizinho. Essa estrutura foi escolhida visando uma economia de memória, considerando uma tentativa anterior de implementar o grafo como uma matriz de adjacência que apresentou custo inviável.

Além disso, foi-se implementada uma lista de arestas ordenada de forma não-decrescente, com uma aresta sendo um objeto que contém dois vértices e um peso, tendo esse peso como fator de ordenação.

3.1.2. *Gaussian Blur*

[Felzenszwalb and Huttenlocher 2004] recomendam a aplicação de um *Gaussian blur* na imagem antes da computação dos pesos, para compensar pelos artefatos de digitalização.

Foi-se utilizada uma biblioteca da *OpenCV* para aplicar o *Gaussian blur* com $\sigma = 0.8$ e $kernel = 7$.

3.1.3. Algoritmo de Segmentação

A segmentação do grafo se dá da seguinte forma:

1. Cada vértice é iniciado como seu próprio componente.
2. A partir da lista de arestas, já ordenada, para cada aresta, compara-se se seus dois vértices pertecem ao mesmo componente.
3. Caso sejam de componentes distintos, verifica-se se o peso da aresta é menor que a diferença interna mínima dos componentes e, se a resposta for verdadeira, une-se os dois componentes.

Dessa forma, garante-se apenas a geração de árvores geradoras mínimas, pois sempre será selecionada a menor aresta para unir componentes conexos disjuntos, graças à lista ordenada de arestas.

3.1.4. Componentes e *Union Find*

O gerenciamento de componentes foi feito por meio da estrutura *Union Find*. Essa estrutura contém uma lista de pais dos componentes, uma lista de alturas dos componentes, uma lista das diferenças internas dos componentes e uma lista dos tamanhos dos componentes. Com essa estrutura, é possível gerenciar os componentes a partir de um vértice pai, de forma que na união, o pai do componente de maior altura se torna o pai do novo componente.

Ao final, são atribuidas cores aos componentes, gerando uma imagem-resultado da segmentação.

3.2. Boykov–Funka-Lea

O algoritmo de [Boykov and Funka-Lea 2006] modela o problema de segmentação como uma minimização de uma função de energia que captura as propriedades desejadas da segmentação. A seguir, são explicados os principais componentes da metodologia.

3.2.1. Representação do Problema com Grafos

O primeiro passo na abordagem de Boykov-Funka-Lea é mapear o problema em um grafo não direcionado $G = (V, E)$. Neste contexto, uma imagem I é representada da seguinte maneira:

- V denota o conjunto de vértices, onde cada vértice corresponde a um pixel ou voxel da imagem.

- E representa o conjunto de arestas, estabelecendo as conexões entre os vértices e refletindo a vizinhança local na imagem.

Cada aresta $(p, q) \in E$ é associada a um peso $w(p, q)$, que simboliza a similaridade ou afinidade entre os pixels p e q.

Durante a execução do algoritmo, é empregado um grafo residual derivado do grafo original. Este grafo residual introduz dois terminais adicionais, s (source) e t (sink). As arestas no grafo residual são classificadas como n-links, que interligam pixels vizinhos e são bidirecionais, e t-links, que conectam cada pixel aos nós terminais e são unidirecionais.

A implementação contém uma lista de adjacência para representar o grafo e um mapa para armazenar as capacidades das arestas.

3.2.2. Ponderamento das Arestas

No algoritmo de Boykov-Funka-Lea, as arestas do grafo são ponderadas para refletir a similaridade entre os pixels conectados e a probabilidade de pertencerem ao objeto ou ao fundo. Existem dois tipos de arestas: n-links e t-links.

N-links: Conectam pixels vizinhos na imagem. O peso de uma aresta n-link é calculado com base na diferença de intensidade e na distância espacial entre os pixels conectados. Quanto mais semelhantes os pixels em termos de intensidade e mais próximos espacialmente, maior o peso da aresta. Consequentemente, arestas com pesos maiores são menos propensas a serem cortadas durante a segmentação, aumentando a chance dos pixels conectados pertencerem à mesma região. Na implementação, a função `Bpq` calcula esse peso. A função considera a diferença de intensidade ponderada por um fator exponencial, dando mais importância a diferenças maiores, e normaliza o resultado pela distância euclidiana entre os pixels.

T-links: Conectam cada pixel aos terminais *source* (objeto) e *sink* (fundo). O peso de uma aresta t-link reflete a probabilidade de um pixel pertencer ao objeto ou ao fundo. Um peso alto na conexão com o terminal do objeto sugere alta probabilidade de o pixel pertencer ao objeto, e analogamente para o fundo. Esses pesos são calculados a partir de histogramas das intensidades de pixels das *seeds* (pixels manualmente selecionados como objeto ou fundo). O histograma fornece uma estimativa da probabilidade de um pixel com determinada intensidade pertencer ao objeto ou fundo, que é então usada para calcular os pesos das arestas t-link.

3.2.3. Algoritmo Boykov-Kolmogorov

O algoritmo de Boykov-Kolmogorov calcula o corte mínimo em um grafo para minimizar a função de energia na segmentação de imagens. Ele opera iterativamente, buscando caminhos no grafo residual que permitam o aumento do fluxo da fonte (*source*) para o destino

(*sink*). Esses caminhos, chamados de caminhos de aumento, são aqueles compostos exclusivamente por arestas com capacidade residual positiva, indicando que ainda podem acomodar mais fluxo.

A cada iteração, o algoritmo encontra um caminho de aumento e aumenta o fluxo ao longo dele pelo máximo possível. Esse valor máximo é determinado pela aresta com a menor capacidade residual presente no caminho de aumento, limitando o fluxo adicional que pode ser enviado. Este aumento de fluxo acarreta alterações no grafo residual: a capacidade residual das arestas utilizadas no caminho é reduzida pelo valor do fluxo aumentado, enquanto a capacidade residual das arestas reversas correspondentes (que permitem "desfazer" fluxos enviados previamente) é aumentada na mesma proporção.

A saturação de uma aresta ocorre quando sua capacidade residual se torna nula, impedindo a passagem de mais fluxo por ela. O processo iterativo de busca por caminhos de aumento e atualização das capacidades residuais continua até que nenhum caminho de aumento possa ser encontrado. Essa condição indica que o fluxo máximo foi alcançado.

Com o fluxo máximo determinado, o corte mínimo, que define a segmentação da imagem, é obtido. O corte mínimo divide os vértices do grafo em dois conjuntos: aqueles alcançáveis a partir da fonte no grafo residual (considerando apenas arestas com capacidade residual não-nula) e aqueles inalcançáveis. Este corte representa a separação entre objeto e fundo na imagem segmentada.

3.2.4. Algoritmo de Segmentação

O processo de implementação foi organizado em seis etapas principais:

1. **Leitura da Imagem:**
    - A imagem de entrada foi carregada e representada em formato matricial, onde cada pixel corresponde a um nó no grafo.
2. **Montagem do Grafo Residual:**
    - Foram definidos os nós do grafo correspondentes aos pixels da imagem.
    - Arestas foram criadas conectando pixels vizinhos (*n-links*) e ligando os pixels às regiões de objeto ou fundo (*t-links*).
3. **Determinação das *Seeds* (Objeto e Fundo):**
    - As *seeds* foram escolhidas manualmente, especificando exemplos de regiões de objeto e fundo.
4. **Construção do Histograma:**
    - Distribuições de intensidade foram geradas com base nas *seeds*.
    - Um histograma foi utilizado para modelar as regiões de objeto e fundo.
5. **Determinação dos Pesos das Arestas:**
    - Os pesos das arestas foram definidos com base na similaridade de intensidade entre pixels vizinhos.
    - As distribuições de intensidade calculadas também foram utilizadas para definir os pesos.
6. **Execução do Algoritmo de Fluxo Máximo:**
    - O algoritmo de Boykov-Kolmogorov foi utilizado para determinar o fluxo máximo no grafo residual.

- O corte mínimo foi identificado, segmentando o grafo em duas regiões correspondentes a objeto e fundo.

A segmentação final foi representada como uma máscara binária aplicada à imagem original, destacando as regiões de interesse.

4. Resultados

4.1. Felzenszwalb-Huttenlocher

A partir do algoritmo apresentado na Seção 3.1, foi realizada a segmentação de duas imagens. Na Figure 1 e na Figure 2, apresentamos as imagens originais e suas respectivas segmentações. Foi-se utilizado um valor de $k = 5000$ para o cálculo do τ.

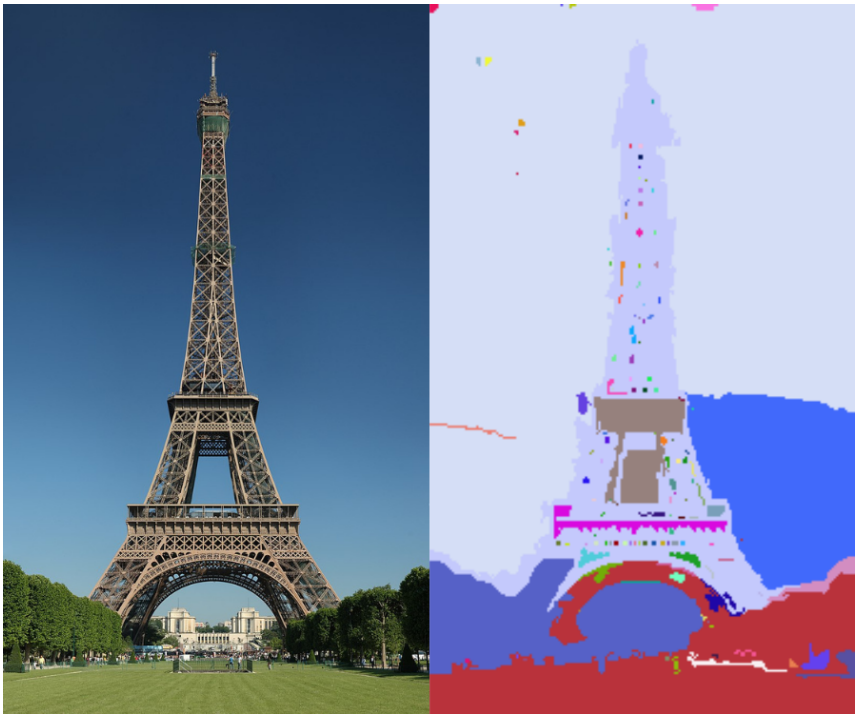

Figure 1. Torre Eiffel

Figure 2. Jogo de baseball

4.2. Boykov–Funka-Lea

A partir do algoritmo apresentado na Seção 3.2, foi realizada a segmentação de uma imagem. Na Figure 3, apresentamos a imagem original e sua segmentação.

Figure 3. Árvore

5. Conclusão

Este trabalho implementou e aplicou dois algoritmos distintos para segmentação de imagens baseados em grafos: Felzenszwalb-Huttenlocher e Boykov–Funka-Lea. Cada algoritmo aborda o problema com uma perspectiva única.

Felzenszwalb-Huttenlocher, baseado na identificação de fronteiras entre regiões, demonstrou ser eficiente em imagens com contornos bem definidos e regiões homogêneas (Figure 1 e Figure 2). A escolha do parâmetro k impacta significativamente o resultado. O algoritmo tende a gerar segmentações com regiões mais uniformes em termos de cor.

Boykov–Funka-Lea, que modela a segmentação como minimização de energia por cortes em grafos, demonstrou ser eficaz em segmentar objetos com contornos mais complexos e em imagens com ruído (Figure 3). A utilização de *seeds* permite um controle mais interativo. Em Boykov–Funka-Lea, o parâmetro λ controla o balanço entre os termos regionais (t-links) e os termos de fronteira (n-links) na função de energia: valores altos de λ resultam em segmentações mais suaves, enquanto valores baixos resultam em segmentações mais detalhadas. O parâmetro σ, presente na função que calcula o peso dos n-links, controla a sensibilidade às diferenças de intensidade: um σ pequeno resulta em segmentações mais detalhadas, e um σ grande resulta em segmentações mais suaves. A escolha adequada desses parâmetros, juntamente com a seleção das *seeds*, é crucial para obter bons resultados.

Felzenszwalb-Huttenlocher destaca-se pela simplicidade e eficiência computacional, adequado para aplicações que demandam velocidade e em imagens com características favoráveis. Boykov–Funka-Lea oferece maior precisão e robustez em cenários

mais desafiadores, porém com um custo computacional mais elevado e dependência da seleção adequada de *seeds* e dos parâmetros λ e σ.

Como trabalhos futuros, sugere-se a investigação de métodos para aprimorar a escolha automática do parâmetro k no algoritmo de Felzenszwalb-Huttenlocher e a exploração de diferentes estratégias para o cálculo dos pesos das arestas, incluindo a otimização dos parâmetros λ e σ, e a seleção automática de *seeds* no algoritmo de Boykov–Funka-Lea. A aplicação dos algoritmos em diferentes tipos de imagens e a comparação com outros métodos de segmentação também são relevantes.

Contribuições Individuais

As contribuições de cada integrante foram:

- **Fábio Antônio C. Andrade:** Implementação do algoritmo Boykov–Funka-Lea. Escrita do artigo.
- **Lucas A. Barros:** Implementação do algoritmo Boykov–Funka-Lea. Escrita do artigo.
- **Pedro M. Lima:** Implementação das estruturas de dados (Grafos, Lista de Arestas e *Union Find*) e dos algoritmos Felzenszwalb-Huttenlocher e Boykov–Funka-Lea.
- **Pedro R. Maia:** Implementação das estruturas de dados (Grafos e Lista de Arestas) e do algoritmo Felzenszwalb-Huttenlocher. Escrita do artigo.

References

Boykov, Y. and Funka-Lea, G. (2006). Graph cuts and efficient n-d image segmentation. *International Journal of Computer Vision*, 70(2):109–131.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181.