Implementação e Análise Comparativa de Metodologias para Segmentação de Grafos: Um Estudo de Caso sobre Segmentação Eficiente

Sophia C. V. de Sousa[1], Luiz Fernando A. S. Frassi[1], Laura Costa[1], Lívia Xavier[1], Luca Gonzaga[1]

[1]Departamento de Ciência da Computação (DCC) – Pontifícia Universidade Católica de Minas Gerais (PUCMG)
Caixa Postal 2071 – 30.535-901 – Belo Horizonte – MG – Brasil

Abstract. *Image segmentation is a constantly evolving field with applications in various areas. In this context, this work contributes to the domain by comparing two widely cited segmentation techniques in the literature. Through the implementation of the methodologies by Felzenszwalb and Huttenlocher and by Boykov and Funka-Lea in C++, the results obtained demonstrate the advantages and disadvantages of each approach, providing insights for selecting techniques for different applications.*

Resumo. *A segmentação de imagens é um campo em constante evolução, com aplicações em diversas áreas. Dessa forma, este trabalho contribui para o domínio ao comparar duas técnicas de segmentação globalmente citadas na literatura. Por meio da implementação das metodologias de Felzenszwalb e Huttenlocher e de Boykov e Funka-Lea em C++, os resultados obtidos demonstram as vantagens e desvantagens de cada abordagem, fornecendo insights para uma seleção de técnicas para diferentes aplicações.*

1. Introdução

Com o rápido desenvolvimento tecnológico, a visão computacional e o processamento de imagens emergem como áreas de grande relevância no cenário brasileiro, e a segmentação de imagens não é uma exceção [Cheng and Li 2021]. A segmentação de imagens, uma técnica consistida em dividir uma imagem em regiões distintas com significado semântico, desempenha um papel crucial nesse contexto. A crescente demanda por soluções automatizadas em diversas áreas, como rastreamento de objetos, controle de qualidade industrial, diagnóstico médico [Chen and Pan 2018] e processamento de imagens satélite, impulsiona a busca por algoritmos de segmentação cada vez mais robustos e eficientes. Neste projeto, propomos uma replicação e uma análise comparativa de duas metodologias de segmentação de imagens utilizadas na literatura: [Felzenszwalb and Huttenlocher 2004] e [Boykov and Funka-Lea 2006], posteriormente abordadas na Seção 3. O código-fonte utilizado neste estudo está disponível publicamente em [gra 2024].

2. Background

A segmentação de imagens é uma técnica de visão computacional que divide uma imagem digital, geralmente em formato PGM (Portable Gray Map) - um tipo de arquivo que

armazena imagens em escala de cinza, em duas dimensões - em grupos de pixels que compartilham uma característica entre si, para detectar recursos visuais denominados por luz, limites de objetos, regiões de plano de fundo ou cor [IBM]. Algumas das técnicas mais comuns de segmentação de imagens são:

- Limiarização: Cria imagens binárias com base na intensidade acima ou abaixo de um determinado ”valor limite”predefinido.
- Detecção de bordas: Identifica as fronteiras entre objetos ou regiões, baseando-se em descontinuidades de intensidade.
- Crescimento de regiões: Agrupa pixels com características semelhantes a partir de sementes iniciais, expandindo as regiões de forma iterativa dentro de frameworks e garantindo segmentações refinadas em múltiplas dimensões, como em [Boykov and Funka-Lea 2006].

3. Trabalhos Relacionados

No contexto em que a segmentação de imagens foi implementada com diversas abordagens propostas em projetos acadêmicos, dois trabalhos se destacam para este estudo: o algoritmo de segmentação eficiente de Felzenszwalb e Huttenlocher e a técnica de Graph Cuts desenvolvida por Boykov e Funka-Lea. O estudo [Felzenszwalb and Huttenlocher 2004] propôs um algoritmo baseado em grafos que mede a evidência de um limite entre regiões através de um predicado adaptativo. A abordagem se destaca por sua eficiência computacional, operando em tempo quase linear em relação ao número de pixels da imagem: $O(mlogm)$ para a maioria dos passos do algoritmo. Esse foi testado em uma variedade de imagens reais e sintéticas, demonstrando uma capacidade considerável de preservar detalhes importantes em regiões de baixa variabilidade, como demonstrado na Figura 1. Assim, esses resultados demonstraram que a metodologia segmenta eficientemente essas imagens, sendo particularmente eficaz em aplicações que exigem segmentação rápida sem comprometer a precisão.

Figura 1. Resultados de segmentação produzidos pelo algoritmo de [Felzenszwalb and Huttenlocher 2004] a partir de uma imagem 320 × 240, cor

A técnica de [Boykov and Funka-Lea 2006] aplica algoritmos de combinação para otimizar funções de energia, formulando o problema de segmentação como um corte mínimo em grafos s/t (a diferença entre a soma do fluxo das arestas de cruzamento st e o fluxo das arestas de cruzamento ts). Assim, em contraste com os métodos baseados em “caminhos”, essa é uma abordagem baseada em “regiões”. Os autores evidenciaram que o método pode alcançar soluções ótimas globais, integrando múltiplas pistas visuais, as “seeds” manuais, para melhorar a robustez da segmentação. Os testes construídos em dados N-D confirmaram a eficácia do método em cenários mais complexos, destacando sua aplicabilidade em contextos que exigem alta precisão.

4. Metodologia

As contribuições deste trabalho baseiam-se na implementação e análise das metodologias propostas em [Felzenszwalb and Huttenlocher 2004] e [Boykov and Funka-Lea 2006]. Ambos foram implementados na linguagem C++ com a adição de alguns métodos em Python, o que permitiu um desenvolvimento mais eficiente.

Em ambos os algoritmos, foi utilizada uma estrutura baseada em grafos, definida como $G = (V, E)$, onde G é um grafo não-direcionado e conexo. Os vértices $v_i \in V$ representam o conjunto de elementos a serem segmentados, os pixels da imagem, enquanto as arestas $(v_i, v_j) \in E$ correspondem a pares de vértices vizinhos. Cada aresta possui um peso $w(v_i, v_j)$, que é uma medida de dissimilaridade entre os dois pixels conectados pela aresta.

4.1. Metodologia 1

Como discutido na seção 3, a abordagem proposta por Pedro Felzenszwalb e Daniel Huttenlocher busca dividir a imagem em regiões homogêneas com base na semelhança de cores entre regiões adjacentes e utiliza o grafo como estrutura de dados para a modelagem do problema. Nesta implementação, foi utilizada a biblioteca OpenCV, com o intuito de facilitar tarefas como leitura e manipulação de imagens em C++. O fluxo desse algoritmo é ilustrado na Figura 2.

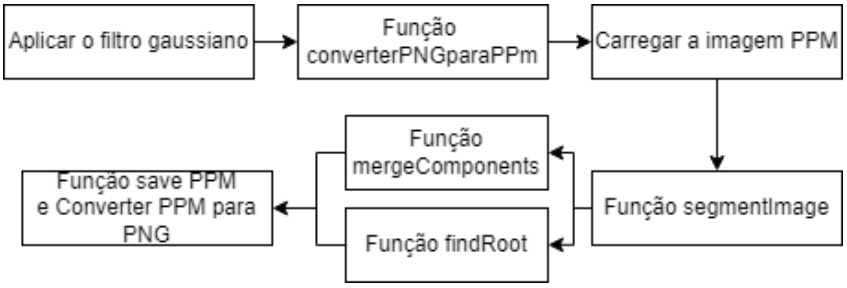

Figura 2. Fluxo de execução do Algoritmo 1

4.1.1. Pré-Processamento da Imagem e Contrução do Grafo

Inicialmente, o algoritmo realiza a aplicação de um filtro gaussiano com σ = 0.8 para suavizar os detalhes e reduzir o ruído da imagem. Após esse processamento, a função `ConverterPNGparaPPM` utiliza a biblioteca sys do Python para realizar a conversão de PNG para PPM. A função `loadPPM` então carrega a imagem PPM gerada anteriormente, realiza a leitura de suas dimensões e inicializa uma estrutura de dados tridimensional para armazenar os valores RGB de cada pixel.

O algoritmo então constrói uma lista de arestas que conecta pixels adjacentes (horizontalmente, verticalmente, e nas diagonais) com pesos determinados pela diferença euclidiana entre os valores de cor (RGB) dos pixels conectados. Foi implementada uma lógica na qual, caso a diferença seja baixa, os pixels são tratados como iguais. Essa técnica se mostra útil para que o algoritmo fique menos sensível à variações de tonalidade em um mesmo objeto. As arestas então são ordenadas em ordem não decrescente de peso.

4.1.2. Segmentação da Imagem

Cada pixel é inicialmente tratado como um componente separado, os quais possuem propriedades como tamanho, peso máximo interno e um identificador único de cor. Para segmentar a imagem em regiões bem definidas, o algoritmo deve decidir se os componentes devem ser fundidos, o que é calculado por critérios baseados na ”diferença interna”de cada componente, ajustada por um parâmetro k e o tamanho desse componente. Se o peso da aresta for menor ou igual à menor diferença interna entre os componentes, `mergeComponents` é chamada para realizar a fusão.

Após a fusão dos componentes iniciais, cada pixel é associado ao rótulo do componente ao qual ele pertence, encontrado pela função `findRoot`. Em seguida, a função `mergeComponents` é responsável por unir componentes disjuntos semelhantes em um único componente maior. Para isso, ela verifica os tamanhos de dois componentes envolvidos e funde o menor ao maior, ajustando suas raízes de forma apropriada. Após isso, a função atualiza o tamanho do novo componente resultante e redefine o peso máximo, selecionando o maior valor entre os pesos máximos dos componentes anteriores e o peso da aresta que os conecta. Por fim, atribui-se uma nova cor única ao componente resultante, garantindo sua identificação na imagem segmentada.

Por fim, o algoritmo gera um mapping que associa cada rótulo de componente segmentado a uma cor RGB única, percorrendo os rótulos de cada pixel e gerando valores aleatórios para os componentes que ainda não possuem cor. Ele então define os valores das cores associadas ao mapa em cada pixel e retorna a imagem em formato PNG.

4.2. Metodologia 2

No algoritmo implementado para replicação da metodologia de [Boykov and Funka-Lea 2006], é proposto um método de segmentação de imagens utilizando o algoritmo de corte mínimo em grafos. Assim como a abordagem anterior, esta modela a imagem como um grafo, onde cada pixel é representado como um nó, e as conexões entre os nós adjacentes são definidas por arestas ponderadas. Porém, o grafo também inclui dois nós terminais especiais, chamados source (fonte) e sink (sorvedouro), representando o objeto e o fundo, respectivamente. Os pesos das arestas refletem a probabilidade de um pixel pertencer a uma dessas categorias. O fluxo do algoritmo é apresentado na Figura 3. Além disso, foi utilizada a biblioteca maxflow-v3.01 [uwaterloo 2024], desenvolvida por Vladimir Kolmogorov, para auxiliar na computação de cortes mínimos em fluxos máximos de um grafo arbitrário.

4.2.1. Pré-Processamento da Imagem e Contrução do Grafo

O processo de segmentação da imagem se inicia com a seleção manual de seeds (sementes) para o objeto e o fundo. A interação do usuário é facilitada pela interface gráfica desenvolvida com a biblioteca OpenCV, onde os pontos de sementes são marcados diretamente na imagem carregada no formato `PPM`. Assim, utilizando o botão esquerdo do mouse, marca-se os pontos pertencentes ao objeto, com a cor em verde e a anotação “O” e ao clicar na imagem com o botão direito do mouse, marca-se os pontos pertencentes ao fundo, exibidos em vermelho com a anotação “F”.

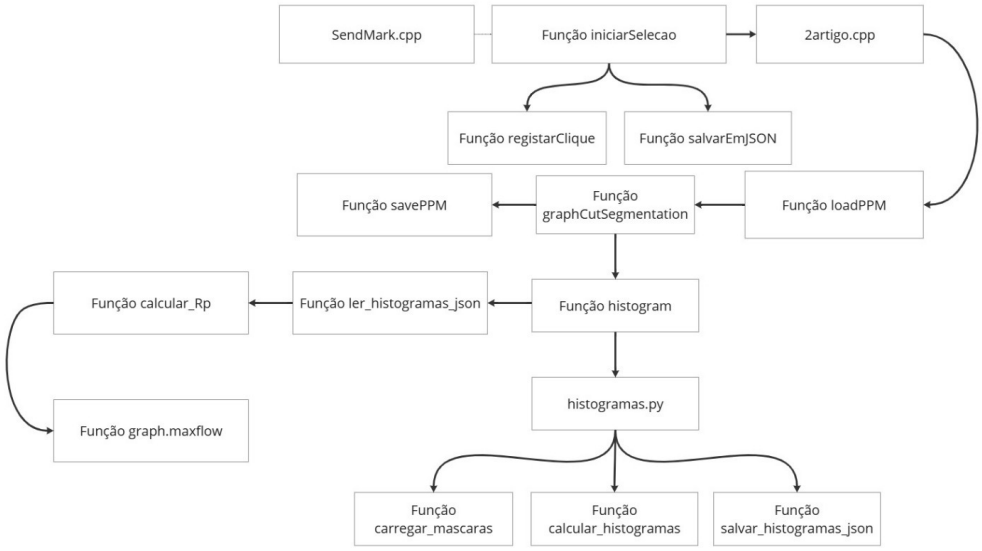

Figura 3. Fluxo de execução do Algoritmo 2

Em seguida, a imagem é lida como um vetor linear de pixels usando a função `loadPPM` e o grafo é construído com auxílio da biblioteca maxflow-v3.01, com cada pixel da imagem como um nó no grafo e os pesos das arestas como a diferença de intensidade entre os pixels conectados, calculada com base na métrica euclidiana. Conexões adicionais são estabelecidas entre cada pixel e os nós terminais source e sink, com pesos baseados na probabilidade de pertencimento (1 e 2). Essa, por sua vez, é calculada com base nos histogramas de intensidade para os valores de intensidade dos pixels referentes aos pontos selecionados como sementes do objeto e do fundo, obtidos por um código python que os retorna em formato Json.

$$R_p(\text{“obj”}) = -\ln \Pr(I_p \mid \text{“obj”}) \tag{1}$$

$$R_p(\text{“bkg”}) = -\ln \Pr(I_p \mid \text{“bkg”}) \tag{2}$$

4.2.2. Cálculo do Corte Mínimo

Uma vez construído o grafo, o algoritmo de corte mínimo é aplicado ao grafo para determinar a segmentação ideal da imagem, utilizando a resolução do fluxo máximo, que identifica o menor conjunto de arestas que, ao serem removidas, separam o grafo em dois subconjuntos: um conectado ao source e o outro ao sink, seguindo os princípios descritos em [Boykov and Funka-Lea 2006] e garantindo uma solução ótima global. Assim, o subconjunto conectado ao nó source representa os pixels classificados como objeto e o subconjunto conectado ao sink são classificados como fundo. Durante o cálculo, as arestas de menor peso (que representam maior similaridade entre pixels) são favorecidas no corte, garantindo uma segmentação precisa.

4.2.3. Pós-Processamento e Geração da Imagem Segmentada

Após a execução do algoritmo, cada pixel é atribuído a uma das duas regiões (objeto ou fundo) com base no conjunto ao qual pertence no grafo segmentado. Em seguida, cada região é associada a uma cor distinta para facilitar a visualização dos resultados, descritas na seção 5. A imagem segmentada é gerada no formato PPM e posteriormente convertida para PNG.

5. Resultados

A primeira metodologia foi testada em uma série de figuras, como a fotografia de uma igreja e a fotografia de uma van em frente a um gramado. Os testes se encontram em pares de original-resultado, ilustrados na Figura 4. As imagens segmentadas revelam uma separação clara entre as regiões dos objetos e o fundo. A igreja, por exemplo, foi destacada com uma cor predominante, enquanto o fundo e outras áreas da imagem foram atribuídos a diferentes cores, demonstrando as regiões segmentadas. Esses resultados indicam que o algoritmo é promissor para aplicações que exigem a identificação precisa de objetos em imagens, como em sistemas de análise de imagens médicas. No entanto, ajustes futuros podem se mostrar necessários para otimização da segmentação conforme o uso específico desejado.

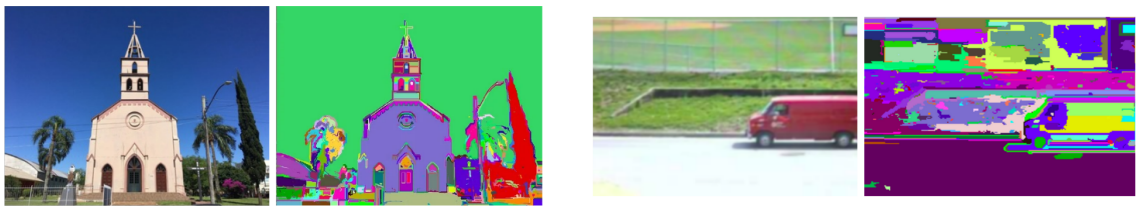

Figura 4. Imagens "Original-Resultado"das segmentações produzidas pelo algoritmo 1 com $K = 300$ e $\sigma = 0.8$, utilizando a fotografia de uma igreja e de uma van.

A segunda metodologia também foi testada em uma série de imagens, como a mesma fotografia da igreja e da van utilizadas anteriormente. Os testes também se encontram em pares de original-resultado, ilustrados na Figura 5. Os resultados dessa segmentação geram uma máscara binária onde o background é destacado em branco e o objeto de interesse em preto, o que demonstrou uma separação eficaz do objeto principal, determinado pelas sementes incluídas manualmente, do fundo, evidenciando sua silhueta em ambos os casos de teste. Elementos menores, como detalhes da vegetação ao redor, foram excluídos da segmentação, indicando um foco maior na estrutura principal. Assim, esse algoritmo se mostra útil para funções como análise ambiental e análise de imagens biomédicas.

6. Conclusões e Trabalhos Futuros

Este trabalho realizou a implementação de duas técnicas de segmentação de imagens citadas na literatura, testadas em um conjunto de fotografias, permitindo uma análise comparativa de seus desempenhos. Os resultados alcançados demonstram que ambas as metodologias são capazes de segmentar imagens de forma eficaz, mas com características

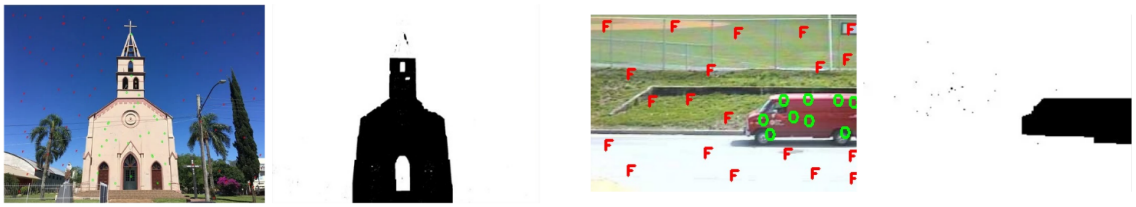

Figura 5. Imagens "Original-Resultado"das segmentações produzidas pelo algoritmo 2, utilizando a fotografia de uma igreja e de uma van.

e aplicações diferentes. O algoritmo de Felzenszwalb se mostrou eficiente em segmentar imagens com diversas regiões e detalhes, sendo vantajosa em aplicações que exigem segmentação rápida sem comprometer a precisão. Por outro ângulo, a técnica de Graph Cuts demonstrou ser boa em segmentar objetos selecionados em primeiro plano, com uma robustez significativa, sendo mais adequada para aplicações que exigem a identificação precisa de objetos em imagens.

Como trabalhos futuros, será vantajosa a exploração de demais técnicas de segmentação, que podem fornecer outras vantagens e desvantagens para a segmentação de imagens, e também integrar as técnicas de segmentação juntamente a outras técnicas de processamento de imagens, como o reconhecimento de padrões.

6.1. Divisão do Projeto

Este estudo foi dividido entre os autores da seguinte forma: A metodologia 1 foi desenvolvida principalmente por Laura Costa e Sophia Carrazza e a metodologia 2 foi desenvolvida principalmente pelo Luiz Fernando, Lívia Xavier e Luca Gonzaga. Porém, ambos os algoritmos foram implementados por meio de discussões em grupo, revisados e compreendidos por todos os integrantes. Por fim, a documentação foi escrita por Sophia Carrazza e Laura Costa.

Referências

(2024). sophiacarrazza/graph_segmentation: Segmentação de imagens baseada em grafos. disponível em: https://github.com/sophiacarrazza/graph_segmentation.

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

Chen, X. and Pan, L. (2018). A Survey of Graph Cuts/Graph Search Based Medical Image Segmentation. *IEEE Reviews in Biomedical Engineering*, 11:112–124. Conference Name: IEEE Reviews in Biomedical Engineering.

Cheng, Y. and Li, B. (2021). Image segmentation technology and its application in digital image processing. In *2021 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, pages 1174–1177.

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

IBM. O que é segmentação de imagem? | IBM.

uwaterloo (2024). Computer Vision at Waterloo - Code.