Segmentação de Imagens por Textura usando Filtros Direcionais

Luan Marko Kujavski GRR20221236
João Pedro Vicente Ramalho GRR20224169
Heloisa Benedet Mendes GRR20221248
Departamento de Informática
Universidade Federal do Paraná – UFPR
Curitiba, Brasil
lmk22@inf.ufpr.br, jpvr22@inf.ufpr.br, hbm22@inf.ufpr.br

***Resumo*—Este trabalho propõe um método para segmentação de imagens baseado em características de textura utilizando filtros direcionais aplicados em três escalas. O sistema aplica kernels nas orientações horizontal, vertical, diagonais (45° e 135°) e circular, processando as imagens em três estágios de escala. As janelas de 32×32 pixels são descritas pela média dos valores filtrados e agrupadas via distância euclidiana com limiar fixo. O método foi validado em 16 imagens de um dataset próprio.**

I. Introdução

A segmentação por textura é fundamental para identificar regiões homogêneas em visão computacional. Neste trabalho, utilizamos filtros espaciais direcionais clássicos para extrair características de textura, sem empregar aprendizado profundo. A proposta engloba três etapas principais: (1) aplicação de bancos de filtros direcionais em múltiplas escalas, (2) extração de vetores de características por janelas, e (3) agrupamento das regiões com base em similaridade.

Utilizamos um conjunto de 16 imagens com diferentes texturas e padrões. A implementação, em Python 3.10 com *Open Source Computer Vision Library* (OpenCV), é reprodutível e está disponível no repositório GitHub.

II. Metodologia

A. Conjunto de Filtros

Foram implementados cinco filtros direcionais: horizontal, vertical, diagonal 45°, diagonal 135° e circular. Os filtros foram definidos em um arquivo YAML e parametrizado para três escalas de kernel: 3×3, 5×5 e 7×7. Exemplo de definição:

```
filtros:
  horizontal:
    kernels: [3,5,7]
    filtros:
      kernel_3: [...]  % escala 3x3
      kernel_5: [...]  % escala 5x5
      kernel_7: [...]  % escala 7x7
  vertical:
    kernels: [3,5,7]
    filtros:
      kernel_3: [...]
      kernel_5: [...]
      ...
```

B. Processamento Multi-escala

Adotamos abordagem sequencial:

1) Redimensionamento inicial para 512×512 pixels.
2) Para cada escala: aplicação de filtro Gaussiano com `cv2.GaussianBlur` e redução da imagem pela metade.
3) Convolução com kernels direcionais no tamanho original ou reduzido.

C. Extração de Características

O tamanho da janela (W) e da imagem redimensionada (R) são definidos pelos parâmetros –window_size e –resize_size (tipicamente 32), gravados em args.json. Para cada filtro definido no arquivo YAML (cada um contendo vários kernels), aplicamos filter2D sobre a imagem redimensionada, dividimos o resultado em janelas de W×W e calculamos a média dos valores filtrados em cada janela, gerando um vetor de características de dimensão igual ao número de janelas:

$$f_i = \frac{1}{N} \sum_{(x,y)\in W} (I * K_i)(x, y) \tag{1}$$

onde K_i é o kernel do filtro e N é o número de pixels.

Implementamos a extração de características em um script chamado `extract_textures.py`, que lê as imagens, aplica os filtros definidos no arquivo YAML e gera dois arquivos de saída: `train_features.json` e `test_features.json`.

D. Agrupamento e Segmentação

No `segmentation.py`, carregamos os arquivos de features gerados anteriormente e agrupamos as regiões com base na distância euclidiana. Para cada par de vetores de características $\mathbf{f}$ e $\mathbf{g}$, calculamos:

$$d = \|\mathbf{f} - \mathbf{g}\|_2 = \sqrt{\sum_{i=1}^{n} (f_i - g_i)^2} \tag{2}$$

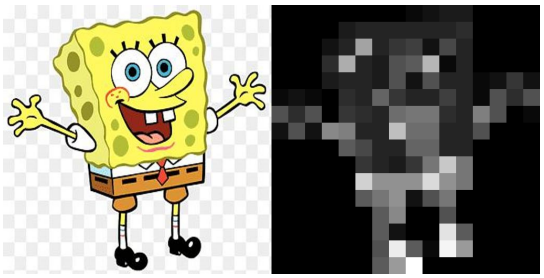
Regiões cujas distâncias satisfazem $d < T$ (onde T é o limiar definido na linha de comando) são atribuídas à mesma classe. Caso alguma região não seja agrupada com qualquer outra, ela recebe uma nova classe única (singleton).

A saída do processo de segmentação é uma imagem em tons de cinza, na qual cada região recebe um valor de intensidade proporcional à sua classe. Esse mapa de classes é normalizado para o intervalo [0, 255] e então sobreposto à imagem original, lado a lado, facilitando a comparação visual.

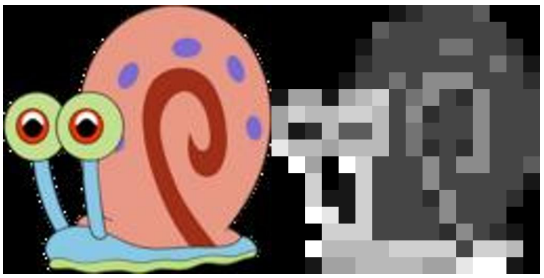
III. Resultados e Discussão

Aplicamos o método em 16 imagens, observando:

- Boa separação de texturas homogêneas.
- Dependência sensível do limiar T (testado em 0.1, 0.2 e 0.3).



(a) Bob Esponja – W:32, limiar:0.2



(b) Garry – W:32, limiar:0.2

IV. Conclusão

O método clássico de segmentação por textura mostrou-se efetivo para diferentes padrões, sem necessidade de técnicas baseadas em aprendizado profundo.

Repositório

O código está disponível em:
https://github.com/joaop-vr/texture_segmentation