Relatório - Trabalho 03

MO443 - Introdução ao Processamento de Imagem Digital

Vinicius Teixeira de Melo - RA: 230223

Universidade Estadual de Campinas
viniciusteixeira@liv.ic.unicamp.br

21 de Maio de 2019

I. Especificação do Problema

O objetivo deste trabalho é aplicar operadores morfológicos para segmentar regiões compreendendo texto e não texto em uma imagem de entrada.

Os seguintes passos devem ser realizados:

1. dilatação da imagem original com um elemento estruturante de 1 pixel de altura e 100 pixels de largura;
2. erosão da imagem resultante com o mesmo elemento estruturante do passo (1);
3. dilatação da imagem original com um elemento estruturante de 200 pixels de altura e 1 pixel de largura;
4. erosão da imagem resultante com o mesmo elemento estruturante do passo (3);
5. aplicação da intersecção (AND) dos resultados dos passos (2) e (4);
6. fechamento do resultado obtido no passo (5) com um elemento estruturante de 1 pixel de altura e 30 pixels de largura;
7. aplicação de algoritmo para identificação de componentes conexos (ver programa fornecido) sobre o resultado do passo (6);
8. para cada retângulo envolvendo um objeto, calcule:
   (a) razão entre o número de pixels pretos e o número total de pixels (altura x largura);
   (b) razão entre o número de transições verticais e horizontais branco para preto e o número total de pixels pretos;
9. criação de uma regra para classificar cada componente conexo, de acordo com as medidas obtidas no passo (8), como texto e não texto.
10. aplicação de operadores morfológicos apropriados para segmentar cada linha do texto em blocos de palavras. Coloque um retângulo envolvendo cada palavra na imagem original. Calcule o número total de linhas de texto e de blocos de palavras na imagem.

II. Entrada de Dados

O código fonte criado para a execução de todas as tarefas está no notebook **Trabalho 03.ipynb**. O código foi criado para aceitar imagens em preto e branco no formato PBM (*Portable Bitmap*).

Para executar o notebook, basta iniciar o ambiente *Jupyter Notebook*, abrir o notebook **Trabalho 03.ipynb** e executar as células em ordem. Todo o algoritmo foi implementado na linguagem Python na versão 3.6.

As imagen de entrada utilizada no teste do algoritmo foi retirada da página do prof. Hélio Pedrini: Imagens. Na pasta **imgs/** está a imagem em preto e branco utilizada no teste: **bitmap.pbm**. As dimensões da imagen de entrada utilizada é 1374 x 2233.

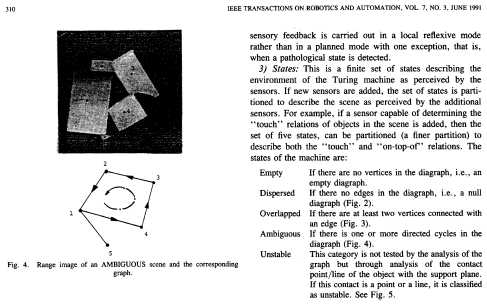

Figura 1: *bitmap.pbm*

III. Dependências e Códigos

As bibliotecas utilizadas neste trabalho foram:

Biblioteca	**Versão**
numpy	1.16.2
cv2	3.4.2
matplotlib	3.0.3
warnings	2.1

A leitura das imagens foi realizada utilizando uma função do **opencv** [1] chamada **imread()**, a qual necessitou de uma constante do próprio **opencv** para que a imagem ficassem apenas com o canal de escala de cinza, essa constante é denominada **IMREAD_GRAYSCALE**.

Os elementos estruturantes foram criados manualmente através da função **np.ones((*shape*),np.uint8)**. Primeiramente, a imagem de entrada foi transformada para que as áreas de interesse estivessem com valor 255. Para as operações de *bits*, foram utilizadas as funções **cv2.bitwise_not** e **cv2.bitwise_and**.

Nas operações fundamentais de morfologia foram utilizadas com base nas funções **cv2.dilate** e **cv2.erose**. Para a operação de fechamento, foi utilizada a função **cv2.morphologyEx** com a constante do próprio OpenCv, chamada **cv2.MORPH_CLOSE**.

No passo (7), para a detecção de componentes e criação dos *bounding boxes* foi utilizada a função **cv2.findContours**, juntamente com as constantes **cv2.RETR_TREE** e **cv2.CHAIN_APPROX_SIMPLE**. Assim, após a obtenção dos *bounding boxes*, foram utilizadas as funções **cv2.boundingRect**, **cv2.rectangle** e **cv2.drawContours** para desenhar os *bounding boxes* na imagem de entrada.

No passo (9), a regra utilizada para classificar um *bounding box* como texto ou não foi: dado o valor x_a obtido no passo (8.a) e o valor x_b obtido no passo (8.b) referentes ao mesmo *bounding box*, se $|x_a - x_b| \leq 0.2$, esse *bounding box* é classificado como texto, se não, é classificado como não texto.

IV. Fundamentação

i. Morfologia Matemática

A morfologia matemática se refere ao estudo e análise de imagens usando operadores não lineares. Dentre as atividades que a morfologia executa no processamento de imagens estão: aumento de qualidade, restauração, detecção de falhas, análise da textura, análise particular, análise de formas, compreensão, dentre outras. Essas ferramentas são aplicadas em diversas áreas como visão em robos, inspeções, microscopia, biologia, metalurgia e documentos digitais. Ambas áreas e técnicas continuam a se expandir.

O processo da morfologia se baseia na geometria, em um contexto bem específico. A idéia básica é percorrer uma imagem com um elemento estruturante e quantificar a maneira com que este se encaixa ou não na imagem. No caso afirmativo, marca-se o local ou nível de cinza onde o elemento estruturante coube na imagem. Dessa forma, pode-se extrair informações relevantes sobre o tamanho e forma de estruturas na imagem.

i.1 Dilatação

A operação de dilatação entre o conjunto A e o elemento estruturante B é definida como a *adição de Minkowski*, ou seja

$$\mathcal{D}(A, B) = A \oplus B = \bigcup_{b \in B} (A + b). \qquad (1)$$

O processo de dilatação entre A e B corresponde ao conjunto de todas as translações de

B com os pontos da imagem em que há pelo menos um elemento não nulo (pixel com valor 1) em comum com o conjunto A.

i.2 Erosão

A operação de erosão entre o conjunto A e o elemento estruturante B é definida como a subtração de *Minkowski*, ou seja

$$\mathcal{E}(A, B) = A \ominus B = \bigcap_{b \in B} (A - b). \qquad (2)$$

A erosão de A por B é o conjunto de todos os elementos de B transladados por p que estão contidas em A. Entretanto, deve-se observar que o resultado da erosão de uma imagem pode não ser um subconjunto de imagem original, caso o elemento estruturante não contenha a origem.

i.3 Fechamento

O fechamento de A por B, denotado por $A \bullet B$, é definido como

$$A \bullet B = (A \oplus B) \ominus B \qquad (3)$$

o qual pode ser interpretado geometricamente como a união de todas as translações de B que não estão contidas em A.

V. Saída de Dados

As imagens resultantes foram salvas dentro da pasta **resultados/** utilizando uma função da biblioteca **opencv** chamada **cv2.imwrite()** [1].

O formato dos nomes de saída estão da seguinte forma: para cada passo do problema, é salva uma imagem com o nome "bitmap_(passo)", e no passo número 10 é uma imagem referente a todos os componentes conexos e outra somente com os componentes que foram classificados como palavras.

VI. Resultados e Discuções

Os resultados obtidos com a aplicação do passo a passo definido no problema estão a seguir:

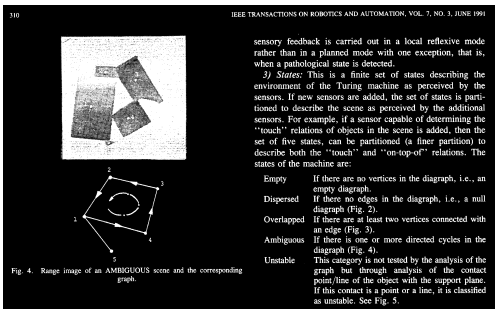

Figura 2: *bitmap_not.pbm*

Primeiramente, para a aplicação das operação básicas de morfologia matemática, a imagem de entrada foi transformada em "negativa". Os *pixels* que tinham o valor 255 passaram a ter valor 0 e os *pixels* que tinham valor 0 passaram a ter o valor 255. O resultado dessa operação é mostrado na Figura 2.

Figura 3: *Passo (1)*

A Figura 3 apresenta o resultado da dilatação da imagem original com um elemento estruturante de 1 *pixel* de altura e 100 *pixels* de largura. Como resultado temos uma imagem em que os componentes foram alargados e alguns foram unidos, formando um componente conexo mais abrangente. Podemos observar os componentes que estavam separados em uma mesma linha agora estão formando somente um componente. E como o elemento estruturante é maior na horizontal, podemos observar que os componentes tiveram sua largura aumentada.

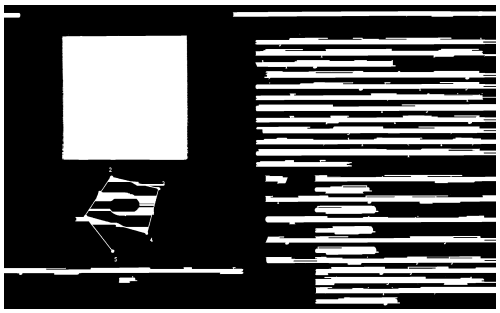

Figura 4: *Passo (2)*

A Figura 4 apresenta o resultado da erosão da imagem da Figura 3 com o mesmo elemento estruturante do passo (1). Podemos observar que com essa operação de erosão a imagem sofreu um redução na parte externa das componentes conexas, assim é possível fazer uma analogia com um ”afinamento” dessas componentes. Com esse passo, podemos ver também que o espaçamento entre as linhas ficou mais bem definido.

Figura 5: *Passo (3)*

A Figura 5 apresenta o resultado da dilatação da imagem original com o elemento estruturante de 200 *pixels* de altura e 1 *pixel* de largura. Podemos observar que a imagem sofreu um alargamento das componentes na vertical, resultando assim na união de muitos componentes.

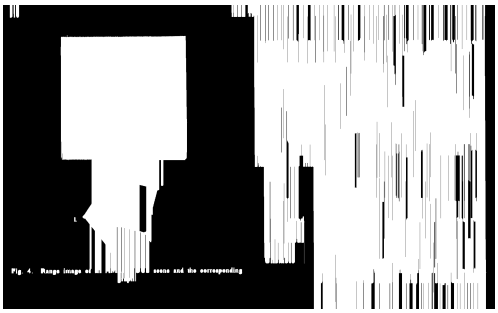

Figura 6: *Passo (4)*

A Figura 6 apresenta o resultado da erosão da imagem da Figura 5 com o mesmo elemento do passo (3). Podemos observar que ocorreu a remoção de uma camada externa dos componentes conexos, e os componentes que não ocupavam uma área considerável foram ”excluídos”.

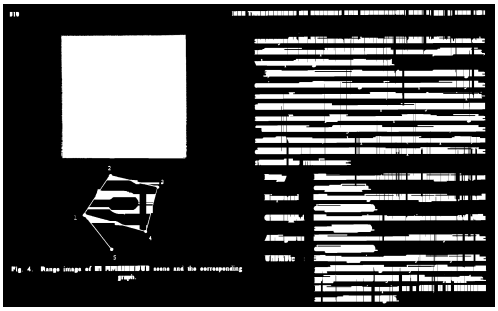

Figura 7: *Passo (5)*

A Figura 7 apresenta o resultado da operação de *AND* entre a imagem obtida após o passo (2) e o passo (4). Com essa operação, a imagem resultante mantém os *pixels* que tinham o valor 255 nas duas imagens, assim, a imagem resultante mantém a divisão por linhas e ainda separações entre componentes de uma mesma linha, obtidas através da imagem do passo (4).

Figura 8: *Passo (6)*

A Figura 8 apresenta o resultado da operação de fechamento da imagem da Figura 7 com um elemento estruturante de 1 *pixel* de altura e 30 *pixels* de largura. Como a operação de fechamento faz uma dilatação seguida de uma erosão, essa operação tem como objetivo "fechar" os buracos que possam existir nos componentes. Podemos observar que os componentes de uma mesma linha que estavam separados foram unidos, assim, no passo (7) é possível fazer uma boa identificação dos componentes conexos.

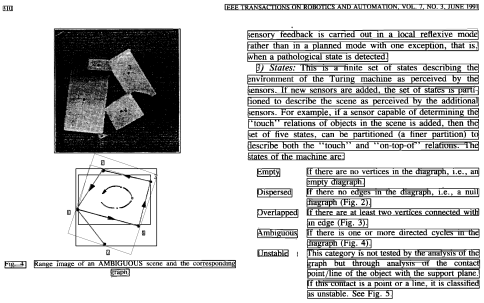

Figura 9: *Passo (7)*

A Figura 9 apresenta o resultado da operação de detecção dos componentes conexos. Podemos observar que cada componente conexo existente na imagem da Figura 8 foi envolvido com um *bounding box*. Para a identificação dos componentes conexos foi considerado a vizinhaça-4.

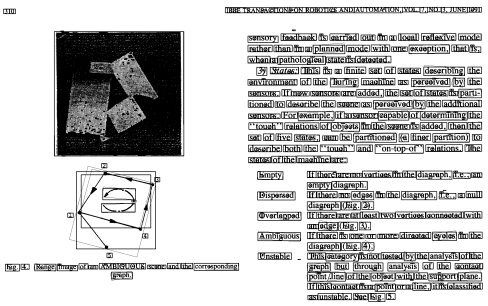

Figura 10: *Passo (10.1)*

A Figura 10 apresenta o resultado da identificação dos componentes conexos e seus *bounding boxex* correspondentes após a execução de 3 iterações da dilatação da imagem da Figura 2 com um elemento estruturante de 1 *pixel* de altura e 3 *pixels* de largura.

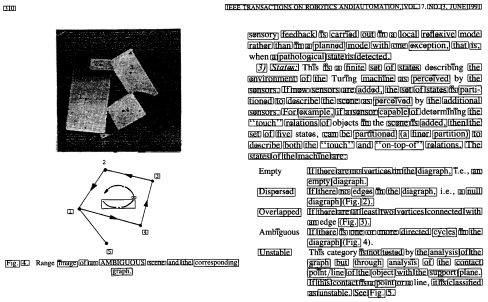

Figura 11: *Passo (10.2)*

A Figura 11 apresenta o resultado da operação realizada na figura anterior, porém, só foram desenhados os *bounding boxex* dos componentes que foram classificados como texto, de acordo com os resultados obtidos no passo (8.a) e (8.b) e a regra definida anteriormente. Assim, essa figura representa os componentes que foram reconhecidos como palavras.

A contagem do número de linhas e palavras foi realizada da seguinte forma: (i) a contagem de linhas foi realizada com base na imagem da Figura 9 e foram contabilizados somente os componentes que foram classificados como texto; (ii) a contagem de palavras foi realizada com base na imagem da Figura 11 e foram contabilizados todos *bounding boxex* que foram

classificados como texto.

Como resultado final temos que o número de linhas é 36 e o número de palavras é 305.

VII. Conclusão

Podemos concluir que os resultados obtidos com as aplicações das técnicas de morfologia matemática para a detecção de linhas e palavras foram satisfatórios quanto a especificação do trabalho e que a criação de uma regra para classificação de texto e não texto com base nos cálculos do passo (8.a) e (8.b) se mostrou razoável para este objetivo, assim, consolida ainda mais os conceitos vistos em sala de aula.

Referências

[1] Welcome to opencv documentation! https://docs.opencv.org/2.4/index.html Acesso em: 04/05/2019.

[2] Pedrini, Hélio, and William Robson Schwartz. Análise de imagens digitais: princípios, algoritmos e aplicações. Thomson Learning, 2008.

[3] Matplotlib Version 3.0.3 https://matplotlib.org/contents.html Acesso em: 04/05/2019.