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

19 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:

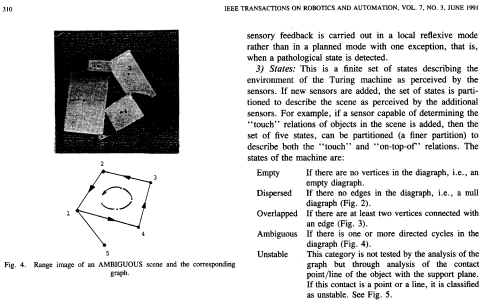
1. dilatação da imagem original com um elemento estruturante de 1 pixel de altura e 100 pixels de largura;
2. erosão da imagem resultante com o mesmo elemento estruturante do passo (1);
3. dilatação da imagem original com um elemento estruturante de 200 pixels de altura e 1 pixel de largura;
4. erosão 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.



310 IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, VOL. 7, NO. 3, JUNE 1991

Fig. 4. Range image of an AMBIGUOUS scene and the corresponding graph.

sensory feedback is carried out in a local reflexive mode rather than in a planned mode with one exception, that is, when a pathological state is detected.

3) States: This is a finite set of states describing the environment of the Turing machine as perceived by the sensors. If new sensors are added, the set of states is partitioned to describe the scene as perceived by the additional sensors. For example, if a sensor capable of determining the "touch" relations of objects in the scene is added, then the set of five states, can be partitioned (a finer partition) to describe both the "touch" and "on-top-of" relations. The states of the machine are:

Empty	If there are no vertices in the diagraph, i.e., an empty diagraph.
Dispersed	If there no edges in the diagraph, i.e., a null diagraph (Fig. 2).
Overlapped	If there are at least two vertices connected with an edge (Fig. 3).
Ambiguous	If there is one or more directed cycles in the diagraph (Fig. 4).
Unstable	This category is not tested by the analysis of the graph but through analysis of the contact point/line of the object with the support plane. If this contact is a point or a line, it is classified as unstable. See Fig. 5.

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

V. Saída de Dados

VI. Resultados e Discuções

VII. Conclusão

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.