Trabalho 4 de Introdução ao Processamento de Imagens

Nome: Miguel Augusto S. Guida - RA: 174847

Junho de 2020

1 Introdução

O objetivo deste trabalho é aplicar operadores morfológicos para segmentar regiões compreendendo texto e não texto em uma imagem de entrada. Aplicamos dilatação, erosão e fechamento com diferentes elementos estruturantes e envolvemos as componentes conexas resultantes em retângulos, classificando as regiões de texto e não texto de acordo com a razão entre o número de pixels pretos e o número total de pixels e de acordo com a razão entre o número de transições verticais e horizontais branco para preto e o número total de pixel pretos.
Segmentamos as regiões de texto primeiramente por linhas e depois por blocos de palavras.

2 O Programa

O programa foi desenvolvido no Jupyter Notebook, utilizando Python 3.7.6. As bibliotecas utilizadas foram Numpy 1.18.1, OpenCV 4.2.0, Matplotlib 3.1.3.
Para executar o programa, basta rodar arquivo "trabalho4_ra174847_mc920.ipynb" no programa Jupyter Notebook.

3 Entrada e Saída de Dados

As imagens de entrada e saída estão no formato PBM. Todas imagens estarão presentes no diretório do programa implementado.

4 Segmentação por linhas de texto

Para segmentar a imagem em linhas de texto, seguimos os passos a seguir:

1. Dilatação da imagem original com um elemento estruturante de 1 pixel de altura e 100 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 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 de largura.

7. Aplicação de algoritmo para identificação de componentes conexas sobre o resultado do passo (6)

8. Para cada retângulo envolvendo um objeto, calcule:

   - Razão entre o número de pixels pretos e o número total de pixels
   - Razão entre o número de transições verticais e horizontais branco para preto e o número total de pixel pretos

9. Criação de uma regra para classificar cada componente conexa, de acordo com as medidas obtidas no passo (8)

4.1 Implementação

Para que conseguíssemos aplicar os operadores morfológicos, invertemos os valores da imagem, deixando o background branco com valor 0 e os pixels pretos com valor 1, assim os elementos estruturantes, populados com valor 1, podem atuar sobre os pixels de texto.
Utilizamos a biblioteca Numpy e OpenCV para as operações.
Criamos o vetor de elemento estruturante com a função np.ones() do numpy, com as dimensões desejadas. Para os operadores morfológicos, utilizamos as funções erode(), dilate() e closing() da biblioteca OpenCV.
Para a identificação de componentes conexas, utilizamos a função findContours() do OpenCV, que destaca as regiões conexas e as retorna numa espécie de lista.Voltamos a imagem para seu valor original, e com a função boundingRect() e rectangle() também do OpenCV, desenhamos na imagem original os retângulos ao redor das componentes conexas resultantes das operações morfológicas e destacadas pelo findContours().

4.2 Resultados

A imagem original e a imagem com seus valores invertidos estão apresentadas na figura 1. O resultado dos passos 1 e 2 podem ser observados na figura 2, e o resultado dos passos 3 e 4 podem ser observados na figura 3. Podemos observar nos dois casos que a dilatação expande as regiões brancas, enquanto a erosão as diminui. Como usamos um elemento estruturante horizontral nos passos 1 e 2, as dilatações e erosões atuam nesse eixo, enquanto que nos passos 3 e 4 elas atuam na vertical.
O resultado dos passos 5 e 6 estão apresentados na figura 4.Note que na imagem do passo 5, resultante da intersecção dos passos 2 e 4, existem linhas verticais bem finas, decorrentes da operação AND das duas imagens. Essas linhas finas desaparecem depois da operação de fechamento do passo 6. No passo 6, temos a aplicação da última operação morfológica, e seu resultado será utilizado para encontrar as componentes conexas que formam as linhas de texto da imagem. Podemos observar que as regiões que compreendem texto na imagem formam uma componente conexa para cada linha.
No passo 7, com a ajuda da biblioteca OpenCV, encontramos 52 componentes conexas, e para desenhar um retângulo ao redor das componentes que compreendem texto, determinamos classificações no passo 8.
Observamos que as componentes que compreendiam texto apresentavam o valor da razão entre o número de pixels pretos e o número total de pixels entre os valores 0.1 e 0.8 e a segunda razão com valor maior que 0.60. Assim, das 52 componentes conexas identificadas, 38 foram classificadas como regiões de texto e tiveram seus contornos destacados por retângulos na imagem original. COmo temos 38 linhas de texto na imagem (contando com os números da imagem), nosso algoritmo foi capaz segmentar e classificar todas linhas de texto.
Podemos observar a imagem com as linhas de texto destacadas por retângulos na figura 5. É interessante observar que os números contidos na na segunda figura da imagem também foram considerados como texto, menos o número 3, que integrou à componente conexa da figura, ao invés de ter uma própria componente, devido à sua proximidade.

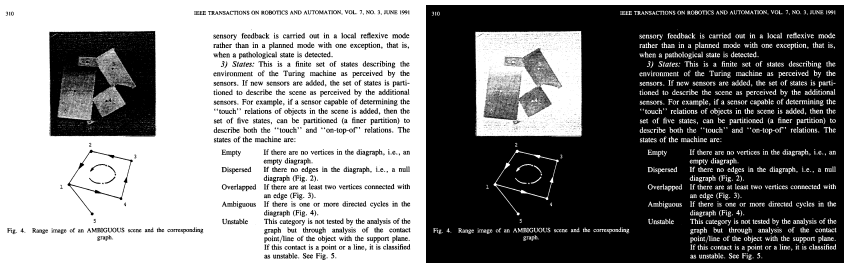

Figure 1: Imagem original e imagem com valores invertidos

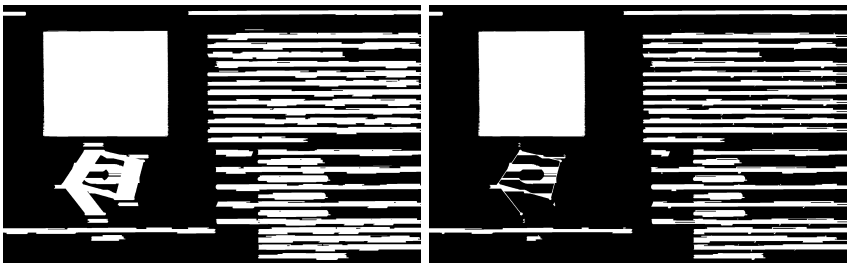

Figure 2: Imagems resultante dos passos 1 e 2

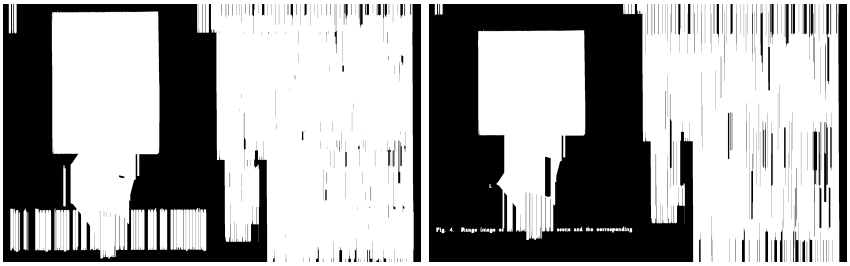

Figure 3: Imagems resultante dos passos 3 e 4

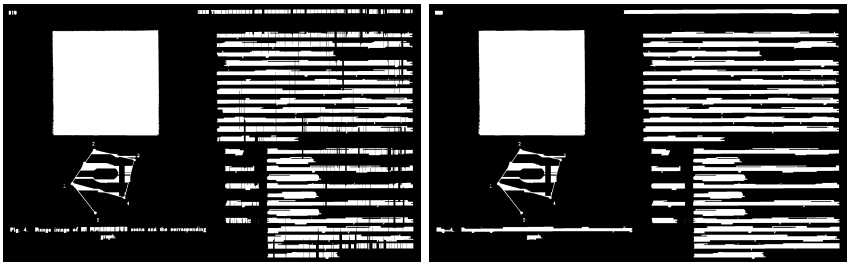

Figure 4: Imagems resultante dos passos 5 e 6

5 Segmentação por blocos de palavras

Para obter as regiões de texto segmentadas por palavras, realizamos os mesmos passos anteriores, apenas mudando as dimensões dos elementos estruturantes. Utilizamos um elemento estruturante com 1 pixel de altura e 12 pixels de largura para os passos 1 e 2, outro de 20 pixels de altura e 1 de largura para eos passos 3 e 4 e para o fechamento o passo 6, utilizamos um elemento estruturante de 1 pixel de altura e 12 pixels de largura.

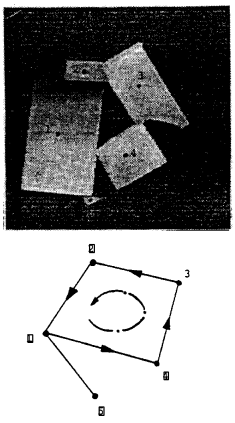

Fig. 4. Range image of an AMBIGUOUS scene and the corresponding graph.

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.

Figure 5: Imagems original com as linhas destacadas

5.1 Resultados

O resultado dos passos 1 e 2 podem ser observados na figura 6, e o resultado dos passos 3 e 4 podem ser observados na figura 7. Podemos observar nos dois casos que a dilatação expande as regiões brancas, enquanto a erosão as diminui. Como usamos um elemento estruturante horizontal nos passos 1 e 2, as dilatações e erosões atuam nesse eixo, enquanto que nos passos 3 e 4 elas atuam na vertical. O resultado dos passos 5 e 6 estão apresentados na figura 8. Ao classificar as componentes conexas com base nas razões apresentadas no passo 8, encontramos o mesmo intervalo de valores para a primeira razão (entre 0.1 e 0.8), porém não fomos capazes de encontrar um valor para a segunda razão que classificasse corretamente uma componente conexa como texto ou não texto. Dessa forma, utilizamos apenas a primeira razão como forma de classificação para esse caso. Identificamos 312 componentes conexas, e classificamos 289 componentes como regiões de texto.
Podemos observar a imagem com as palavras destacadas por retângulos na figura 9.

Foi feito um experimento com 2 tamanhos diferentes de elementros estruturantes para o fechamento no passo 6, um utilizando 1 pixel de altura e 6 de largura e outro utilizando 1 pixel de altura e 12 de largura. Percebemos uma diferença grande na quantidade de componentes conexas identificadas, sendo que no primeiro foram identificadas 371 componentes e 345 regiões de texto e na segunda 312 componentes e 289 regiões de texto. Podemos observar na figura 10 como o fechamento não une as aspas no primeiro, mas une no segundo. Ao

unir as acentuações e outros sinais gráficos às componentes conexas das palavras próximas, por meio de um elemento estruturante com dimensões maiores para o fechamento, melhoramos a contagem de palavras, considerando que cada componente conexa classificada como texto corresponde a uma palavra na imagem. A quantidade de regiões de texto encontradas (289 regiões) foi a melhor aproximação encontrada, sem que uníssemos duas componentes conexas de qualquer palavra e ao mesmo tempo minimizando as componentes conexas classificadas como texto que não correspondessem a palavras, como foi mostrado na comparação anterior.

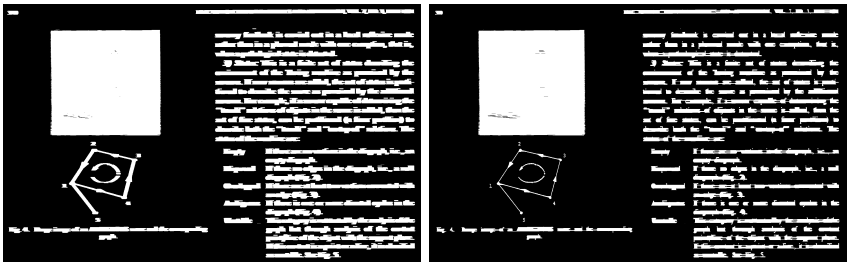

Figure 6: Imagems resultante dos passos 1 e 2

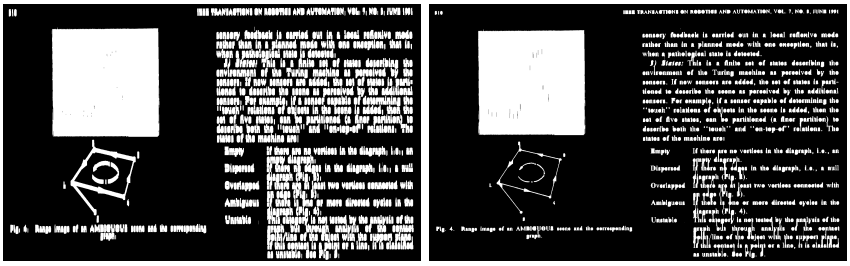

Figure 7: Imagems resultante dos passos 3 e 4

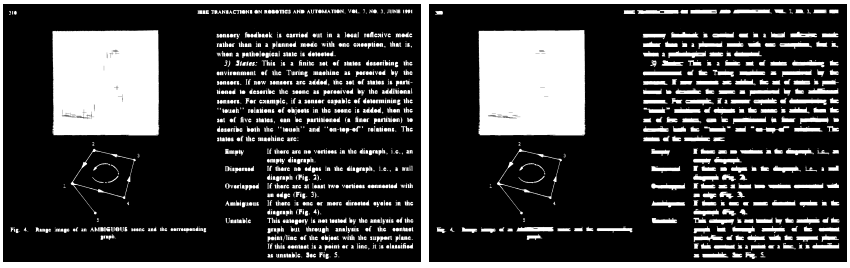

Figure 8: Imagems resultante dos passos 5 e 6

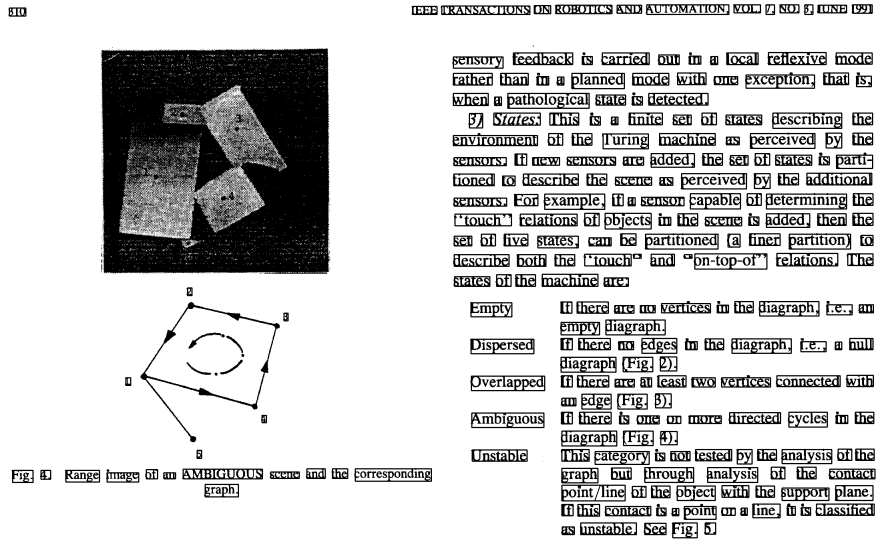

Figure 9: Imagems original com as palavras destacadas.

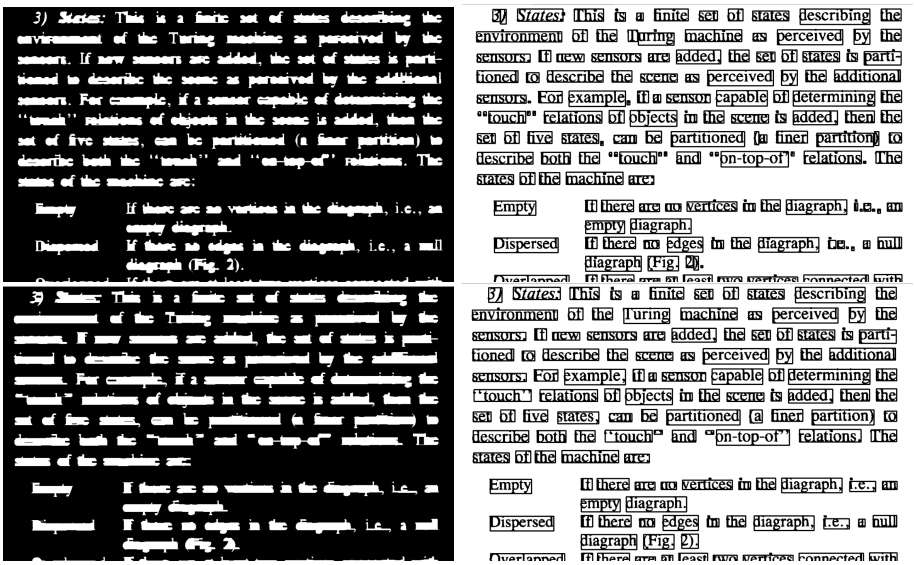

Figure 10: Comparações entre imagens resultantes do fechamento com elementos estruturantes diferentes (1x6 pixels e 1x12 pixels) e seus resultados finais com as palavras destacadas.