Trabalho 04 de Introdução ao Processamento de Imagem Digital

MARIANNA DE PINHO SEVERO

RA: 264960

I. INTRODUÇÃO

O processamento de imagens é uma importante atividade desempenhada dentro da área de Visão Computacional, sendo responsável pela captura, representação e transformação de imagens. Ele permite a extração de informações e a transformação de certas características de imagens, de maneira a facilitar tanto a percepção humana como a interpretação realizada por computadores [1].

Com respeito à transformação de imagens, diversas técnicas podem ser utilizadas, de acordo tanto com as necessidades do domínio de aplicação em que as imagens serão empregadas, como com os objetivos das transformações.

Neste trabalho, empregamos técnicas de processamento de imagens para o propósito de Análise de Documentos Digitais. Mais especificamente, empregamos operadores morfológicos para a identificação de regiões de texto e de não texto em uma imagem de um documento, e contamos a quantidade de linhas de texto e a quantidade de blocos de palavras presentes nele.

A Análise de Documentos Digitais é uma área de pesquisa que visa desenvolver metodologias para o processamento e a análise automática de imagens de documentos, contribuindo para diversas aplicações na economia, na ciência e nas áreas de estudos sociais. Dentre essas aplicações, estão o processamento de documentos históricos, a recuperação de imagens de documentos e a identificação de regiões de texto e de imagens em documentos [2].

Na Seção 2 são apresentadas as bibliotecas utilizadas neste trabalho; na Seção 3 são descritas a saída e entrada de dados; na Seção 4 apresenta-se as técnicas empregadas para a identificação das regiões que contêm texto e para a contagem do número de linhas e de palavras; na Seção 5 são apresentados os resultados e discussões; por fim, na Seção 6, apresenta-se a conclusão.

II. DEPENDÊNCIAS

Para a implementação dos algoritmos utilizados neste trabalho, as seguintes bibliotecas foram empregadas:

- numpy: usada para a manipulação dos *arrays* que representam as imagens e para a criação de estruturas de dados auxiliares.
- opencv: utilizada para a leitura e escrita de imagens e para a aplicação dos operadores morfológicos.
- matplotlib: empregada para a apresentação das imagens.

Além dessas bibliotecas, foram empregadadas funções para a apresentação das imagens (**imshowComponents** e **imshowComponentsRectangles**), baseadas no código disponibilizado em Connected Component Labeling in Python.

III. ENTRADA E SAÍDA DE DADOS

A imagem utilizada para os testes neste trabalho foi retirada do Site da Disciplina e está no formato PBM (*Portable Bit Map*). Ela foi armazenada em um diretório chamado **input_images**.

Para utilizá-la, ela foi carregada utilizando o trecho de código 1, em que o primeiro parâmetro indica o caminho da imagem e o segundo informa o formato para o qual queremos carregá-la, neste caso, 0 indica que carregaremos uma imagem no formato monocromático. A função empregada retorna um *array* numpy em que cada elemento representa a intensidade de um pixel da imagem.

```
image = cv.imread('input_images/name_image.pbm',0)
```

Código 1: Carregar uma imagem com OpenCV.

Por sua vez, as imagens geradas - que também são representadas por *arrays* numpy no programa - são armazenadas, também no formato PBM - exceto uma delas, que é armazenada no formato PNG (*Portable Network Graphics*) - em um diretório chamado **output_images**. Para salvar as imagens, utilizamos o trecho de código 2, em que o primeiro argumento é o caminho onde salvaremos a imagem e o segundo é a estrutura de dados que a representa.

```
cv.imwrite('output_images/name_image.pbm', image)
```

Código 2: Salvar uma imagem com OpenCV.

Os algoritmos implementados neste trabalho foram escritos utilizando-se a plataforma Jupyter Notebook. Dessa forma, o arquivo onde os algoritmos foram escritos possui a extensão .ipynb.

IV. QUESTÕES E SOLUÇÕES

Neste trabalho, foi pedido que aplicássemos operadores morfológicos sobre a imagem de um documento, de maneira a identificar que regiões da imagem contêm texto. Além disso, também foi pedido que contássemos a quantidade de linhas de texto, o número de blocos de palavras e que colocássemos retângulos envolventes em torno desses blocos. Nas próximas subseções, descreveremos os passos realizados para atingir esses objetivos.

A. *Tratamento da imagem de entrada*

A imagem de entrada deste trabalho possui formato PBM, o que significa que seus *pixels* possuem apenas dois valores, 0 ou 1, em que 0 representa o fundo (branco) e 1 representa o objeto (preto).

Entretanto, ao lermos a imagem utilizando o método **cv.imread**, do *opencv*, conforme mostrado no trecho de código 1, a imagem é carregada no formato monocromático, em que os *pixels* do fundo possuem valor 255 (branco) e os *pixels* dos objetos possuem valor 0 (preto).

Dessa maneira, para obtermos uma representação da imagem como esperada para o formato PBM – *pixels* do objeto com valor 1 e os do fundo com valor 0 –, aplicamos duas transformações à imagem lida: primeiro, invertemos todos os valores de seus *pixels*, de maneira que os do fundo recebessem valor 0 e os dos objetos ficassem com valor 255; depois, dividimos os valores de todos os *pixels* por 255, de modo que os únicos valores presentes na imagem fossem 0 e 1.

No trecho de códido 3, podemos observar a implementação dessas transformações e, nas Figuras 1 e 2, podemos observar a imagem lida e como ela ficou após as transformações descritas, respectivamente.

```
image = cv.imread('input_images/bitmap.pbm',0)
image = cv.bitwise_not(image)//255
```

Código 3: Salvar uma imagem com OpenCV.

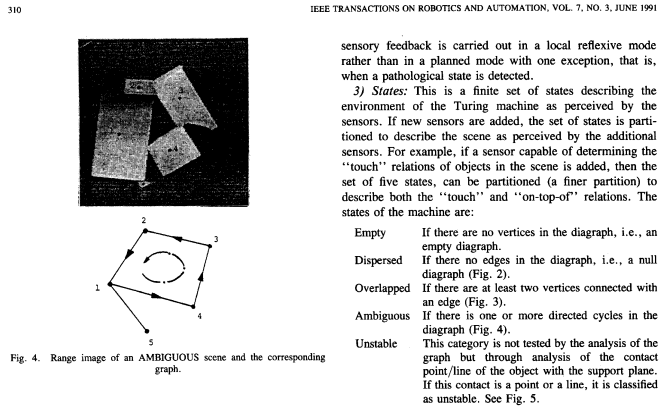
310 IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, VOL. 7, NO. 3, JUNE 1991

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: Imagem original do documento.

B. *Operações morfológicas*

Neste trabalho, três tipos de operações morfológicas foram empregadas: a Dilatação, a Erosão e o Fechamento. Essas operações podem ser vistas como um mapeamento entre uma imagem A e um elemento estruturante B (IV-C), ambos definidos em $\mathbb{Z}^2$.

A dilatação entre uma imagem A e um elemento estruturante B é realizada por meio de uma Soma de Minkowski, e é definida pela Equação 1.

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

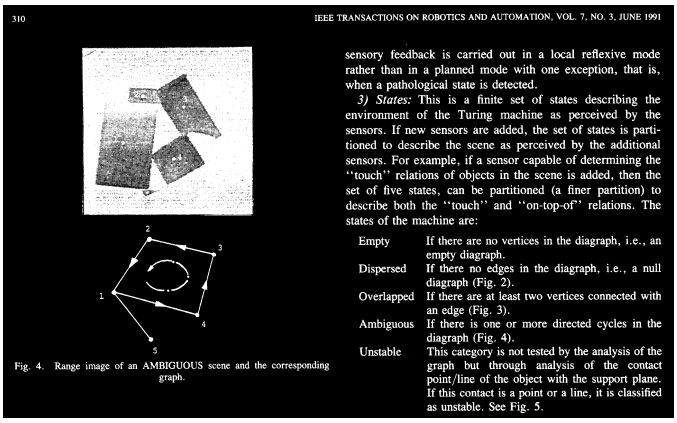
310 IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, VOL. 7, NO. 3, JUNE 1991

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 2: Imagem do documento após transformações.

A dilatação pode ser interpretada como o conjunto de todas as translações do elemento estruturante B pelos pontos da imagem A, de maneira que há, pelo menos, um elemento não nulo em comum com A [1]. Também podemos pensar na dilatação como uma operação que expande as dimensões do objeto e é capaz de preencher algumas lacunas existentes em seu interior, sendo esses resultados dependentes do elemento estruturante usado.

Já, a erosão de uma imagem A por um elemento estruturante B é realizada por meio de uma Subtração de Minkowski, e é definida pela Equação 2.

$$\mathcal{E}(A,B) = A \ominus B = \bigcap_{b \in B} (A - b) = \bigcap_{b \in \hat{B}} (A + b) \tag{2}$$

A erosão pode ser interpretada como o conjunto de todas as translações do elemento estruturante B pelos pontos da imagem A, de maneira que elas estejam contidas em A. Entretanto, se o elemento estruturante não contiver a origem, o resultado da erosão pode não ser subconjunto da imagem A [1]. Também podemos pensar na erosão como uma operação que reduz as dimensões do objeto e aumenta lacunas existentes em seu interior, sendo os resultados também dependentes do elemento estruturante empregado.

Por fim, o fechamento de uma imagem A por um elemento estruturante B é uma combinação das operações de dilatação e erosão, conforme apresentada na Equação 3.

$$A \bullet B = (A \oplus B) \ominus B \tag{3}$$

O fechamento geralmente é utilizado para juntar separações estreitas entre objetos, preencher pequenos buracos no interior de um objeto e preencher lacunas em seu contorno [1].

C. *Elementos estruturantes*

Elementos estruturantes são considerados pequenos conjuntos de *pixels*, ou uma pequena imagem, empregada quando desejamos analisar determinadas propriedades de uma imagem de interesse e processá-la de determinada maneira [3]. Esses elementos podem possuir diferentes formatos e os resultados

de suas aplicações dependem do ponto adotado como sua origem, que pode pertencer ou não ao elemento [1].

Neste trabalho, sete elementos estruturantes foram empregados, sendo três deles sugeridos na definição do trabalho, e os outros quatro escolhidos com o propósito de identificação de blocos de palavras. Além disso, todos eles tiveram suas origens definidas como pertencentes a eles e localizadas em seus centros, sendo implementados como *arrays numpy*.

O primeiro elemento estruturante, chamado de **kernel1_100**, possui 1 *pixel* de altura e 100 *pixels* de largura e foi empregado para operações de dilatação e erosão. Como será descrito posteriormente, esse elemento foi empregado para a identificação de linhas de texto. Dessa maneira, o objetivo de sua utilização na dilatação, seguida da erosão, foi expandir os objetos da imagem, horizontalmente, de maneira a conectar aqueles que estavam alinhados em relação ao eixo y da imagem e fechar buracos em seu interior.

O segundo elemento estruturante, chamado de **kernel200_1**, possui 200 *pixels* de altura e 1 *pixel* de largura, e também foi empregado para operações de dilatação e erosão, com o objetivo de reconhecimento de linhas de texto. A finalidade de seu uso na dilatação, seguida da erosão, foi a expansão dos objetos da imagem, verticalmente, de maneira a preencher buracos no interior desses objetos e ajudar na sua transformação em blocos mais ou menos preenchidos.

O terceiro elemento estruturante foi chamado de **kernel1_30** e possui 1 *pixel* de altura e 30 *pixels* de largura. Ele foi empregado em uma operação de fechamento, tendo como objetivo unir blocos de objetos, criados pela aplicação de outras operações morfológicas, que estavam próximos uns dos outros – proximidade determinada pelas dimensões do elemento estruturante –, além de preencher buracos em seu interior.

Os outros quatro elementos estruturantes foram empregados com o objetivo de identificar blocos de palavras. Dois deles foram empregados em operações de dilatação, **kernel6_6** e **kernel10_1**, um deles foi utilizado em operações de erosão, **kernel1_3** e o último foi usado em uma operação de fechamento, **kernel1_8**. Ademais, as dimensões de cada um deles foram determinadas de modo a permitir a junção de objetos da imagem que estavam alinhados em relação ao eixo y (horizontalmente), de maneira a formar blocos preenchidos, mas limitando-se às dimensões de uma palavra, ou seja, de maneira que objetos pertencentes a palavras diferentes não fossem unidos. Para isso, considerou-se o fato de que o espaçamento entre duas letras de uma mesma palavra é menor do que o espaçamento entre duas letras de palavras adjacentes.

D. Identificação de linhas de texto e blocos de palavras

Para a obtenção das informações pedidas no trabalho, duas abordagens foram empregadas: a primeira delas teve como objetivo a identificação de regiões que representam texto e não texto, e a transformação dos objetos da imagem, de maneira a construir blocos que representassem linhas; a segunda abordagem teve como objetivo a identificação de regiões que representam texto e não texto, e a transformação dos objetos de maneira a construir blocos que representassem palavras.

Na primeira abordagem, empregamos as etapas apresentadas na descrição do trabalho, que foram:

1) A dilatação da imagem original com o elemento estruturante *kernel1_100*.
2) A erosão da imagem resultante, com o mesmo elemento estruturante.
3) A dilatação da imagem original com o elemento estruturante *kernel200_1*.
4) A erosão da imagem resultante, com o mesmo elemento estruturante.
5) A interseção entre as imagens resultantes das Etapas 2 e 4.
6) O fechamento da imagem obtida na etapa anterior, com o elemento estruturante *kernel1_30*.
7) A aplicação de um algoritmo de identificação de componentes conexos na imagem resultante da etapa anterior.
8) O cálculo da razão entre o número de *pixels* pretos (objetos) e a área de cada retângulo envolvendo um componente conexo, assim como a razão entre o número de transições de branco para preto (fundo para objeto) de vizinhança quatro e o número de *pixels* pretos. É importante destacar que esses cálculos foram realizados empregando a imagem original.
9) A aplicação de uma regra para identificar, a partir dessas razões, as regiões de texto e de não texto.
10) A realização de operações para identificar e contar as linhas de texto.

Como, em alguns casos, mais de um componente conexo foi identificado em uma mesma linha, na Etapa 10 aplicamos operações para a identificar quais componentes conexos pertenciam a uma mesma linha e contabilizamos as linhas com mais de um componente conexo apenas uma vez.

Com relação à segunda abordagem, os seguintes passos foram seguidos:

1) A dilatação da imagem original com o elemento estruturante *kernel6_6*.
2) A erosão da imagem resultante, com o elemento estruturante *kernel1_3*.
3) A dilatação da imagem obtida no passo anterior, com o elemento estruturante *kernel10_1*.
4) A erosão, com o elemento estruturante *kernel1_3*, da imagem obtida na Etapa 3.
5) A interseção entre as imagens obtidas nas Etapas 2 e 4.
6) O fechamento da imagem resultante, com o elemento estruturante *kernel1_8*.
7) A aplicação de um algoritmo para a identificação de componentes conexos, sobre a imagem resultante do fechamento.
8) O cálculo das mesmas razões descritas na etapa 8 da Abordagem 1, entretanto, dessa vez sobre a imagem resultante do fechamento.
9) A identificação de regiões de texto e de não texto, de acordo com as razões obtidas.

10) A aplicação de operações para contar a quantidade de blocos de palavras.

As imagens produzidas até a Etapa6 das Abordagens 1 e 2 foram armazenada em dicionários chamados **line_processed_images** e **word_processed_images**, respectivamente. As imagens que representam os componentes conexos foram armazenadas *arrays numpy* chamados **line_image_labels** e **word_image_labels**, e as coordenadas de cada componte conexo foram salvas em *arrays* chamados **line_stats** e **word_stats**. Além disso, as coordenadas dos componentes que representam apenas texto foram armazenadas em *arrays* chamados **clean_line_stats** e **clean_word_stats**.

Por fim, para colocarmos retângulos envolventes nos blocos de palavras da imagem original, precisamos apenas utilizar as coordenadas dos retângulos envolventes, armazenadas no *array* **clean_word_stats**.

V. Resultados e discussões

Nos próximos parágrafos, apresentamos os testes realizados e os resultados obtidos para cada abordagem empregada e para cada uma de suas etapas.

A. Primeira abordagem e a contagem de linhas de texto

1) Etapas 1 a 7: Na Figura 3, podemos observar a imagem produzida na Etapa 1 da primeira abordagem. Conforme esperado, os objetos da imagem (letras, números e figuras), foram expandidos horizontalmente, unindo-se e formando blocos mais ou menos preenchidos.

Figura 3: Etapa 1 – Figura 2 após dilatação por elemento estruturante 1 x 100.

Já, na Figura 4, podemos observar o resultado da erosão pelo elemento estruturante *kernel1_100*, aplicada sobre a imagem mostrada na Figura 3. Podemos observar que, embora a maioria dos blocos formados na Etapa 1 tenham sido mantidos, todos eles tiveram suas dimensões reduzidas, o que é esperado da operação de erosão, e alguns voltaram a ter forma semelhante a de objetos observados na imagem original (Figura 2).

Na Figura 5, por sua vez, podemos observar o resultado da dilatação da imagem original pelo elemento estruturante *kernel200_1* (Etapa 3). Como esperado, os objetos da imagem foram expandidos verticalmente, devido à grande altura do elemento estruturante.

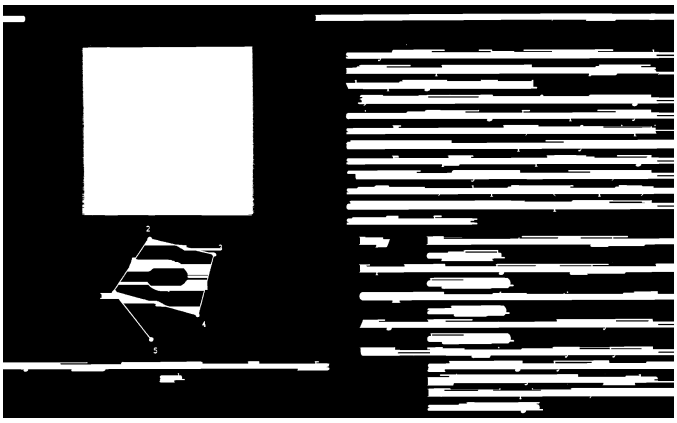

Figura 4: Etapa 2 – Figura 3 após erosão por elemento estruturante 1 x 100.

Figura 5: Etapa 3 – Figura 2 após dilatação por elemento estruturante 200 x 1.

Na Figura 6, podemos observar o resultado da erosão, também pelo elemento estruturante *kernel200_1* , da imagem apresentada na Figura 5. Conforme esperado de uma operação de erosão, as dimensões dos blocos criados foram reduzidas, alguns deles também apresentando formas semelhantes às encontradas na imagem original, como as letras vistas na parte inferior da imagem, abaixo das figuras.

Após obtermos as imagens das Etapas 2 e 4, calculamos a sua interseção, conforme pode ser visto na Figura 7. Como podemos observar, alguns dos blocos criados foram subdivididos, devido à interseção entre regiões de valor 1 da imagem da Etapa 2 e regiões de valor 0 da imagem da Etapa 4. Da mesma forma, a região em branco que juntava as duas figuras do lado esquerdo da imagem, apresentada na Figura 6, foi eliminada pela presença de *pixels* de valor zero nessa mesma região da Figura 4. Também, podemos observar que os números existentes na figura de baixo, da imagem original, ficaram novamente percepitíveis na imagem resultante da interseção.

Na Figura 8, podemos observar o resultado do fechamento

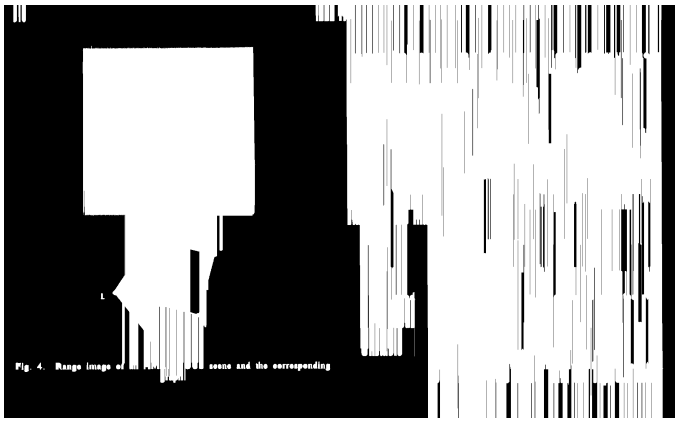

Figura 6: Etapa 4 – Figura 5 após erosão por elemento estruturante 200 x 1.

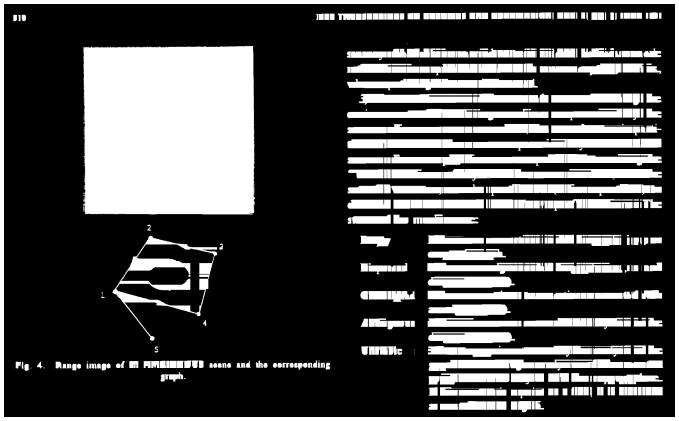

Figura 7: Etapa 5 – Interseção das imagens mostradas nas Figuras 4 e 6.

Figura 8: Etapa 6 – Fechamento aplicado sobre a imagem da Figura 7.

Figura 9: Etapa 8 – Retângulos indicando componentes conexos identificados na Figura 8.

aplicado sobre a imagem da Figura 7, com o elemento estruturante *kernel1_30*. Podemos observar que as regiões pretas que subdividiam alguns dos blocos criados na Etapa 2 foram preenchidas, o que caracteriza a junção de separações estreitas entre objetos esperada de uma operação de fechamento. Da mesma maneira, as palavras da legenda das figuras da imagem original foram preenchidas, formando três blocos diferentes, caracterizando uma operação de preenchimento de pequenos buracos, também esperada do fechamento.

Após o fechamento, calculamos os componentes conexos da imagem resultante dessa operação (Etapa 7). Na Figura 9, podemos observar os blocos da Figura 8 envolvidos por retângulos, que indicam os componentes conexos encontrados. Podemos observar que tanto regiões de texto como de figuras foram identificadas como componentes conexos. Além disso, elementos indesejados também foram identificados, como partes de uma das figuras e pequenos blocos criados devido a ruídos presentes na imagem original e resultantes da interseção realizada na Etapa 5. Também podemos perceber que o fundo da imagem também foi considerado um componente conexo.

2) Etapas 8 e 9: Na Etapa 8, utilizamos as coordenadas dos retângulos envolventes obtidos na Etapa 7 e a imagem original, apresentada na Figura 2, para calcular as razões.

Com relação à razão entre a quantidade de *pixels* pretos dentro de um retângulo e a área do mesmo retângulo envolvente, que aqui chamaremos de Razão 1, foi possível observar que apenas uma das figuras presentes na imagem original apresentou valor que nos permitiu a diferenciar do restante das regiões, sendo ele igual a 0.06213158654934691. Também, alguns componentes conexos bem pequenos obtiveram razão igual a zero.

Assim, de acordo com essa métrica, na Etapa 9 criamos uma regra que considerou todo componente conexo com um valor de Razão 1 menor do que 0.07 uma região de não texto, nos permitindo identificar a figura de baixo e os pequenos componentes conexos que optamos por ignorar.

Por sua vez, a razão entre o número de transições de branco para preto e o número de *pixels* pretos dentro de um retângulo envolvente, chamada aqui de Razão 2, nos permitiu estabelecer um outro limiar para identificar regiões de não texto, e também gerou resultados iguais a zero para alguns dos componentes.

Dessa maneira, na Etapa 9 adicionamos mais uma regra, indicando que todo componente conexo com valor de Razão 2 menor do que 0.46 fosse considerado como não texto.

Além disso, também observamos que, no cálculo dos componentes conexos, o fundo da imagem sempre é classificado como o componente conexo de *label* igual a 0. Assim, utilizamos essa informação para indicar que o fundo também não é uma região de texto.

Para indicar se um componente conexo é ou não um texto, na Etapa 9 criamos um dicionário, chamado **line_component_classes**, em que a chave indica o *label* do componente conexo e o valor indica se ele é ou não uma região de texto, representado por 1 ou 0, respectivamente.

Na Figura 10, podemos observar os retângulos envolvendo os componentes conexos considerados textos. Como esperado, as figuras deixaram de ser consideradas texto, apesar de que a maioria dos números presentes em uma delas continuou a ser considerada.

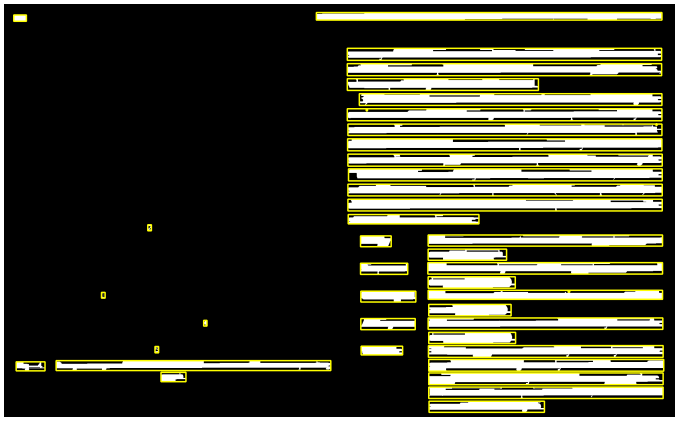

Figura 10: Etapas 8 e 9 – Retângulos indicando componentes conexos considerados texto.

3) Etapa 10: Por fim, precisamos contar a quantidade de linhas de texto. Entretanto, conforme pode ser observado da Figura 10, algumas linhas possuem mais de um componente conexo.

Para resolver esse problema, para cada coordenada de um retângulo envolvente de um componente conexo, verificamos se algum dos outros componentes possuia coordenada inicial y dentro do intervalo de valores das coordenadas y do componente utilizado como referência.

Então, salvamos essas informações em um dicionário chamado **same_line_dict**, cujas chaves são os *labels* dos componentes conexos de referência e os valores são listas de *labels* dos componentes conexos que estão na mesma linha que o de referência. Por fim, percorremos esse dicionário, contando uma linha para cada chave que possuia uma lista vazia, removendo as chaves que possuiam componentes na lista de outro componente e contando apenas uma linha para cada componente que tinha outros componentes em sua lista.

Ao final dessa etapa, obtivemos um total de **26 linhas de texto**, que é o valor esperado ao levarmos em consideração que a legenda das figuras foi considerada como estando na mesma linha de uma das linhas de texto da direita, assim como o número presente no canto superior esquerdo da imagem original e os números da figura de baixo que permaneceram.

B. Segunda abordagem e contagem de palavras.

1) Etapas 1 a 7: Na Figura 11, podemos observar o resultado da dilatação da imagem original pelo elemento estruturante *kernel6_6*. Podemos perceber que os objetos da imagem tiveram suas dimensões expandidas e alguns buracos internos preenchidos. Entretanto, como esperado, palavras diferentes não chegaram a se juntar, como aconteceu na Etapa 1 da Abordagem 1.

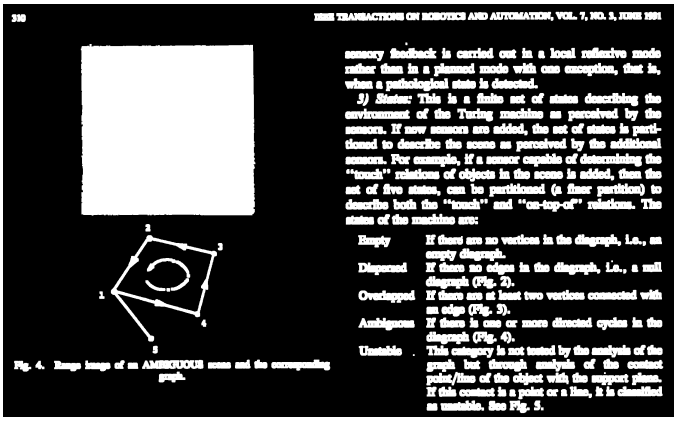

Figura 11: Etapa 1 – Figura 2 após dilatação por elemento estruturante 6 x 6.

Na Figura 12, podemos observar o resultado erosão da imagem presente na Figura 11 pelo elemento estruturante *kernel1_3*. Como o elemento estruturante é pequeno, com relação à largura e à altura da maioria dos objetos, percebemos apenas uma pequena redução das dimensões desses objetos e um aumento de alguns buracos internos.

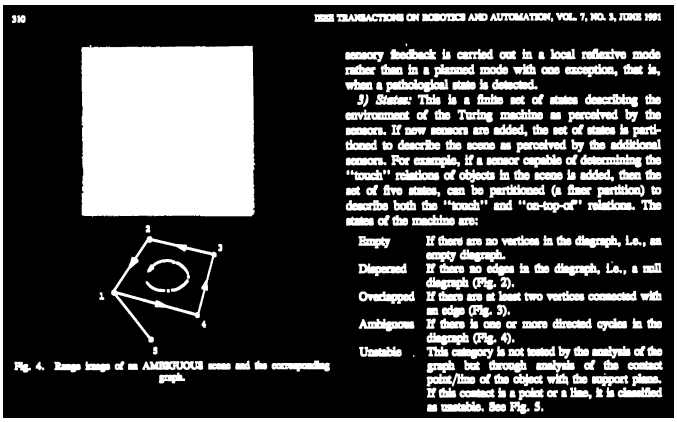

Figura 12: Etapa 2 – Imagem da Figura 11 após erosão por elemento estruturante 1 x 3.

Já, na Figura 13, podemos observar o resultado da dilatação, da imagem mostrada na Figura 12, pelo elemento estruturante *kernel10_1*. Vemos que, conforme esperado, os objetos também tiveram suas dimensões expandidas, dessa vez na direção vertical, e que palavras também não se conectaram com outras, da mesma forma que figuras também não.

Além disso, buracos internos foram preenchidos e pequenas separações foram fundidas.

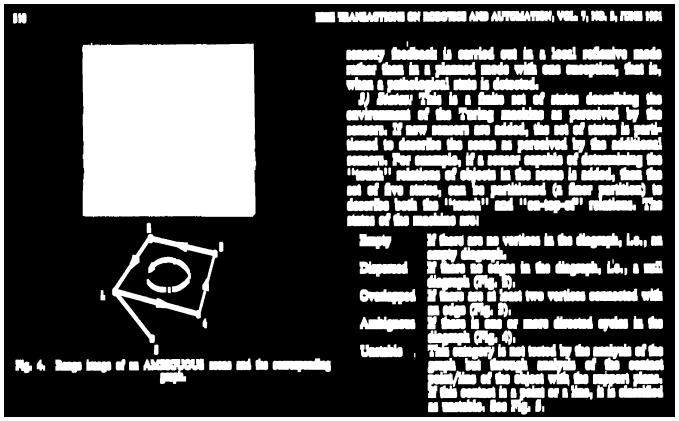

Figura 13: Etapa 3 – Imagem da Figura 12 após dilatação por elemento estruturante 10 x 1.

Na Figura 14, observamos o resultado da erosão da imagem da Figura 13 pelo elemento estruturante *kernel1_3*. Novamente, percebemos que essa operação causou poucas transformações na imagem dada como entrada, assim como esperado.

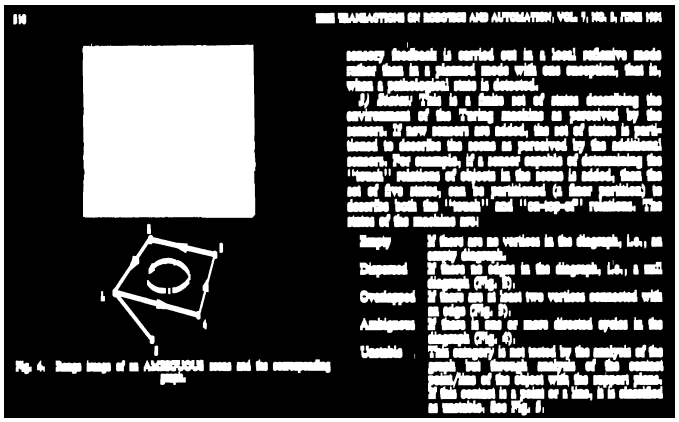

Figura 14: Etapa 4 – Imagem da Figura 13 após erosão por elemento estruturante 1 x 3.

Na Etapa 5, realizamos a interseção entre as imagens obtidas nas Etapas 2 e 4, e o resultado pode ser visto na Figura 15. Observamos que a imagem produzida pela interseção possui objetos com dimensões maiores, apesar de ainda conseguirmos reconhecer o formato da maioria deles. Além disso, percebemos que, embora palavras diferentes não tenham sido conectadas, letras pertencentes a uma mesma palavra estão mais unidas, conforme esperado.

Depois, na Etapa 6, realizamos o fechamento da imagem obtida na Etapa 5, utilizando o elemento estruturante *kernel1_8*, cujo resultado pode ser visto na Figura 16. Podemos observar que as letras das palavras conectaram-se ainda mais, formando blocos de palavras mais ou menos preenchidos, como esperado, devido à característica da operação de fechamento de fundir pequenas separações entre objetos e preencher pequenos buracos. Ademais, a separação entre as palavras foi preservada, assim como aquela entre figuras. Apesar disso, algumas palavras, como *point* e *line*, que estavam separadas por algum separador, como a barra (/), foram unidas. Também, acentos e pontuações foram considerados como pertencentes às palavras mais próximas.

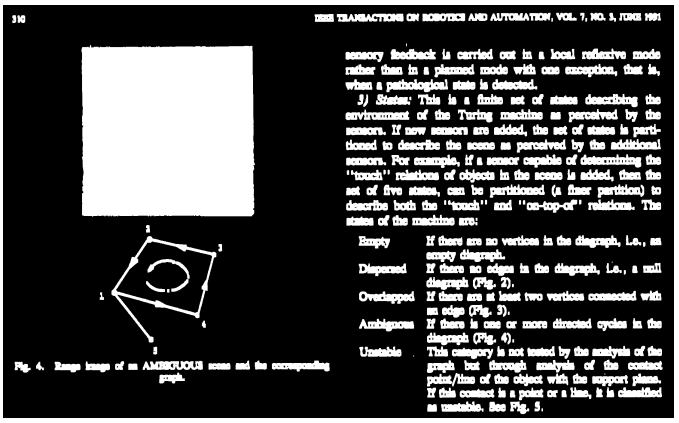

Figura 15: Etapa 5 – Interseção das imagens mostradas nas Figuras 12 e 14.

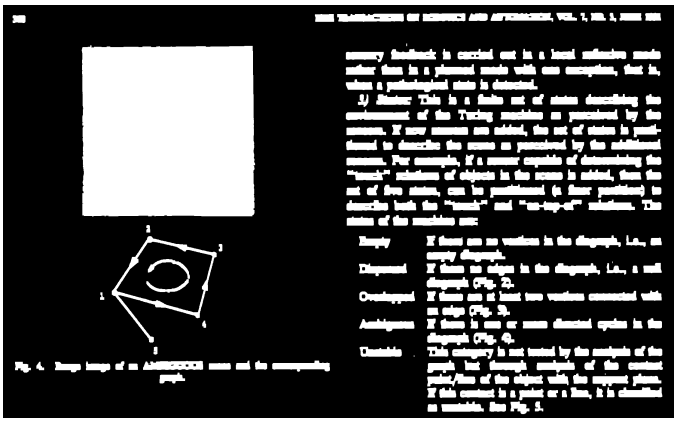

Figura 16: Etapa 6 – Fechamento aplicado sobre a imagem da Figura 15.

Por fim, na Etapa 7, aplicamos um algoritmo para identificação de componentes conexos, empregando como entrada a imagem apresentada na Figura 16. Na Figura 17, podemos observar retângulos envolvendo os blocos da Figura 16 considerados componentes conexos. Conforme podemos observar, os retângulos envolvem tanto blocos de palavras, como figuras e partes delas. Além disso, alguns pequenos elementos indesejados foram considerados componentes conexos, assim como o fundo da imagem.

2) Etapas 8 e 9: As Etapas 8 e 9 da Abordagem 2 são semelhantes às respectivas etapas da Abordagem 1. Entretanto, ao invés de calcularmos as razões sobre a imagem original, calculamo-nas sobre a imagem resultante do fechamento, mostrada na Figura 16.

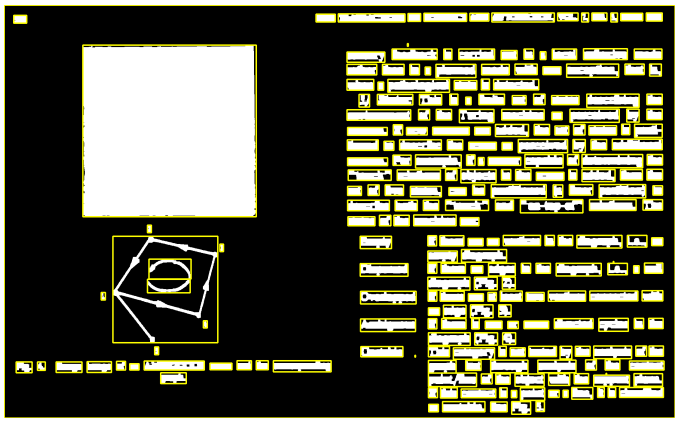

Figura 17: Etapa 7 – Retângulos envolvendo componentes conexos da imagem da Figura 16.

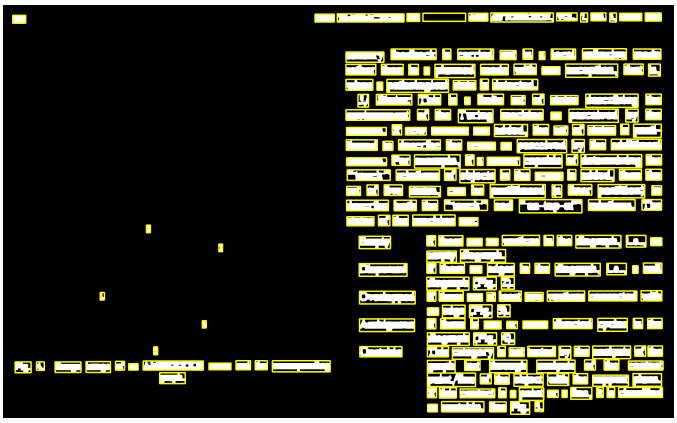

Figura 18: Etapas 8 e 9 – Retângulos envolvendo regiões de texto.

Novamente, cada uma das razões gerou uma regra diferente na Etapa 9. Para a razão do número de *pixels* pretos por área de retângulo envolvente, estabelecemos que todos os componentes conexos com valores dessa razão menores do que 0.27 devem ser considerados como região de não texto.

Além disso, para a razão entre o número de transições de branco para preto e a quantidade de *pixels* pretos em um retângulo envolvente, todo componente conexo com valor dessa razão menor do que 0.008 também não foi considerado como uma região de texto. Também, observando novamente que o fundo da imagem foi considerado o componente conexo de *label* igual a zero, determinamos que esse componente também não representa texto.

Para isso, salvamos as informações em um dicionário, chamado **word_component_classes**, em que as chaves representam as *labels* dos componentes conexos e os valores representam se o componente é uma região de texto ou não, sendo atribuídos valores 1 ou 0, respectivamente.

Na Figura 18, podemos observar os retângulos envolvendo apenas as regiões da imagem consideradas texto. Observamos que, tanto as figuras, como os pequenos ruídos que eram encontrados logo acima da primeira linha do primeiro parágrafo, e entre a primeira e a segunda palavra do último parágrafo, também não foram considerados texto.

3) Etapa 10: Para a contagem do número de blocos de palavras, observamos quantos componentes conexos indicados no dicionário *word_component_classes* possuiam valores iguais a 1. Dessa forma, obtivemos um total de **242 blocos de palavras**, que é próximo do valor esperado de 244, levando em consideração os números da figura de baixo, definindo acentos e pontuações como pertencentes à palavra mais próxima, e considerando números com mais de um dígito como uma palavra.

Também é importante destacar que nosso algoritmo considerou como uma única palavra os trechos ”*point/line*”, ”on-top-of”e considerou como duas palavras o trecho ”parti-tioned”, em que primeira parte está em uma linha e a segunda está em outra.

C. Mostrar retângulos envolventes em blocos de palavras.

Para capturarmos apenas os retângulos que envolvem palavras, criamos uma lista chamada **clean_word_stats**, em que armazenamos apenas as coordenadas dos retângulos que envolvem regiões de texto, descobertos na Etapa 9 e obtidos a partir do dicionário **word_stats**, que armazena todos os retângulos envolventes. O resultado pode ser observado na Figura 19.

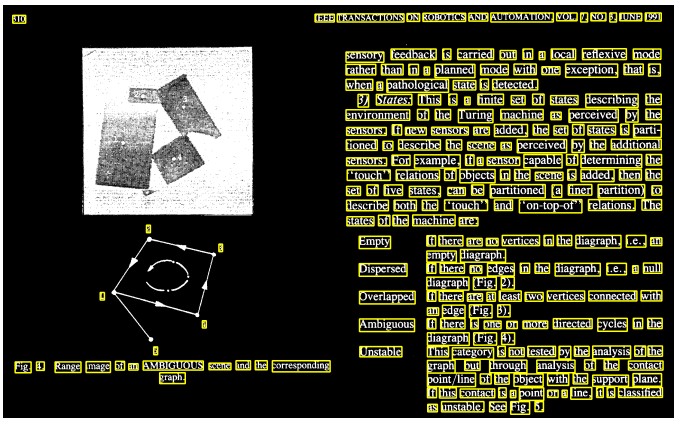

Figura 19: Retângulos envolvendo blocos de palavras na imagem original.

VI. Conclusão

Conforme pôde ser observado, conseguimos fazer o que foi pedido no Trabalho 04. Foram aplicadas operações morfológicas sobre uma imagem binária dada como entrada, de maneira a se identificar regiões de texto e contar a quantidade de linhas e palavras, além de mostrarmos os retângulos envolvendo os blocos de palavras.

Foram necessários sete diferentes elementos estruturantes e apenas três tipos de operações morfológicas: a dilatação, a erosão e o fechamento. Além disso, de acordo com a ordem de aplicação das operações empregada na Abordagem 1, as

Etapas 1 e 2 e as Etapas 3 e 4 poderiam ser simplificadas em duas operações de fechamento.

Também observamos que, embora nosso algoritmo não tenha conseguido o número exato de blocos de palavras, alcançamos um valor bem próximo.

Por fim, percebemos que as operações, os elementos estruturantes e os valores empregados nas regras desenvolvidas na Etapa 9 de cada abordagem foram especifícos para a imagem utilizada para testes, sendo possivelmente necessário alterá-los quando empregarmos outras imagens, que possuem textos, figuras e organizações diferentes. Dessa forma, percebemos as limitações de nossa abordagem, mas experimentamos o interessante desafio de empregar operadores morfológicos para as questões sugeridas.

REFERÊNCIAS

[1] H. Pedrini and W. R. Schwartz, *Análise de imagens digitais: princípios, algoritmos e aplicações*. Thomson Learning, 2008. 1, 2, 3

[2] C. A. B. De Mello, A. L. I. de Oliveira, and W. P. Dos Santos, *Digital document analysis and processing*. Nova Science Publishers, 2012. 1

[3] R. C. Gonzalez, R. E. Woods, and B. R. Masters, "Digital image processing third edition," *Pearson International Edition*, 2008. 2