Relatório Trabalho 3

MC920 - Introdução ao Processamento de Imagem Digital

Victor Costa Dominguite - RA: 245003

Maio 2023

1 Introdução

Neste trabalho foi feita a aplicação de operadores morfológicos em uma imagem binária que continha texto, em busca de extrair informações a respeito desse, tais como número de linhas e número de palavras. Nesse contexto, os operadores morfológicos atuaram na identificação de objetos e na consequente identificação desses como elementos textuais ou não.

Neste relatório serão discutidos os passos tomados para realizar as operações, os algoritmos empregados, as estruturas de dados utilizadas, assim como possíveis limitações da solução apresentada e desafios enfrentados.

2 Execução dos programas

O trabalho foi inteiramente resolvido em um arquivo (t3.py), sendo cada passo executado sequencialmente e os resultados apresentados na tela. Para executar esse arquivo, basta rodar seu *script* em python (isto é, em um terminal, executar o comando `python3 t3.py`).

Vale notar que a imagem utilizada ("bitmap.pbm") deve estar no mesmo diretório do programa que está sendo executado.

É necessário também que os pacotes `cv2`, `numpy` e `matplotlib` estejam instalados, pois o programa os utilizam para realizar a leitura, manipulação e apresentação das imagens.

3 Análise das soluções e resultados obtidos

3.1 Processamento inicial

A leitura da imagem foi feita com o uso da função `imread` do `cv2`, a qual devolve uma matriz bidimensional do tipo `numpy.array` contendo os valores de cada pixel na imagem.

A biblioteca `cv2` também foi utilizada para a aplicação dos operadores morfológicos. Porém, vale ressaltar que as funções utilizadas, como `cv2.erode` ou `cv2.dilate`, consideram a cor branca (com bits de valor 1) como objetos e a cor preta (com bits de valor 0) como fundo. Dessa forma, para a manipulação da imagem, foi necessário obter seu negativo, uma vez que na imagem original o texto, que deve ser considerado como o objeto, é da cor preta e o fundo é branco. Com a obtenção do negativo, essa relação se inverte e é possível aplicar as funções do `cv2` sem quaisquer problemas, já que o texto se torna branco e, portanto, é considerado como objeto. Tanto a imagem original quanto a imagem negativada utilizada nas operações podem ser observadas nas figuras 1 e 2 a seguir.

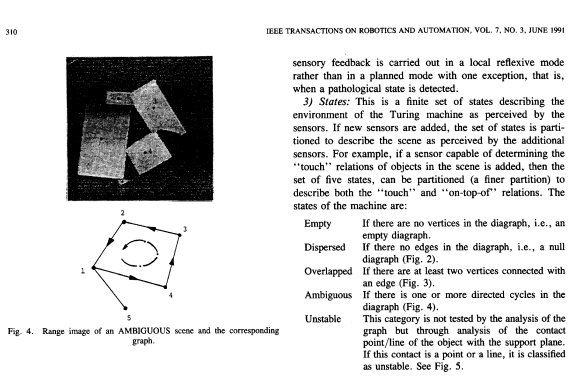

Figura 1: imagem original

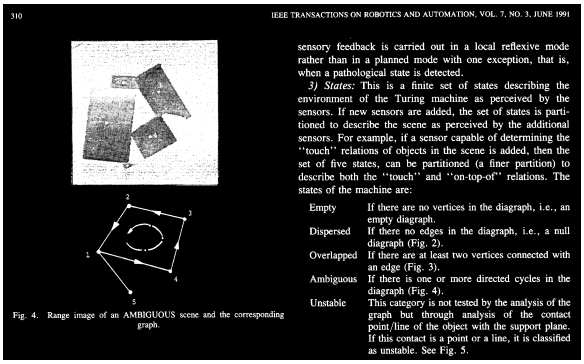

Figura 2: imagem negativada para aplicação dos operadores morfológicos

3.2 Aplicação dos operadores morfológicos

3.2.1 Passo 1

A primeira operação a ser executada na imagem era sua dilatação por um elemento estruturante de 1 pixel de altura por 100 pixels de largura. Esse elemento estruturante foi criado com a função `np.ones` que devolve um `np.array` com um *shape* passado como argumento, sendo todos os elementos de valor 1. O resultado dessa operação pode ser observado na Figura 3.

Figura 3: Resultado da aplicação da dilatação com elemento estruturante de 1 x 100 pixels

Percebe-se que a dilatação causou uma união das linhas do texto, perdendo completamente a informação a respeito das palavras. Ao mesmo tempo, a imagem que estava presente no canto superior esquerdo tornou-se um grande retângulo branco. É possível observar em alguns pontos traços finos que fogem das linhas, que podem ser originados, por exemplo, por pingos em i que foram dilatados.

3.2.2 Passo 2

O segundo passo consistiu na erosão da imagem resultante do passo anterior, mantendo o mesmo elemento estruturante. Assim, ao realizar uma dilatação seguida de uma erosão, foi feita uma operação equivalente à aplicação de uma fechamento. O resultado está apresentado na Figura 4.

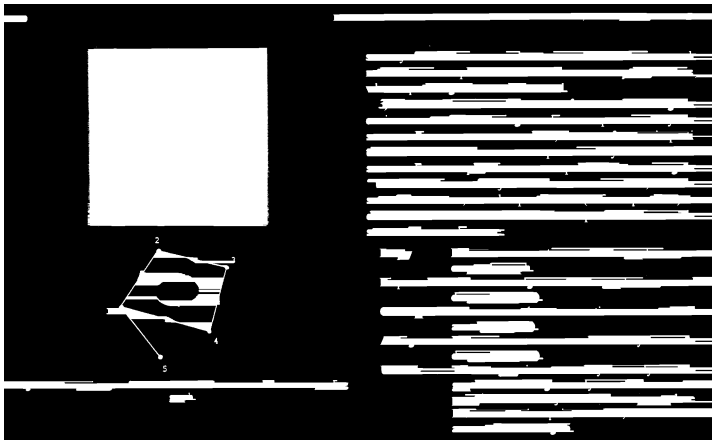

Figura 4: Resultado da erosão na imagem resultante do passo 1

Ao analisar o resultado, percebe-se que, em relação à imagem apresentada na Figura 3, houve um afinamento dos objetos, o que era esperado, considerando a natureza da operação de erosão. Com isso, há uma melhor definição e isolamento das linhas e demais objetos da imagem.

3.2.3 Passo 3

Nessa etapa, houve uma redefinição do elemento estruturante, o qual passou a ter 200 pixels de altura por 1 de largura. A seguir, foi aplicada uma dilatação na imagem da figura 2 utilizando esse novo *kernel*. O resultado está apresentado na Figura 5.

Figura 5: Resultado da dilatação com elemento estruturante de 200 x 1 pixels

Nitidamente, houve um prolongamento vertical dos objetos, o que é condizente com o formato do elemento estruturante utilizado na dilatação. Com isso, ainda é possível observar uma certa separação de colunas do texto, porém, parágrafos alinhados numa mesma coluna formam praticamente um grande objeto.

3.2.4 Passo 4

Nesse passo foi realizada a erosão da imagem resultante do passo anterior, mantendo o mesmo elemento estruturante. Novamente, tem-se uma sequência de operações que equivalem à aplicação de uma fechamento na imagem. A imagem resultante pode ser observada na Figura 6 a seguir.

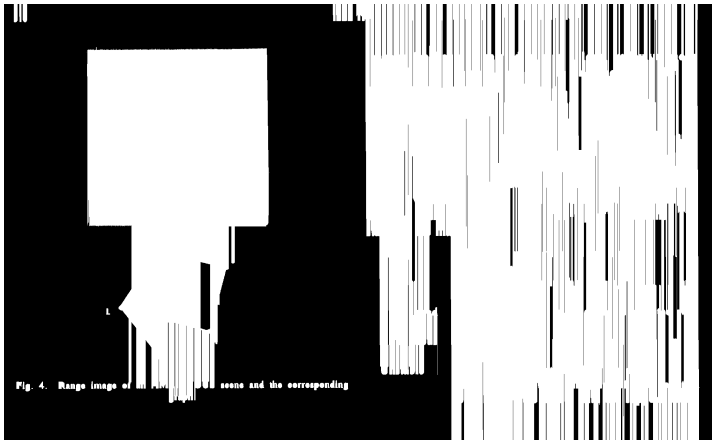

Figura 6: Resultado da erosão na imagem resultante do passo 3

Assim como no passo 2, percebe-se que houve um afinamento já esperado dos objetos quando comparado ao resultado anterior (Figura 5). Assim, houve uma melhor definição das colunas do texto, além do surgimento de diversas lacunas no meio do objeto correspondente ao texto principal.

3.2.5 Passo 5

Para o quinto passo, foi realizada a operação lógica AND nas imagens resultantes dos passos 2 e 4 (Figuras 4 e 6), com o objetivo de obter a intersecção entre as imagens. O resultado está apresentado na Figura 7.

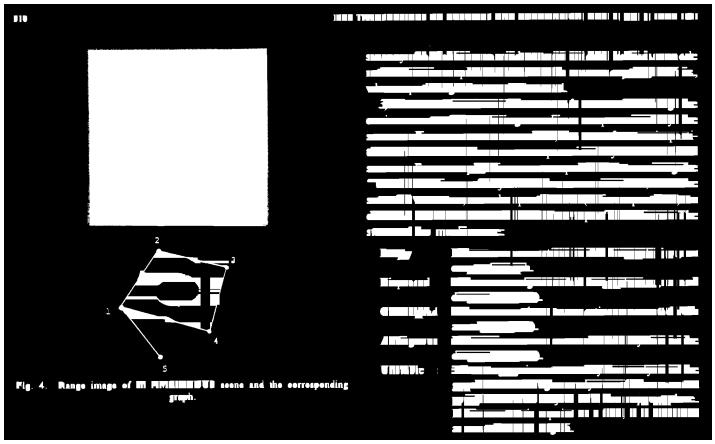

Figura 7: Resultado da intersecção das imagens resultantes dos passos 2 e 4

Na imagem da Figura 7, é possível discernir bem as linhas, embora ainda haja diversas descontinuidades nelas.

3.2.6 Passo 6

Finalmente, foi realizado um fechamento, isto é, uma dilatação seguida de erosão, na imagem resultante do passo 5 (apresentada na Figura 7), utilizando um elemento estruturante de 1 pixel de altura e 30 pixels de largura.

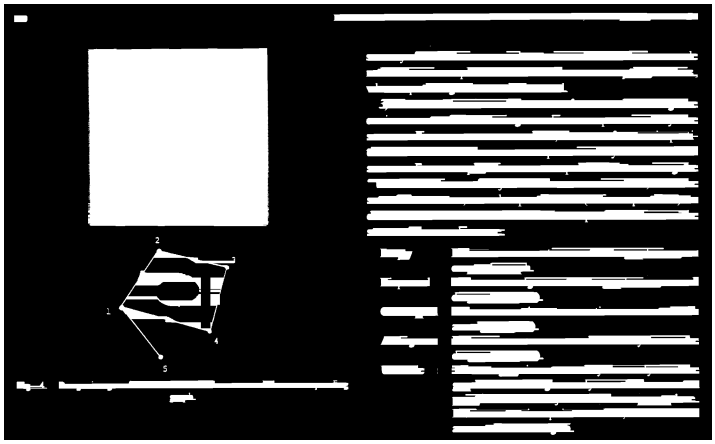

Figura 8: Resultado da aplicação do fechamento na imagem resultante do passo 5

Observa-se que, em relação à imagem resultante do passo anterior (Figura 7), houve um preenchimento das lacunas nas linhas, as quais agora estão muito mais próximas de blocos contínuos, se assemelhando a retângulos.

3.3 Identificação de objetos e elementos textuais

3.3.1 Componentes conexos

Após a aplicação das operações morfológicas descritas na seção anterior, foi realizada a detecção de componentes conexos na imagem resultante do passo 6, apresentada na Figura 8. Para realizar essa operação, foi utilizada a função `connectedComponents` da biblioteca `cv2`, a qual recebe como argumento uma matriz (no caso, a imagem) e devolve tanto o número de componentes conexos, quanto uma imagem em que os componentes conexos estão separados por números distintos, isto é, todos os pixels de um mesmo componentes conexo tem o mesmo valor, o qual é diferente dos valores de pixels de outros componentes conexos.

Para uma melhor visualização dos resultados, foi feita a conversão do resultado devolvido pela função `cv2.connectedComponents` para o modelo HSV, com a matiz (camada H) correspondendo ao próprio valor devolvido pela função, redimensionado para o intervalo $[1, 179]$ que são os valores aceitos pelo `cv2`, as camadas de saturação (S) e valor (V) possuem valor 1 para todos os pixels. Em seguida, a imagem foi convertida para RGB para que pudesse ser apresentada na tela, de forma que cada componente conexo fosse representado por uma cor diferente.

Os componentes conexos identificados na imagem resultante do passo 6 podem ser observados na Figura 9 a seguir.

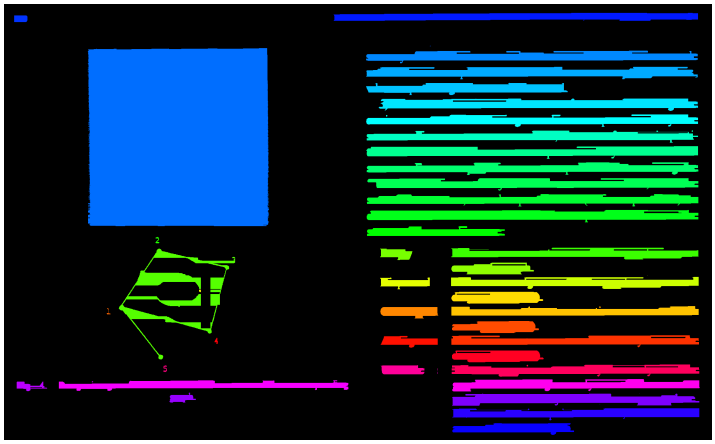

Figura 9: Componentes conexos identificados na imagem gerada pelo passo 6

3.3.2 Classificação de componentes como texto ou não texto

Já havendo identificado os componentes conexos presentes na imagem, foi feita uma classificação de cada um deles como elementos que constituíam *texto* ou *não texto*. Para isso, primeiramente, foi desenhado um retângulo em torno de cada componente conexo.

A obtenção desses retângulos foi feita utlizando majoritariamente funções do `cv2`. Em primeiro lugar, a função `cv2.findContours` foi utilizada para obter os contornos dos componentes conexos da imagem. Em seguida, para cada contorno, foi aplicada a função `cv2.boundingRect`, a qual devolve as coordenadas `x` e `y` do canto superior esquerdo do retângulo que envolve o componente conexo, assim como sua largura e altura.

Vale notar que foi criada uma classe denominada `Rect` para representar um retângulo. Cada objeto dessa classe apenas armazena os 4 atributos (`x`, `y`, `w` e `h`) devolvidos pela função `cv2.boundingRect`, que representam o retângulo.

Cada retângulo identificado na imagem foi então armazenado num *array* de objetos do tipo `Rect`. Por fim, os retângulos são todos desenhados sobre a imagem original com o uso da função `rectangle` do `cv2`, a qual recebe as coordenadas para o canto superior esquerdo e inferior direito de um retângulo e o desenha numa imagem.

Os retângulos desenhados ao redor dos componentes conexos podem ser observados na Figura 10 a seguir.

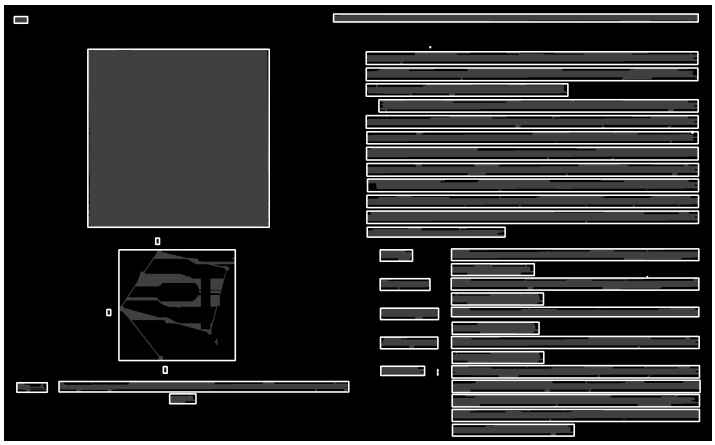

Figura 10: Retângulos desenhados ao redor de componentes conexos

A seguir, para cada um dos retângulos, desejava-se identificar seu conteúdo como *texto* ou *não texto*. Esse procedimento foi feito seguindo dois critérios: primeiramente a razão entre o número de

pixels pretos e o número total de pixels dentro do retângulo (critério 1) e, segundamente, a razão entre o número de transições verticais e horizontais branco para preto e o número total de pixels pretos (critério 2).

Nesse caso, para classificação de um objeto como texto, foi definido que o número gerado pelo critério 1 deve estar no intervalo $(0.2, 0.4)$, enquanto para o critério 2, a razão deve ser um número no intervalo $(0.3, 0.38)$. Os valores para essas constantes foram obtidos empiricamente, sendo que foram testados diversos valores diferentes até que fossem obtidos resultados mais satisfatórios, isto é, que identificassem com maior acurácia os elementos que fossem texto ou não.

Por fim, os retângulos que foram definidos como contendo conteúdo textual podem ser observados na imagem da Figura 11.

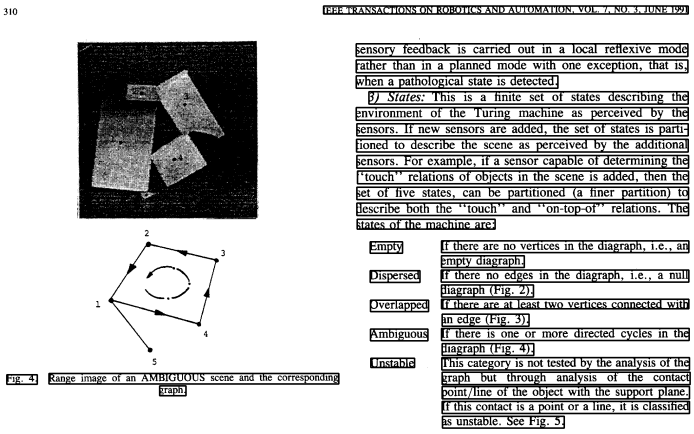

Figura 11: Retângulos desenhados ao redor de elementos detectados como texto

A partir disso, foi possível realizar uma estimativa do número total de linhas do texto, simplesmente contando o número de retângulos que foram identificados como contendo texto. No caso o programa contou um total de 34 linhas.

Claramente, pelos retângulos que podem ser vistos na imagem presente na Figura 11, é possível observar que essa contagem não é 100% precisa. Elementos alinhados horizontalmente, porém com um grande espaçamento uns dos outros, são considerados como de linhas distintas e, portanto, resultam como 2 linhas no total. Em alguns casos isso pode ser positivo, por exemplo, caso o texto tiver mais de uma coluna, linhas de colunas diferentes devem contar como 2 linhas diferentes, mesmo que estejam alinhadas horizontalmente. Por outro lado, quando dois elementos são da mesma coluna e da mesma linha, porém encontram-se muito separados uns dos outros, eles serão contados como duas linhas. Isso ocorre algumas vezes no setor inferior direito da imagem, em que algumas linhas se iniciam com um pequeno título. Por outro lado, essas são ocorrências pontuais e não causam um impacto muito grande na contagem final de linhas.

3.3.3 Contagem de palavras

Na parte final do trabalho, foram realizados procedimentos muito semelhantes aos anteriores utilizados na identificação de linhas, porém agora com o objetivo de identificar as palavras do texto.

Primeiramente foi realizada uma dilatação na imagem original negativada (apresentada na Figura 2). O elemento estruturante utilizado foi consideravelmente menor que os demais utilizados nos outros passos do trabalho, sendo de apenas 1 pixel de altura por 11 de largura. Essas menores dimensões se devem à nova natureza do problema, sendo que os espaços entre palavras distintas agora devem ser mantidos, para que elas sejam identificados como componentes conexos diferentes. Ao mesmo tempo, as letras de uma mesma palavra devem ser unidas pela dilatação, para que elas não sejam contadas individualmente como pertencentes a objetos distintos.

O resultado da operação de dilatação pode ser observado na Figura 12 a seguir.

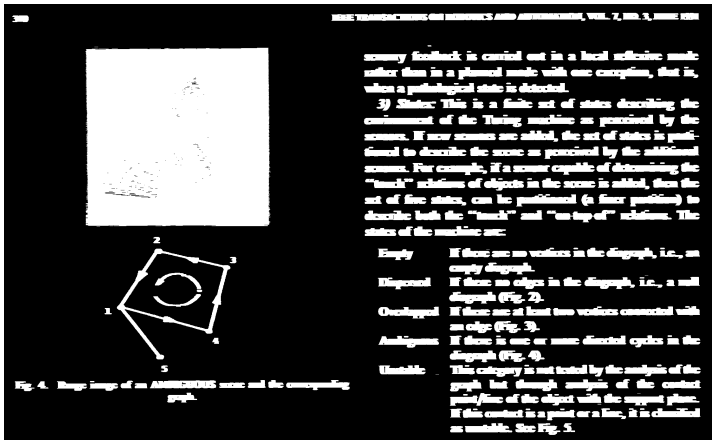

Figura 12: Resultado da dilatação da imagem original por um elemento estruturante de 1 x 11 pixels

Observa-se que o resultado obtido está adequado para a aplicação desejada. A maioria das palavras aparenta apresentar suas letras unidas, enquanto palavras distintas ainda continuam separadas. Além disso, os sinais de pontuação, que não devem ser levados em consideração na contagem de palavras, também foram em sua maioria unidos às palavras adjacentes a eles, o que fará com que eles pertençam a um mesmo componente conexo que a palavra e, portanto, não afetem a contagem final.

A seguir, assim como havia sido feito anteriormente, foram aplicadas as funções `findContours` e `boundingRect` do `cv2` para encontrar os retângulos que envolvem cada componente conexo.

O resultado dessa etapa pode ser visto na Figura 13.

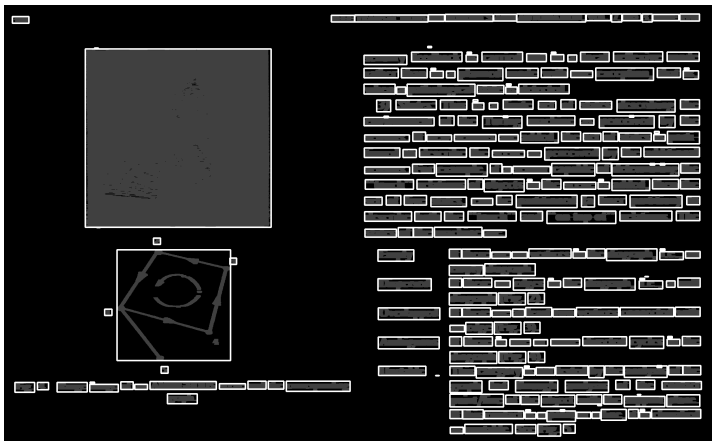

Figura 13: Retângulos envolvendo objetos identificados na imagem da Figura 12

A partir disso, resta classificar o conteúdo de cada retângulo como texto ou não texto. Para isso, foi utilizado exatamente o mesmo procedimento, com os mesmos critérios da detecção de textos para as linhas. Porém, os intervalos considerados para cada critério foram alterados. Para essa etapa, foi definido que a razão entre o número de pixels pretos e o número total de pixels dentro do retângulo deve estar no intervalo $(0.14, 0.67)$ e a razão entre o número de transições verticais e horizontais branco para preto e o número total de pixels pretos deve estar no intervalo $(0.28, 0.47)$. Essa diferença nos intervalos se deve à sequência de operações que foram aplicadas em cada caso. Para o caso das linhas, foram realizadas uma sequência operações que geraram retângulos menos rentes ao texto, enquanto no caso para a detecção de palavras, foi realizada apenas uma dilatação, o que gerou a formação de retângulos que ficam mais próximos de tangenciar os textos. Assim, a aplicação de diferentes operadores morfológicos ocasionou a detecção de componentes conexos e consequentemente retângulos que os cercam com tamanhos e proporções diferentes, o que acaba por interferir nas proporções de

pixels de cada cor dentro de um retângulo, assim como na proporções de transições.

O resultado para a identificação de elementos textuais pode ser observado na Figura 14

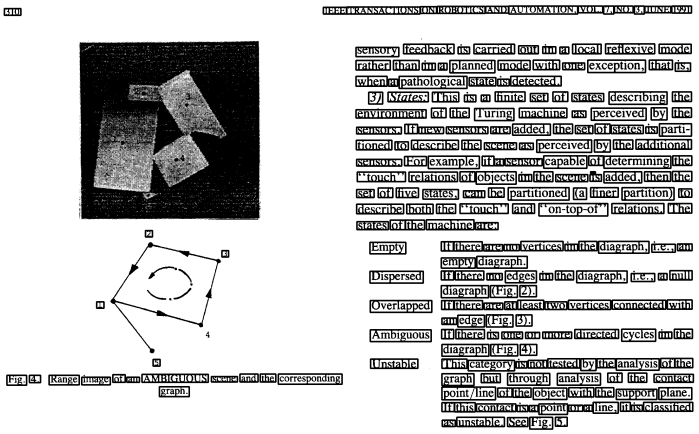

Figura 14: Retângulos envolvendo elementos detectados como texto na imagem original

Como é possível observar pela imagem acima, o processo adotado para a detecção de palavras aparenta apresentar um desempenho satisfatório em relação à acurácia da detecção de elementos textuais, porém, não é à prova de falhas, sendo possível constatar algumas inconsistências. Por exemplo, palavras que são quebradas por um hífen ao final de uma linha e continuam na linha seguida serão contadas como duas palavras. Além disso, alguns sinais de pontuação ou mesmo pingos nas letras i ou j podem acabar sendo contabilizados como palavras caso a dilatação não for capaz de uni-los às letras de uma palavra. No caso dos pingos em i e j, isso pode ocorrer quando não há nenhuma letra mais "alta"próxima do pingo, dificultando que a dilatação una esse pingo à palavra, pois não haverá nenhuma letra em sua altura. Ainda assim, observa-se que esses casos são exceções e quase não ocorrem, sendo que a maioria dos sinais de pontuações e pingos das letras i e j não são contabilizados como palavras. Também é possível observar que há retângulos em torno de alguns elementos que a princípio não constituem o texto principal, como é o caso do número da página no canto superior esquerdo da imagem, mas há uma grande dificuldade de eliminar esses elementos da contagem, uma vez que, dado os critérios de classificação de elementos textuais, dificilmente esses números isolados não seriam classificados como texto.

Por fim, o programa resultou numa contagem total de 242 palavras no texto.

4 Conclusão

Os operadores morfológicos, tais como dilatação e erosão, têm relevantes aplicações práticas no processamento de imagens digitais. Os procedimentos realizados nesse trabalho para a análise de textos são grandes exemplos dessa utilidade.

Ainda assim, o uso desses operadores vem acompanhado de diversos desafios para que se tornem efetivos. Por exemplo, caso a imagem estivesse rotacionada, de forma que as linhas não estivessem alinhadas horizontalmente, haveria a necessidade de realizar um tratamento desse desalinhamento. Além disso, como comentado ao longo do trabalho, há diversas inconsistências que podem dificultar o funcionamento preciso de métodos de contagem de linhas ou palavras com o uso de operadores morfológicos.

Há também características específicas do textos que podem influenciar nesse processo, como a quantidade de colunas, o espaçamento entre palavras, entre linhas e até mesmo entre letras. A própria fonte da letra pode alterar o resultado esperado, uma vez que altera a proporção de pixels de cada cor e o formato do texto. Assim, o programa desenvolvido, ainda que funcione satisfatoriamente para a imagem apresentada, talvez tivesse dificuldades em se mostrar efetivo caso essas características do texto fossem alteradas. Porém, para o caso da imagem analisada no trabalho, embora o método desenvolvido não seja perfeito, ele é capaz de fornecer uma boa estimativa tanto para o número de linhas (resultando em 34 linhas), quanto para o número de palavras (242) do texto.