MO443 - Processamento de Imagem Digital
Trabalho 04

Patrick de Carvalho Tavares Rezende Ferreira - 175480

26 de junho de 2020

1 Introdução

O objetivo deste trabalho é aplicar operadores morfológicos para segmentar regiões que compreendem "texto" e "não texto" em uma dada imagem de entrada no formato "pbm" de bitmap.

2 Descrição do código elaborado

O código desenvolvido neste relatório é contido no arquivo "main.py" e foi desenvolvido para que o utilizador possa simplesmente importar todas as suas funções e utilizá-las independentemente. Para fins de demonstração, o código também possui um script dentro da função "main()" que é executado sempre que chamado como arquivo principal pelo interpretador python e cria um diretório "output" contendo as imagens solicitadas neste roteiro. Todas as funções com argumentos numéricos aceitam que estes sejam inteiros positivos, tratando de forma adequada e convertendo para o tipo "unit8" a matriz da imagem antes de salvar o arquivo de saída. O diretório das imagens de entrada é passado também como argumento de função.

As funções criadas para a execução deste roteiro são descritas nas subseções a seguir e podem ser importadas conforme a figura 1. As demais funções foram importas diretamente da biblioteca do OpenCV [1].

2.1 Função draw-bounding-boxes

A função "draw-bounding-boxes" recebe como argumento uma imagem a serem inscritas as marcações ao redor de cada componente conexo e um array de contornos gerado no padrão da função "findContours" do OpenCV. O resultado é uma imagem com as mesmas dimensões da original e componentes conexos identificados com uma numeração para poderos nos referir a eles. Descrição na tabela 1.

2.2 Função draw-text-non-text

A função "draw-text-non-text" é similar à função "draw-bounding-boxes", porém adicionando rótulos ao lado de cada componente conexo marcado, indicando se ele é reconhecido como texto ou não. Descrição na tabela 2.

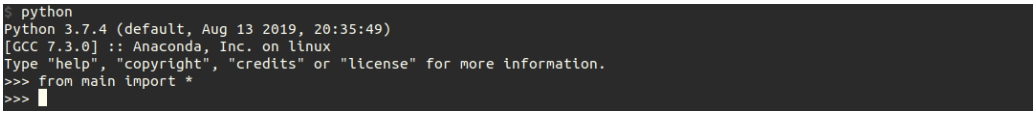

Figura 1: Comando para importar as funções utilizadas para o roteiro.

Tabela 1: Referência da função draw-bounding-boxes.

Parâmetro	Descrição
image	Imagem original com componentes conexos gerados através de operações morfológicas.
contornos	Array de contornos gerados no padrão da função "findContours" do OpenCV.
retorno	Imagem de entrada adicionada de marcações no entorno de cada componente conexo.

2.3 Função draw-only-text

A função "draw-only-text" é similar à função "draw-bounding-boxes", porém contornando apenas os componentes conexos reconhecidos como texto. Descrição na tabela 3.

2.4 Função get-ratios

A função "get-ratios" recebe como argumento uma imagem e um array de contornos gerado no padrão da função "findContours" do OpenCV, indicando o contorno de cada componente conexo. Ela calcula a proporção de pixels pretos e pixels totais em cada componente conexo contornado por cada retângulo, bem como a proporção entre o número de transições verticais e horizontais de branco para preto em cada componente conexo, retornando duas listas com as taxas calculadas. Cada elemento de cada uma destas listas é relativo à razão calculada para cada elemento do array de contornos passado como argumento (em ordem), e a ordem pode ser conferida nos índices gerados pela função "draw-bounding-boxes" (que gerou a figura 9). Descrição na tabela 4.

3 Execução do script demonstrativo e tarefas do roteiro

A geração das saídas solicitadas por este roteiro pode ser realizada disponibilizando-se a imagem "bitmap.pbm" no mesmo diretório que o script "main.py" e utilizando-se do comando exibido na figura 2. As imagens são geradas dentro do diretório "output".

3.1 Item 1

O item 1 do roteiro pede apenas que façamos a dilatação da imagem original com um elemento estruturante de 1 pixel de altura e 100 pixels de largura, a qual é realizada com a função "dilate" da biblioteca do OpenCV. O resultado é exibido na figura 3 e dá a impressão que todas as regiões onde haviam pixels foram dilatadas horizontalmente, como era previsto.

Tabela 2: Referência da função draw-text-non-text.

Parâmetro	Descrição
image	Imagem original com componentes conexos gerados através de operações morfológicas.
contornos	Array de contornos gerados no padrão da função "findContours" do OpenCV.
text-line-indexes	Array contendo índices dos elementos conexos identificados como texto.
retorno	Imagem de entrada adicionada de marcações no entorno de cada componente conexo.

Tabela 3: Referência da função draw-only-text.

Parâmetro	Descrição
image	Imagem original com componentes conexos gerados através de operações morfológicas.
contornos	Array de contornos gerados no padrão da função "findContours" do OpenCV.
text-line-indexes	Array contendo índices dos elementos conexos identificados como texto.
retorno	Imagem de entrada adicionada de marcações no entorno de cada componente conexo.

3.2 Item 2

O item 2 do roteiro pede que façamos a erosão da imagem resultante do item 1 com um elemento estruturante de 1 pixel de altura e 100 pixels de largura, a qual é realizada com a função "erode" da biblioteca do OpenCV. O resultado é exibido na figura 4 e mostra que a imagem foi "afinada" preservando os detalhes que possuía antes da operação, mas não desfazendo a dilatação inicial, o que é coerente com o fato de que dilatação e erosão não são operações exatamente inversas.

3.3 Item 3

O item 3 do roteiro pede apenas que façamos a dilatação da imagem original com um elemento estruturante de 200 pixels de altura e 1 pixel de largura, a qual é realizada com a função "dilate" da biblioteca do OpenCV. O resultado é exibido na figura 3 e dá a impressão que todas as regiões onde haviam pixels foram dilatadas verticalmente, como era previsto.

3.4 Item 4

O item 4 do roteiro pede que façamos a erosão da imagem resultante do item 2 com um elemento estruturante de 200 pixels de altura e 1 pixel de largura, a qual é realizada com a função "erode" da biblioteca do OpenCV. O resultado é exibido na figura 4 e mostra que a imagem foi "encurtada" preservando os detalhes que possuía antes da operação, mas não desfazendo a dilatação inicial, o que é coerente com o fato de que dilatação e erosão não são operações exatamente inversas.

3.5 Item 5

O item 5 pede que realizemos a operação AND entre cada um dos pixels das imagens resultantes dos itens 2 e 4, o que claramente resulta na interseção das áreas preenchidas das imagens (que são binárias).

Tabela 4: Referência da função get-ratios.

Parâmetro	Descrição
image	Imagem original com componentes conexos gerados através de operações morfológicas.
contornos	Array de contornos gerados no padrão da função "findContours" do OpenCV.
retorno	2 listas contendo as taxas de pixels pretos e de transição de cada componente.

```
$ python main.py
```

Figura 2: Comando para gerar as saídas requisitadas pelo roteiro.

Estas imagens são resultado de uma operação de fechamento (dilatação seguida de erosão com mesmo elemento estruturante) horizontal e vertical, respectivamente ao item 2 e 4. Assim, a imagem resultante (figura 7) é basicamente um fechamento da imagem original sem pontos discrepantes saindo de cada componente resultante (eliminados durante a interseção AND).

3.6 Item 6

No item 6, aplicamos um fechamento diretamente com a função "morphologyEx" do OpenCV, utilizando um elemento estruturante de 1 pixel de altura por 30 pixels de largura. O resultado é que os componentes conexos originários de cada palavra se juntam formando um único componente por frase, como mostra a figura 8.

3.7 Item 7

Utilizando a função de "connectedComponents" junta à "findContours", ambas do pacote OpenCV, aplicamos o algoritmo já implementado de maneira eficiente para encontrar componentes conexos, conforme definidos nos slides da disciplina. São encontrados um total de 52 componentes conexos (figura 9) e nós marcamos seus contornos com retângulos, usando as funções "boundingRect" e "Rectangle", também do OpenCV. Cada componente possui um índice ao lado esquerdo, o qual foi colocado para indicar a ordem em que eles foram mapeados.

3.8 Item 8

No item 8 é solicitado apenas que calculemos a proporção entre pixels pretos e pixels totais em cada *bounding box* (retângulo), e também a relação entre a quantidade de transições verticais e horizontais, em relação à quantidade de pixels pretos. O resultado é calculado e salvo em duas listas, mas não será exibido aqui, devido à quantidade de saídas. Esta saída é gerada em um arquivo de texto ao se executar o script fornecido.

Para calcular a proporção de pixels pretos, utilizou-se a função "numpy.where" comparando-se os pixels com valores verdadeiros e falsos e contando-se o comprimento da saída da função, assim evita-se o loop para este caso. Para contar a quantidade de pixels totais, necessitou-se apenas multiplicar a altura em pixels pela largura em pixels de cada retângulo, já fornecida na saída do OpenCV. Para calcular a quantidade de transições na imagem, foi necessário comparar cada array

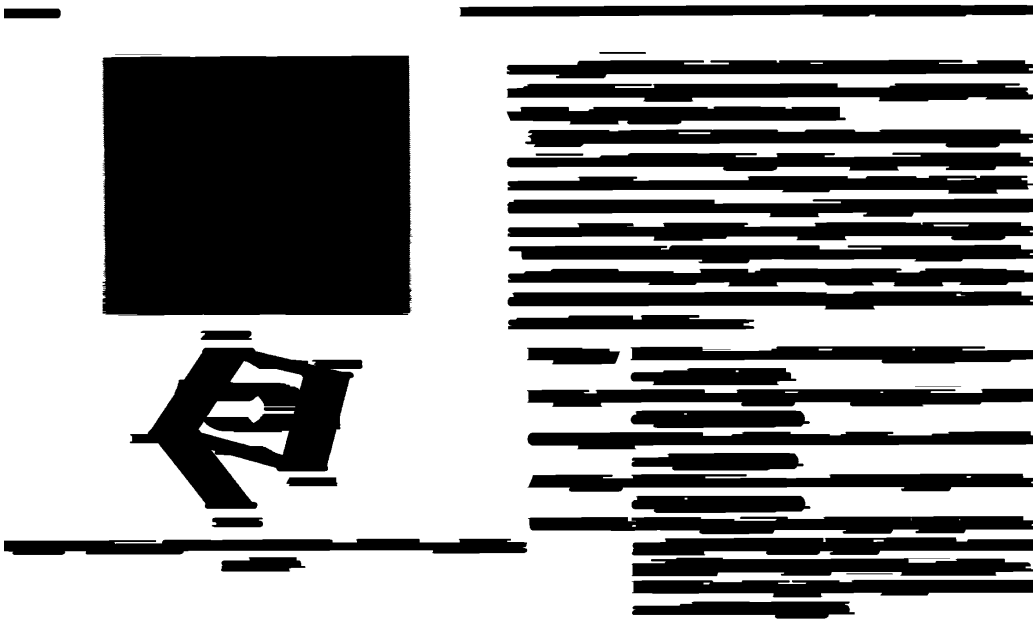

Figura 3: Imagem de saída para o item 1.

representando o retângulo envolvendo um dado componente conexo com o próprio retângulo deslocado em 1 elemento (horizontalmente e depois veticalmente), aplicando-se a função "numpy.where" para identificar sempre que o array elemento anterior fosse branco e o atual preto. Por último, calculou-se as frações pedidas para cada item.

3.9 Item 9

Analisando quais componentes encontrados na imagem resultaram em cada componente conexo indicado na figura 9, elaborou-se uma regra para classificá-los em "texto" e "não-texto". Quatro condições deveriam ser atendidas para o componente ser classificado como texto: A proporção de pixels pretos deveria ser maior que 0.3, menor que 0.98, maior que a proporção de transições e a própria proporção de transições em relação aos pixels pretos deveria ser menor que 0.4. Caso qualquer uma destas regras fosse descumprida, o componente era classificado como não texto. A figura 10 mostra como cada componente conexo foi classificado sob esta regra e a figura 11 mostra apenas as regiões identificadas como texto sobre a figura original sendo tratada.

Nota-se que o classificador obteve um bom desempenho, identificando quase todas as frases indicadas pelos componentes conexos, falhando apenas em identificar a numeração da página (canto superior esquerdo da fotografia) e o número "1" da figura presente na foto, apesar de diferentes configurações terem sido testadas. Por serem números e com poucos algarismos, provavelmente obtiveram uma proporção de transições alta para serem classificados como texto junto aos demais componentes conexos assim identificados.

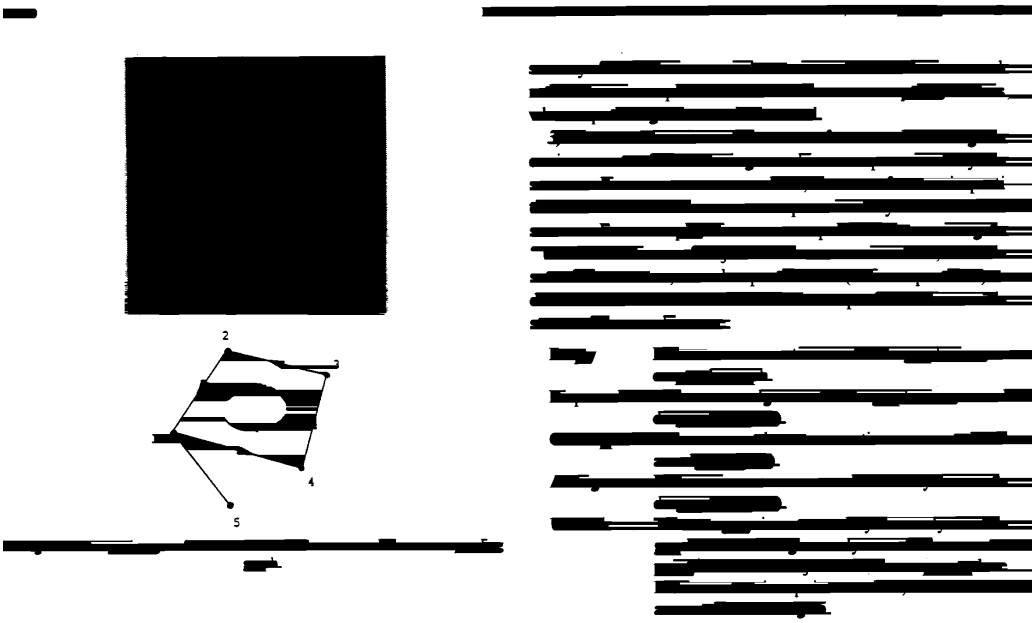

Figura 4: Imagem de saída para o item 2.

3.10 Item 10

O item 10 solicita que elaboremos operadores morfológicos adequados para identificação e contagem de palavras no texto. Para tal, realizou-se a dilatação da imagem original com um elemento estruturante de 6 pixels de altura por 2 de largura, seguida de uma operação de fechamento aplicada 3 vezes com um elemento estruturante de 6 pixels de altura por 4 de largura. O objetivo desta operação era fechar os espaços entre as letras de cada palavra, principalmente para cima, mas sem permitir que o componente conexo resultante de cada palavra encostasse nos vizinhos. Para garantir que os componentes conexos não tocassem nos vizinhos de cima ou debaixo, realizamos a operação AND (interseção) com a imagem resultante do item 6, que já possui um elemento bem estruturado ao longo das frases inteiras e terá os componentes conexos de cada palavra separados ao realizar a operação AND com a imagem gerada neste passo. As palavras encontradas são exibidas na figura 12, tendo uma taxa de acerto alta, falhando apenas em identificar novamente o número "1" da figura inserida na fotografia e o número 5, identificando porém o número da página "310" desta vez, com o uso deste novo operador morfológico proposto.

As saídas que foram são exibidas na tela ao longo da execução do script indicam que reconheceram 52 componentes conexos, no item 7, sendo 42 identificados como texto. Após a aplicação das operações morfológicas propostas neste item, o script retornou a identificação de 275 blocos de palavras, que podem ser verificados na figura 12.

Figura 5: Imagem de saída para o item 3.

4 Conclusão

Este roteiro mostrou a natureza e aplicação das operações morfológicas. Fica claro que, através destes operadores, é possível fazer a identificação de diversos padrões presentes em imagens, evitando-se um esforço manual que seria repetitivo e grande demais, e possibilitando-se uma automatização do processo.

Este roteiro consistiu apenas em identificar onde existem palavras ou frases na imagem, mas com algum desenvolvimento a mais é possível identificar também o conteúdo das mesmas.

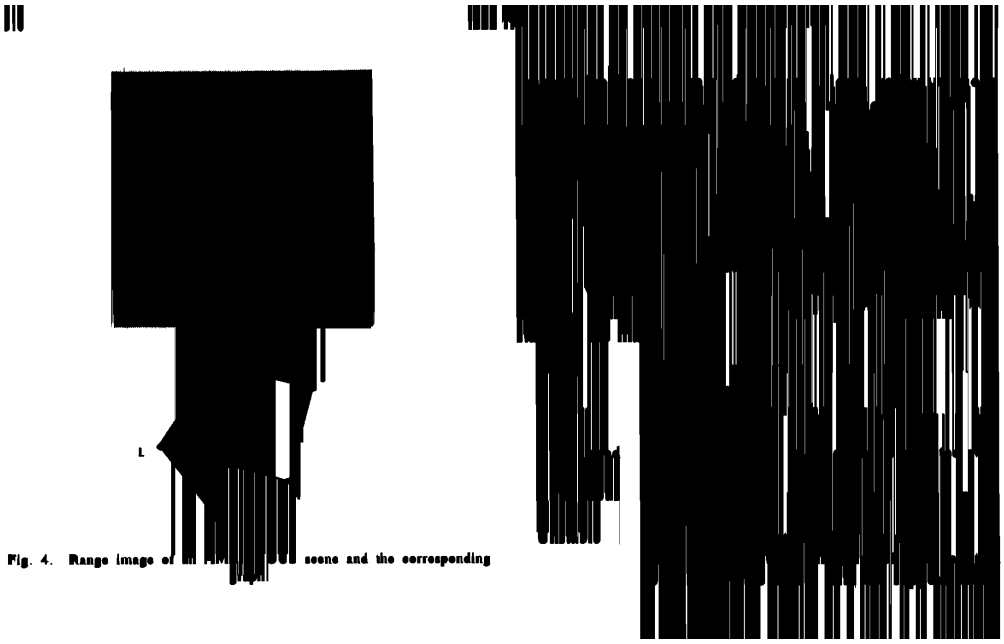

Fig. 4. Range image of [redacted] scene and the corresponding

Figura 6: Imagem de saída para o item 4.

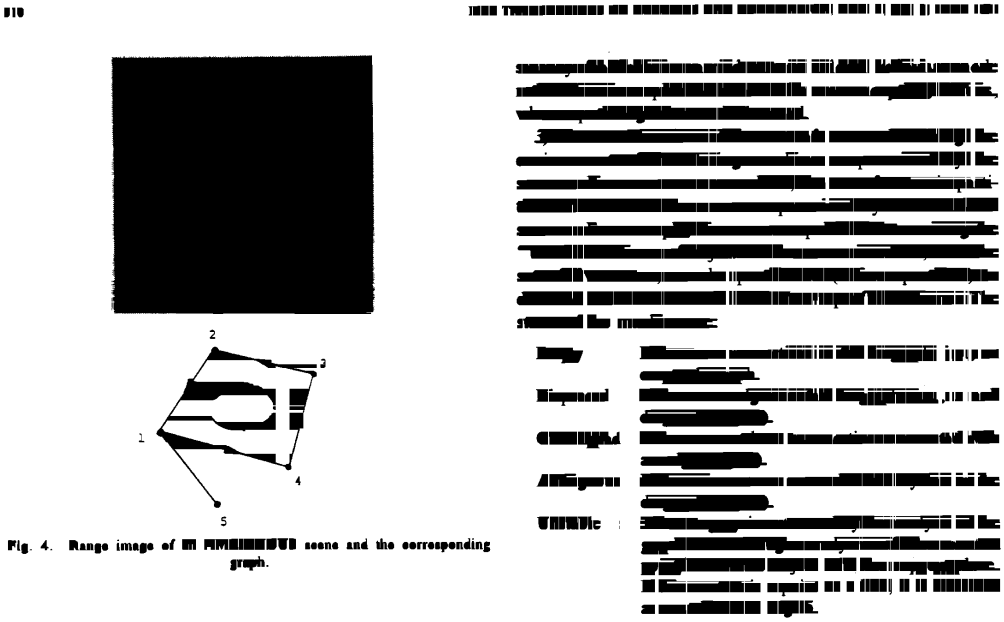

Fig. 4. Range image of [redacted] scene and the corresponding graph.

Figura 7: Imagem de saída para o item 5.

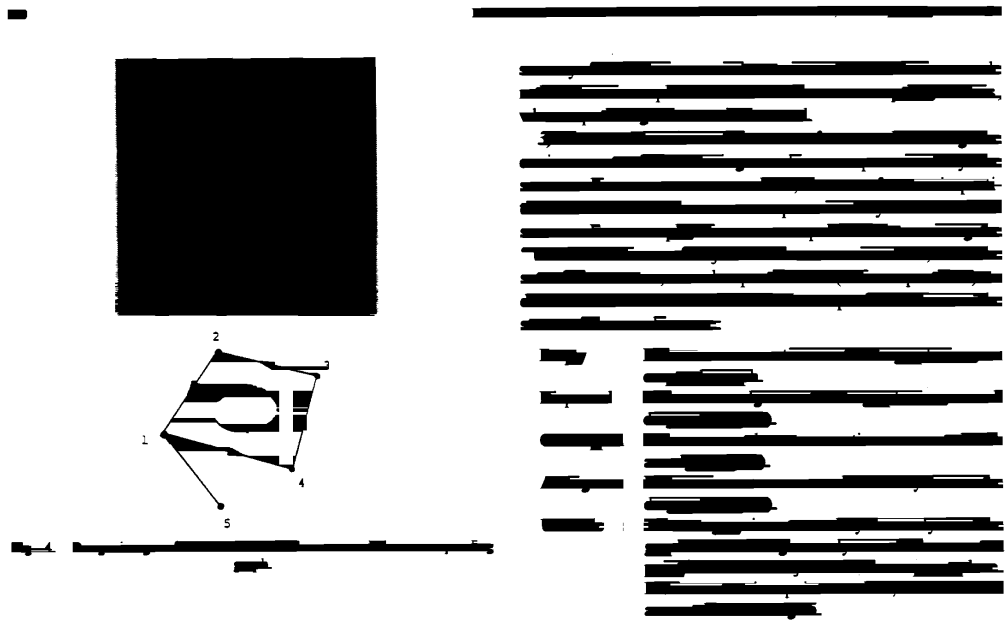

Figura 8: Imagem de saída para o item 6.

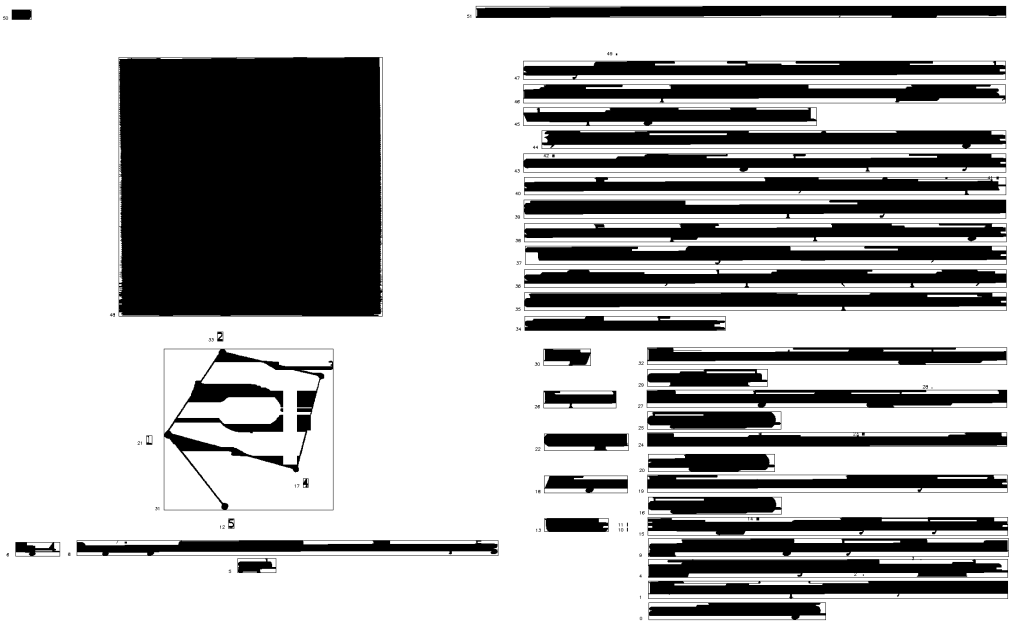

Figura 9: Imagem de saída para o item 7.

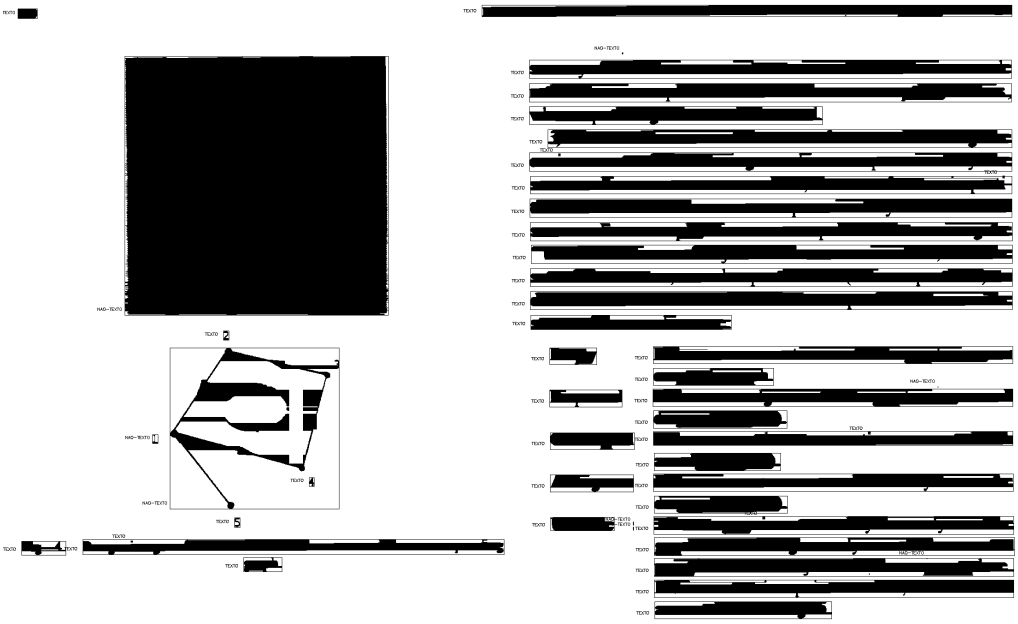

Figura 10: Imagem de saída para o item 9 (parte 1).

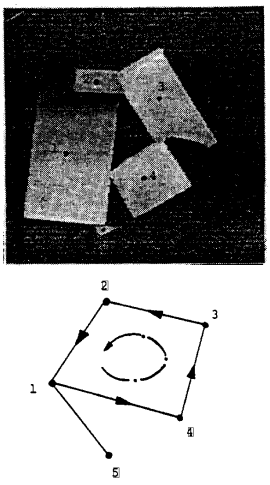

Fig. 4. Range image of an AMBIGUOUS scene and the corresponding graph.

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 11: Imagem de saída para o item 9 (parte 2).

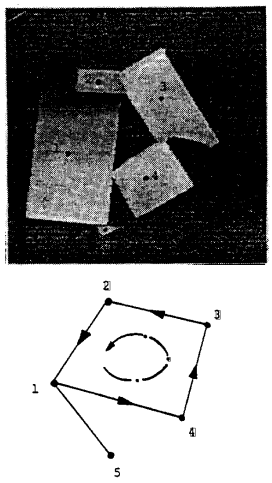

Fig. 4. Range image of an AMBIGUOUS scene and the corresponding graph.

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 12: Imagem de saída para o item 10.

Referências

[1] G. Bradski. “The OpenCV Library”. Em: *Dr. Dobb's Journal of Software Tools* (2000).