

Relatório do Trabalho 3

Raissa Cavalcante Correia - RA: 150619

Disciplina: MC920 - 1s2019 - Turma A

22 de Maio de 2019

Introdução

Foi utilizado a mais nova versão do CV o CV 4(atualmente o 4.1) para facilitar lidar com o formato .pbm tanto na entrada como na saída. Como pode ser observado no vídeo da referência [1] ou através da linha: `"pip3 install opencv-python"`.

Desde que o pip3, o python3, e o cv3 esteja atualizados ou em versões relativamente recentes, conforme pode ser visto no vídeo.

É necessário, tanto na hora do input quanto no output passar toda imagem pelo operador NOT Bitwise pois o código em C fornecido inverte a relação entre preto e branco com os bits 0 e 1.

```
img = cv.imread('bitmap.pbm',0)
img = cv.bitwise_not(img)
```

Em seguida foram criadas uma série de np arrays únicos para que não haja interferência com as operações entre eles para que eles possam receber o resultado das operação e erosão, dilatação e fechamento do CV4.

Os elementos estruturantes foram criados utilizado o método "ones" do numpy, para criar uma matriz de "1"s dado sua altura e largura, sendo assim a forma mais prática e eficaz.

```
structuring = np.ones((1,100), np.uint8)
```

Erosão

Aqui temos a erosão e a criação da imagem de output para os itens 2 e 4.

O método de erosão do cv4 tem a sintaxe de Imagem, Kernel, e número de iterações que por default é igual a 1. Conforme dito na referência[2]:

“ Um pixel da imagem original (either 1 or 0) será considerado 1 somente se todos os pixel sobre o kernel for 1, caso contrário será erodido(transformado em 0)”

```
imgAfter2 = cv.erode(imgAfter, structuring)
cv.imwrite('./imgAfter2.pbm', cv.bitwise_not(imgAfter2))

imgAfter4 = cv.erode(imgAfter3, structuring_3)
cv.imwrite('./imgAfter4.pbm', cv.bitwise_not(imgAfter4))
```

Dilatação

Aqui temos a dilatação e a criação da imagem de output para os itens 1 e 3.

O método de dilatação do cv4 tem a sintaxe de Imagem, Kernel, e número de iterações que por default é igual a 1. Conforme dito na referência[2]:

“Um pixel é 1 se pelo menos um pixel sobre o kernel é igual a 1”

```
imgAfter = cv.dilate(img,structuring)
cv.imwrite('./imgAfter1.pbm', cv.bitwise_not(imgAfter))

structuring_3 = np.ones((200,1), np.uint8)
imgAfter3 = cv.dilate(img,structuring_3)
cv.imwrite('./imgAfter3.pbm', cv.bitwise_not(imgAfter3))
```

Fechamento e Bitwise AND

Para os itens 5 e 6, criamos um novo kernel(1x30) realizamos um and bit a bit das imagens após o item 2 e o item 4[4]. Para o item 6 fizemos a operação de fechamento que descrita na documentação do CV4[2]:

“Fechamento é o reverso da Abertura, Dilatação seguida de erosão. É útil para fechar pequenos pontos nos objetos”

```
imgAfter5 = cv.bitwise_and(imgAfter2, imgAfter4)
cv.imwrite('./imgAfter5.pbm', cv.bitwise_not(imgAfter5))

structuring_6 = np.ones((1,30), np.uint8)
imgAfter6 = cv.morphologyEx(imgAfter5, cv.MORPH_CLOSE, structuring_6)
cv.imwrite('./imgAfter6.pbm', cv.bitwise_not(imgAfter6))
```

Componentes Conexas

Por fim utilizamos a biblioteca de subprocessos do python3 para:

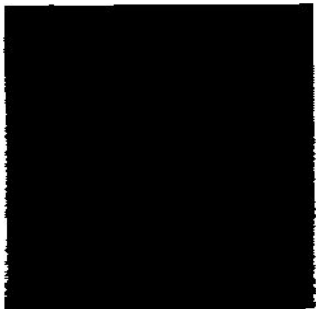
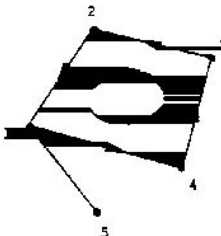
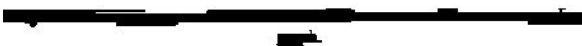
- 1) Compilar o código C através do GCC com o uso da flag -lm cuja necessidade é explicada na referência[3] passando para a shell como faz a função call.
- 2) Execução do código passando 3 parâmetros para a shell o binário comp_conexos, o arquivo de entrada imgAfter6 e o arquivo de saída imgAfter7.

Com isso temos o resultado da resolução da imagem: 2233 x 1374 pixels

O número de componentes conexas: 53

E claro a imagem resultante com os retângulos destacados.

```
subprocess.call(['gcc', '-o', 'comp_conexos', 'comp_conexos.c', '-lm'])
subprocess.call(['./comp_conexos', './imgAfter6.pbm', './imgAfter7.pbm'])
```

[illegible]





Após o item 3



Fig. 4. Range image of an RGB-D scene and the corresponding

Após item 4

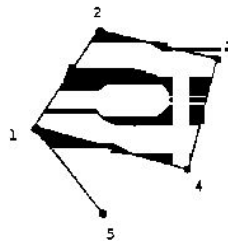


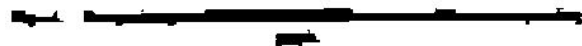
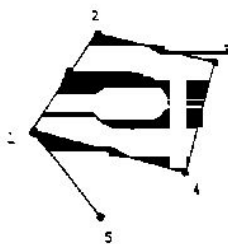
Fig. 4. Range image of the PPM range sensor and the corresponding graph.

...the range image of the PPM range sensor is shown in Fig. 4. The range image is a square image with a side length of 1024 pixels. The range image is a grayscale image where the intensity of each pixel represents the range of the sensor at that point.

The range image is a square image with a side length of 1024 pixels. The range image is a grayscale image where the intensity of each pixel represents the range of the sensor at that point. The range image is a square image with a side length of 1024 pixels. The range image is a grayscale image where the intensity of each pixel represents the range of the sensor at that point.

The range image is a square image with a side length of 1024 pixels. The range image is a grayscale image where the intensity of each pixel represents the range of the sensor at that point. The range image is a square image with a side length of 1024 pixels. The range image is a grayscale image where the intensity of each pixel represents the range of the sensor at that point.

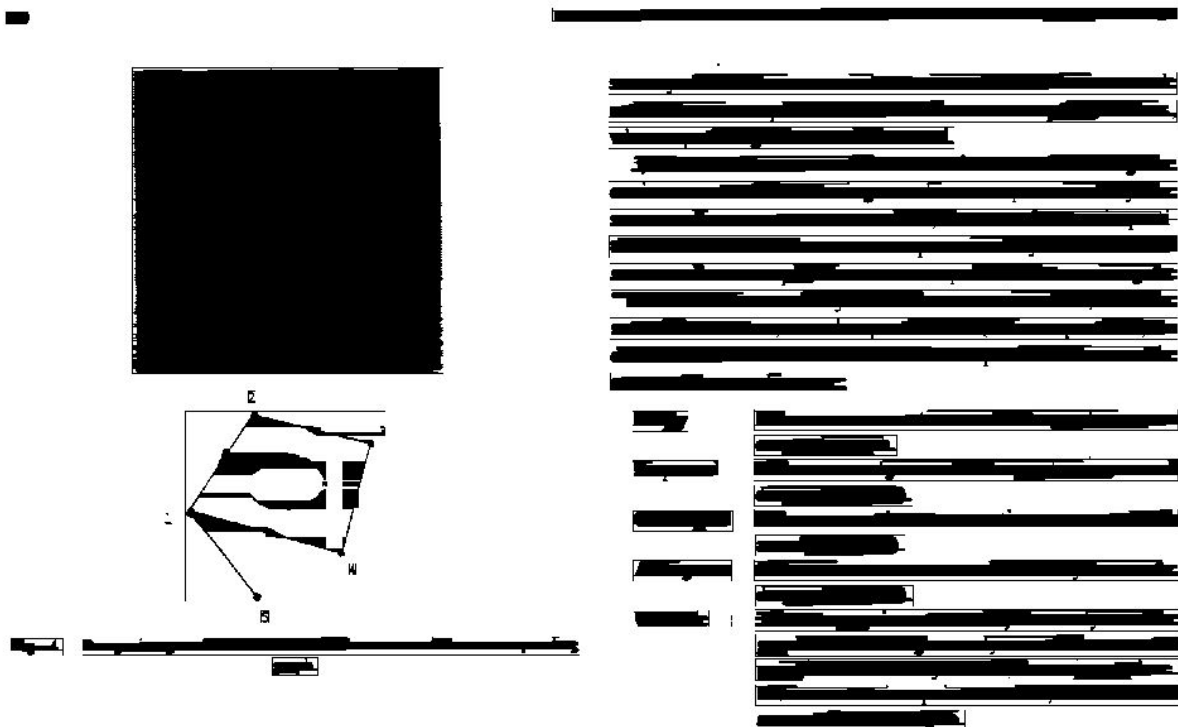
Após item 5



...the range image of the PPM range sensor is shown in Fig. 4. The range image is a square image with a side length of 1024 pixels. The range image is a grayscale image where the intensity of each pixel represents the range of the sensor at that point.

The range image is a square image with a side length of 1024 pixels. The range image is a grayscale image where the intensity of each pixel represents the range of the sensor at that point. The range image is a square image with a side length of 1024 pixels. The range image is a grayscale image where the intensity of each pixel represents the range of the sensor at that point.

Após item 6



Após item 7

Conclusões

Os resultados foram de acordo com o previsto pela literatura e consistentes até o item 7. Devido a questões com o prazo e outras dificuldades os itens de 8 a 10 não foram realizados, infelizmente.

Referências

[1] **Instalação CV4:** <https://www.youtube.com/watch?v=cGmGOi2kkJ4>

[2] **Erosão, Dilatação e Fechamento CV4:**
https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html

[3] **O uso do “-lm” ao compilar o programa em:**
<https://stackoverflow.com/questions/44175151/what-is-the-meaning-of-lm-in-gcc>

[4] **Operações Bitwise CV4:**
https://docs.opencv.org/4.1.0/d0/d86/tutorial_py_image_arithmetics.html