Trabalho 3

MO443/MC920 - Introdução ao Processamento de Imagem Digital

Lucas de Oliveira Silva - 220715

1 Introdução

Neste trabalho o objetivo foi aplicar métodos de limiarização global e local em imagens monocromáticas. As abordagens locais foram feitas com auxílio da função *generic_filter* do scipy. A solução foi feita com Python 3.8.5 em um único arquivo .py. Foram utilizadas as seguintes bibliotecas: numpy (1.19.1), scikit-image (0.16.2), scipy (1.5.2) e opencv-python (4.4.0.46).

2 Como executar

A implementação está no *script* trabalho3.py, as imagens de teste na raiz e as saídas obtidas na pasta output. O programa precisa necessariamente de uma imagem de entrada (-i img) em formato pgm e um padrão para as saídas (-o padrão). Os resultados são salvos em formato png seguindo a forma "padrão-método.png". O resto dos parâmetros são opcionais e possuem valores padrões. Os parâmetros opcionais são os seguintes:

- -n O tamanho da máscara usada em cada método local. O padrão é 5.
- -f Os resultados binários são multiplicados por f para melhorar o contraste. O padrão é 255.

Ex: python3 ./trabalho3.py -i sonnet.pgm -o ./output/sonnet3 -n 3

Para mais detalhes é possível executar o *script* com o parâmetro -h.

A imagem é lida de forma monocromática usando a função *cv2.imread(..., cv2.IMREAD_GRAYSCALE)* do opencv, e logo depois os valores são convertidos para float. Ao executar os métodos locais o tratamento de bordas é feito usando o modo *nearest* da função *generic_filter* do scipy, onde o valor atribuído é aquele mais próximo da posição a ser considerada. Antes de

salvar a imagem em arquivo é feito uma conversão direta para uint8, já que o resultado de cada filtro já terá valores inteiros no intervalo [0, 255].

Ao executar, todos os métodos são aplicados a mesma imagem para reaproveitar o cálculo de valores como μ e σ. Os parâmetros dos métodos que estão no código foram considerados os mais apropriados, mas para alterar basta colocar um novo valor na função desejada.

3 Avaliação dos resultados

Nesta seção será avaliado o resultado de cada método. Foi testado, nos métodos locais, vizinhanças dos seguintes tamanhos: 3, 5, 15, 25, 35 e 70. Já o método global foi aplicado com o limiar de 128.

Em geral os resultados foram melhores em imagens com pouca textura e iluminação homogênea, ou seja, aquelas que possuem dois picos bem definidos em seus histogramas (Figura 1). A imagem monarch que possui variações bruscas no objeto (asa da borboleta) teve sua saída segmentada em várias partes. Já os objetos mais definidos do fiducial foram os que geraram saídas mais promissoras.

- Método Global: É muito sensível a iluminação da imagem, o resultado não fica bom quando a iluminação varia como nas entradas fiducial, sonnet e wedge. Além disso, imagens como a baboon que possuem muita textura resultam em objetos com bastante ruído e desconexos.

- Método de Bernsen: Não é tão sensível a iluminação quanto o método global, mas as saídas ainda possuem muito ruído. Esse ruído pode ser reduzido aumentando a vizinhança considerada, além disso as bordas são fortemente marcadas. Com vizinhanças pequenas (abaixo de 10) o fundo fica repleto de artefatos. Esse método foi mais eficaz na imagem retina pois conseguiu separar com relativa clareza a região de interesse.

- Método de Niblack: Obteve um resultado muito próximo ao método de Bernsen e com basicamente as mesmas características. O resultado foi quase perfeito na imagem fiducial usando uma vizinhança de 70, onde mesmo com iluminação variada os objetos foram destacados corretamente.

- Método de Sauvola e Pietaksinen: Esse método produziu imagens com altas porcentagens de pixels preto, produzindo até algumas saídas sem pixels brancos. O resultado foi bom na imagem fiducial e nas com bastante variação de textura como o baboon, mas o resultado não foi bom na imagem monarch em que todo o fundo foi considerado como objeto. O resultado foi ainda pior nas imagem com textos e objetos geométricos como o wedge.

- Método de Phansalskar, More e Sabale: Esse método obteve resultados satisfatórios com a maioria das imagens, mas como os autores sugerem esse método se sai melhor segmentando imagens de microscópio.

- Método do Contraste: O resultado é o mesmo que o método de Bernsen só que com os valores invertidos. Isso porque se um pixel possui intensidade maior que o limiar de Bernsen significa que este será classificado como objeto, além disso ele se aproxima mais do máximo local. Só que o método do contraste atribui como fundo valores mais próximos do máximo local, e portando um é a negação do outro.

- Método da Média: Saída muito parecida com o método de Bernsen, só que a saída é mais sensível a ruídos na imagem de entrada e portando produz objetos mais borrados.

- Método da Mediana: Resultado semelhante ao anterior mas ainda mais borrado e sensível a ruídos, além disso a segmentação passa a piorar em imagens com iluminação variada.

Alguns exemplos são ilustrados nas figuras a seguir, além do histograma de cada imagem de entrada. Alguns métodos foram melhores que outros em um dado tipo de imagem, já que os métodos locais são extremamente custosos computacionalmente é possível utilizar apenas o método global em alguns casos mesmo que a saída não seja tão boa. Com relação ao tamanho das vizinhanças é possível notar que a maioria dos métodos precisam que ela seja ao menos 15 para obter um resultado satisfatório, portanto métodos como o da mediana se tornam quase impraticáveis em aplicações de tempo real, dado o alto número de operações necessárias para encontrar a mediana de muitos números.

Alguns métodos, como o de Bernsen, poderiam se beneficiar de uma implementação mais específica em GPU, onde seria possível considerar a intersecção de vizinhanças entre os pixels da imagem.

4 Conclusão

Cada método produziu um bom resultado para um tipo de imagem, os métodos locais sofrem pouco com a mudança na iluminação só que o método global não foi capaz de lidar com isso. Apesar dos resultados pouco promissores, o método global ainda pode ser usado no pré processamento de imagens em tempo real já que é mais eficiente. Dado que valor da imagem pode ser computado paralelamente uma possível implementação em GPU poderia acelerar drasticamente a aplicação. considerando a simplicidade dos algoritmos no geral os resultados foram bem satisfatórios.

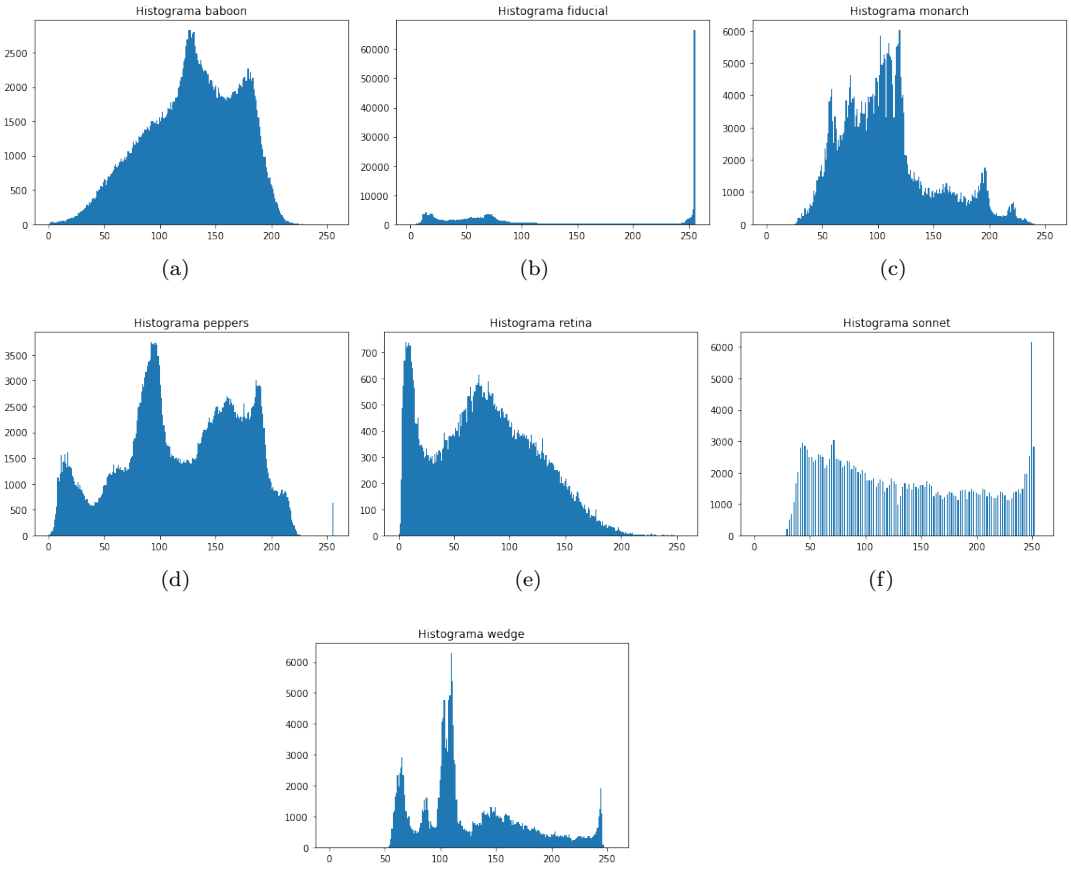
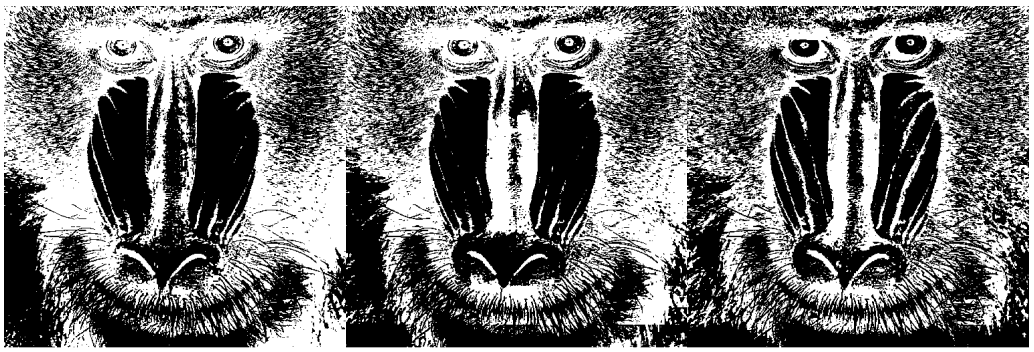
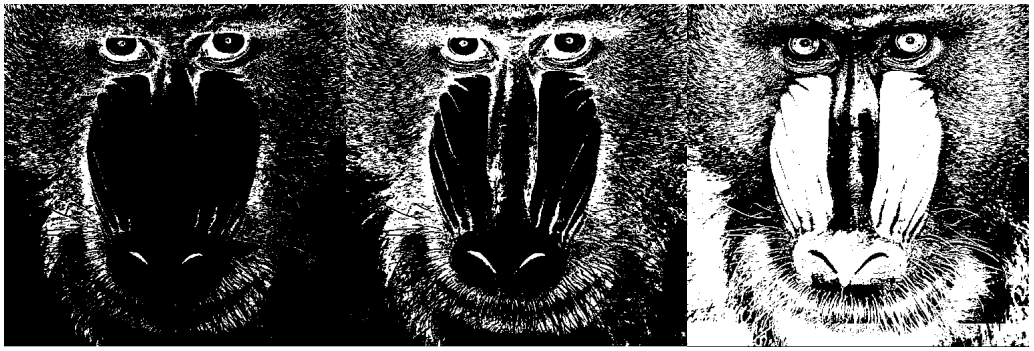


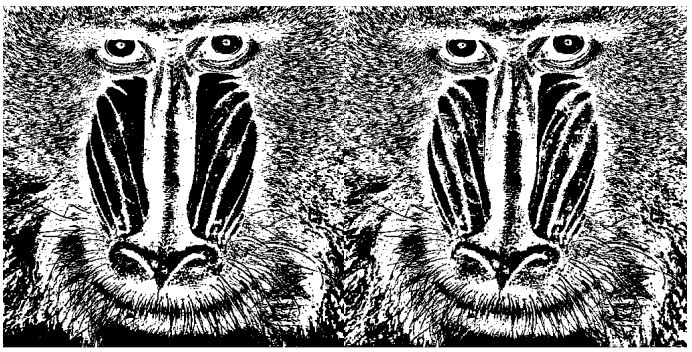
Figura 1: Histograma de cada imagem de entrada



(a) Global 52,81% de preto (b) Bernsen 56,52% de preto (c) Niblack 61,02% de preto

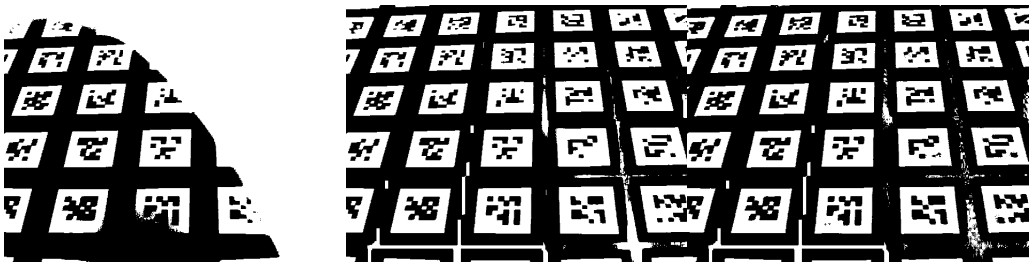


(d) Sauvola 89,70% de preto (e) More 76,31% de preto (f) Contraste 43,03% de preto

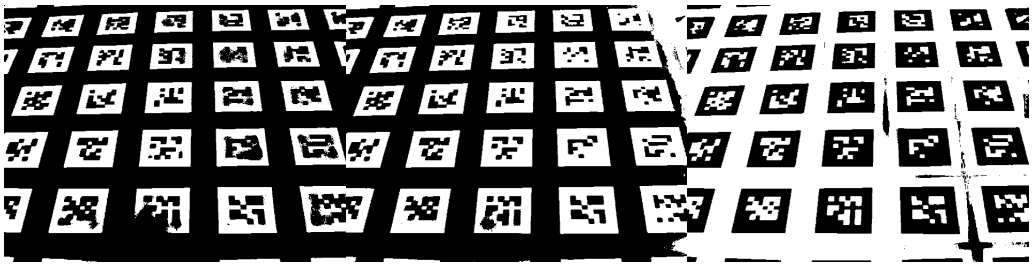


(g) Média 53,78% de preto (h) Mediana 50,60% de preto

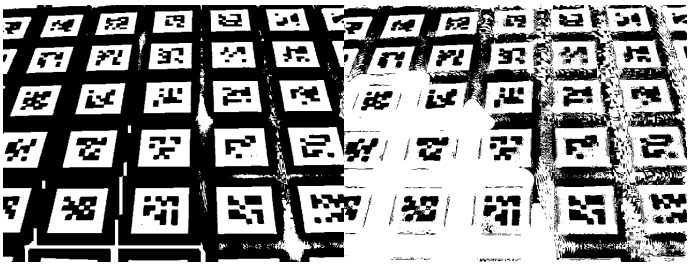
Figura 2: Cada método, usando vizinhança 70, aplicado a uma imagem com muita textura (baboon)



(a) Global 38,02% de preto (b) Bernsen 61,90% de preto (c) Niblack 64,21% de preto

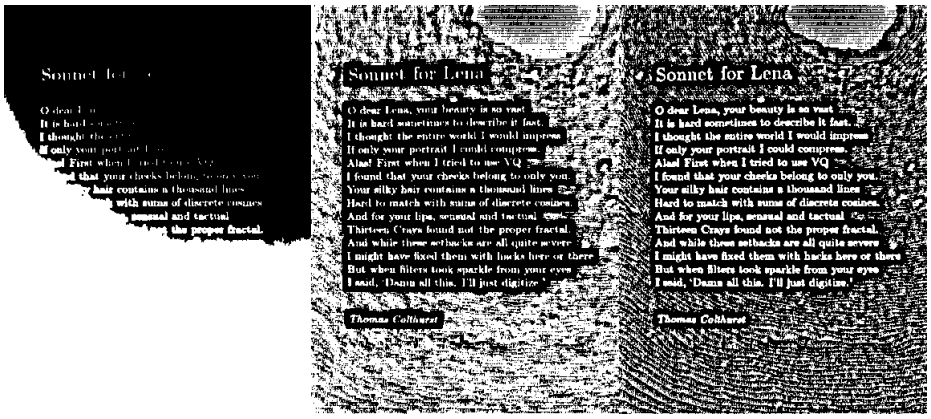


(d) Sauvola 70,41% de preto (e) More 62,90% de preto (f) Contraste 36,51% de preto

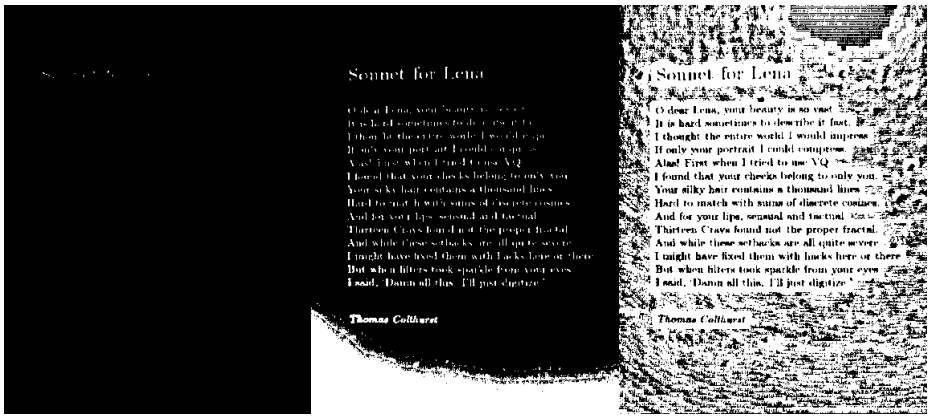


(g) Média 61,78% de preto (h) Mediana 33,33% de preto

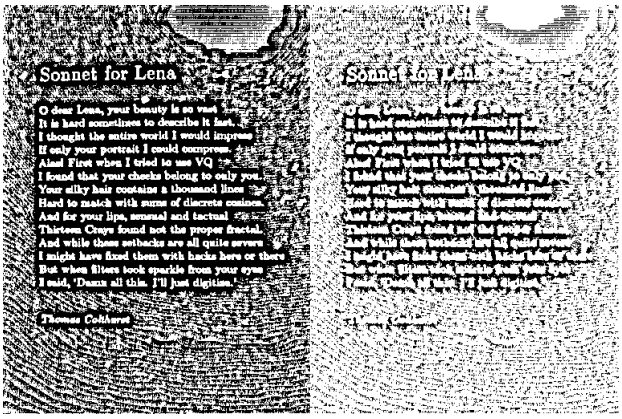
Figura 3: Cada método, usando vizinhança 35, aplicado a uma imagem com iluminação variada (fiducial)



(a) Global 47,94% de preto (b) Bernsen 54,06% de preto (c) Niblack 67,97% de preto

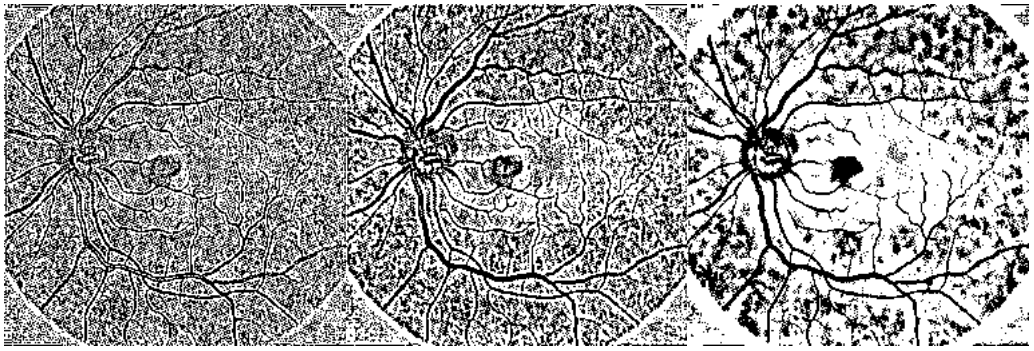


(d) Sauvola 99,99% de preto (e) More 85,28% de preto (f) Contraste 26,98% de preto



(g) Média 57,21% de preto (h) Mediana 26,58% de preto

Figura 4: Cada método, usando vizinhança 15, aplicado a uma imagem com texto (sonnet)



(a) Vizinhança 3 com 45,36% preto (b) Vizinhança 5 com 40,66% preto (c) Vizinhança 15 com 30,37% preto

(d) Vizinhança 25 com 27,33% preto (e) Vizinhança 35 com 26,34% preto (f) Vizinhança 70 com 25,81% preto

Figura 5: Efeito do aumento da vizinhança ilustrado pelo método de Bersen e a imagem retina