MC920 - INTRODUÇÃO AO PROCESSAMENTO DIGITAL DE IMAGEM
2S2019
RELATÓRIO - TRABALHO 2

Andressa Gabrielly Macedo Marçal
RA 262878

1. Especificação do Problema

O propósito deste trabalho é realizar experimentos com métodos de limiarização global e local em imagens monocromáticas. Métodos de limiarização global determinam um único valor de limiar para toda a imagem. Por outro lado, métodos de limiarização adaptativa local calculam um limiar para cada pixel da imagem com base na informação contida em uma vizinhança do pixel. Em ambos os casos, se um pixel p(x, y) na imagem de entrada possuir um valor mais alto que o limiar, então o pixel p(x, y) é classificado como objeto (preto), caso contrário, ele é rotulado como fundo (branco). Implemente métodos de limiarização global e local, conforme descrição abaixo, aplicando-os à imagem de entrada.

1.1. Método Global

Se o valor de intensidade de um pixel(x, y) for maior do que um limiar T (por exemplo, T = 128), o pixel será considerado como pertencente ao objeto; caso contrário, será considerado como fundo.

1.1.1 Método Otsu Global

Técnica que determina um limiar ótimo considerando uma imagem I, que apresenta melhor funcionamento em imagens cujos histogramas são bimodais. A ideia é aproximar o histograma de uma imagem por duas funções Gaussianas e escolher o limiar de forma a minimizar a variância intra-classes. Cada classe possui suas próprias características, ou seja, sua média e desvio-padrão.

Parâmetros e Tipos dos Dados do Código Otsu:

- @param1 **image:** a imagem de entrada;
  - **@type** image: ndarray;
- @param2 **hist:** o histograma da imagem de entrada;
  - **@type** hist: ndarray;
- **@return:** O limite de Otsu;
  - **@type** do return: int;

1.2. Método de Bernsen

$$T(x, y) = (z\min + z\max)/2$$

Parâmetros e Tipos dos Dados do Código Bernsen:

- @param1 **img:** a imagem de entrada. Deve ser uma imagem em escala de cinza;
  - **@type** img: ndarray;
- @param2 **w_size:** o tamanho da janela local para calcular cada limite de pixels. Deve ser uma janela de tamanho ímpar; Em suma é o Raio do kernel.
  - **@type** w_size: int;

- @param3 **c_thr:** o contraste do limiar para determinar uma região homogênea; O limiar de contraste terá esse valor padrão que é 15. Qualquer número diferente de 0 alterará o valor padrão.
  - **@type** c_thr: int;
- **@return:** o limite local estimado para cada pixel;
  - **@type** do return: ndarray;

1.3. Método de Niblack

T (x, y) = $\mu(x, y) + k\ \sigma(x, y)$

Parâmetros e Tipos dos Dados do Código Niblack:

- @param1 **img:** a imagem de entrada;
  - **@type** img: ndarray;
- @param2 **w_size:** o tamanho da janela local para calcular cada limite de pixels. Deve ser um valor ímpar; Em suma é o Raio do kernel.
  - **@type** w_size: int;
- @param3 **k:** controla o valor do limite local. Deve estar no intervalo [-0,2, -0,1]; O valor padrão é 0,2 para objetos brilhantes e -0,2 para objetos escuros. Qualquer outro número que não 0 alterará o valor padrão.
  - **@type** k: float;
- **@return:** o limite local estimado para cada pixel;
  - **@type** do return: ndarray;

1.4. Método de Sauvola e Pietaksinen

$$T(x, y) = \mu(x, y)\left[1 + k\ \left(\frac{\sigma(x, y)}{R} - 1\right)\right]$$

Parâmetros e Tipos dos Dados do Código Sauvola:

- @param1 **img:** a imagem de entrada;
  - **@type** img: ndarray;
- @param2 **w_size:** o tamanho da janela local a ser calculada cada limite de pixel. Deve ser um valor ímpar; Em suma é o Raio do kernel.
  - **@type** w_size: int;
- @param3 **k:** controla o valor do limite local. Encontra-se no intervalo [0,2, 0,5];
  - **@type** k: float;

  @return: o limite local estimado para cada pixel;
  - **@type** do return: ndarray;

1.5. Método de Phansalskar, More e Sabale

Esta é uma modificação do método de limiar de Sauvola para lidar com imagens de baixo contraste.

$$T = \mu(x, y)\ \left[1 + p\ \exp(-q\ \mu(x, y)) + k\ \left(\frac{\sigma(x, y)}{R} - 1\right)\right]$$

Parâmetros e Tipos dos Dados do Código Phansalskar:

- Nesse método, o limite t é calculado como
  - **T = média * (1 + p * exp (-q * média) + k * ((stdev / r) - 1))**
  - Onde média e stdev são a média local e o desvio padrão, respectivamente. Phansalkar recomenda utilizar na formula, os valores k = 0,25, r = 0,5, p = 2 e q = 10. Na formula k e r são os parâmetros 1 e 2 respectivamente, mas os valores de p e q são fixos.

  - @param1 : é o valor k . O valor padrão é 0,25. Qualquer outro número que não 0 alterará seu valor.
  - @param2 : é o valor r . O valor padrão é 0,5. Este valor é diferente do de Sauvola porque usa a intensidade normalizada da imagem. Qualquer outro número que não 0 alterará seu valor.

1.6. Método do Contraste

Atribui o valor de um pixel como fundo ou objeto, dependendo se seu valor está mais próximo do máximo ou mínimo local, respectivamente.

Parâmetros e Tipos dos Dados do Código Contraste Local:

- @param1 **img:** a imagem de entrada. Deve ser uma imagem em escala de cinza;
  - **@type** img: ndarray;
- @param2 **w_size:** o tamanho da janela local para calcular cada limite de pixels. Deve ser uma janela de numero ímpar; Em suma é o Raio do kernel.
  - **@type** w_size: int;
- **@return:** o limite local estimado para cada pixel;
  - **@type** do return: ndarray;

1.7. Método da Média

Seleciona o limiar como a média da distribuição local de intensidade. Dessa forma, se o valor de um pixel(x, y) for maior do que a média de sua vizinhança, o pixel será considerado como pertencente ao objeto; caso contrário, será considerado como fundo.

Parâmetros e Tipos dos Dados do Código Média Local:

- @param1 **img:** a imagem de entrada;
  - **@type** img: ndarray;
- @param2 **w_size:** o tamanho da janela local para calcular cada limite de pixels. Deve ser um valor ímpar; Em suma é o Raio do kernel.
  - **@type** w_size: int;
- **@return:** o limite local estimado para cada pixel;
  - **@type** do return: ndarray;
- **np.mean():** retorna a media da matriz;

1.8. Método da Mediana Local

Seleciona o limiar como a mediana da distribuição local de intensidade. Dessa forma, se o valor de um pixel (x, y) for maior do que a mediana de sua vizinhança, o pixel será considerado como pertencente ao objeto; caso contrário, será considerado como fundo.

Parâmetros e Tipos dos Dados do Código Mediada Local:

- @param1 **img:** a imagem de entrada;
  - **@type** img: ndarray;
- @param2 **w_size:** o tamanho da janela local para calcular cada limite de pixels. Deve ser um valor ímpar; Em suma é o Raio do kernel.
  - **@type** w_size: int;
- **@return:** o limite local estimado para cada pixel;
  - **@type** do return: ndarray;
- **np.median():** retorna a mediana da matriz;

2. Especificações do Script

- A implementação foi feita no **Jupyter Notebook**;
- A linguagem utilizada foi **Python** na sua versão 3.7.3;

- Os resultados foram processados e gerados em um arquivo PDF, para auxiliar de maneira mais clara e objetiva a analise e execução dos resultados;
- **As bibliotecas utilizadas foram:**
  - **Matplotlib:** Plotagens e gráficos;
  - **Math:** Operações Matemáticas;
  - **Numpy:** Operações Numéricas;
  - **OpenCV:** Processamento de Imagens;
  - **Scipy:** Operações em imagens;
  - **Scikit-Image:** Operações em Imagens;
  - **TimeIt:** Calculo de tempo de execução dos algoritmos;

2.1. Execução do Script

- O programa deve ser executado no terminal (tomando como base o S.O que foi executado os scritps, *Linux Ubuntu 16.04*), chamando o comando **jupyter-notebook** e ao abrir a interface, selecionar o arquivo *trabalho2_ra262878.ipynb*.
  - Executar blocos de células do arquivo **.ipynb** .
  - Nessa opção as células estarão disponíveis para reexecução, lembrando que foram executas e geradas através de toda a sua completude um relatório do jupyter notebook em formato PDF, que também segue em anexo no envio do projeto.

2.2. Função (apply_threshold):

A função obterá uma imagem binária com base em um determinado threshold global ou um conjunto de thresholds locais;

- Na função o threshold está setado por padrão igual a 128. **T = 128**
- O tamanho da janela padrão será igual a 15, pois notei que os resultados ficaram melhores. Mas foram feitos testes em janela 7x7 também, e ambos se encontram no diretorio Output, que segue em anexo no projeto.
- @param1 **img:**
  - Imagem de entrada
- @param2 **wp_val = 255**
  - O valor atribuído aos pixels do primeiro plano (pixels brancos).
- @param3 **threshold = 128**
  - O Threshold global ou local correspondentes para cada pixel da imagem. O valor passado como parâmetro foi 128.

2.3. Entrada de Dados

As imagens de entrada estão no formato **PGM (Portable GrayMap)**. Alguns exemplos encontram-se disponíveis no diretório:

http://www.ic.unicamp.br/ helio/imagens_pgm/

2.3.1 Leitura das Amostras

Das imagens de entrada usadas na implementação do projeto, foram utilizadas duas delas para comparativos nesse relatório, são elas **baboon.pgm** e **fiducial.pgm** . A figura abaixo mostrará o exemplo de imagem de entrada que foi utilizada para processar os resultados dos scripts.

A imagem será lida pela função **cv2.imread()** que encontra-se na biblioteca **OpenCV2**, na qual será passado o diretório onde as imagens estarão armazenadas no projeto.

2.4. Saída de Dados

As imagens processadas pelos algoritmos, serão armazenadas no diretório **Output** do projeto, que também seguirá em anexo.

3. Resultados

Nesta seção irei apresentar os resultados coletados da execução do algoritmo de cada abordagem

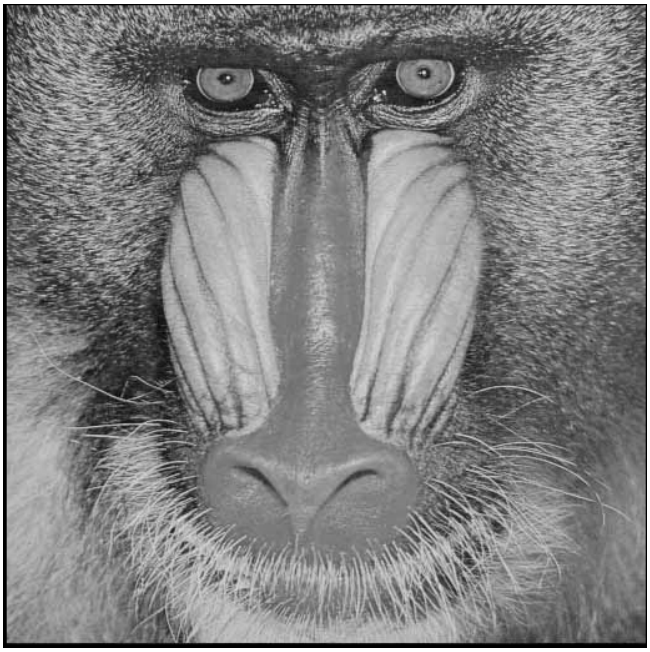

Figure 1. Baboon

Figure 2. Monarch

solicitada no trabalho. Lembrando que os resultados apresentados foram em cima de uma janela tamanho 15. Parametro w_size=15, quanto menor for o tamanho da janela, menos ”nítida” fica a imagem resultante.

3.1 Resultado do Histograma da Imagem

Para plotagem do histograma da imagem de entrada, foram utilizados os parâmetros:

Parâmetros da Função:

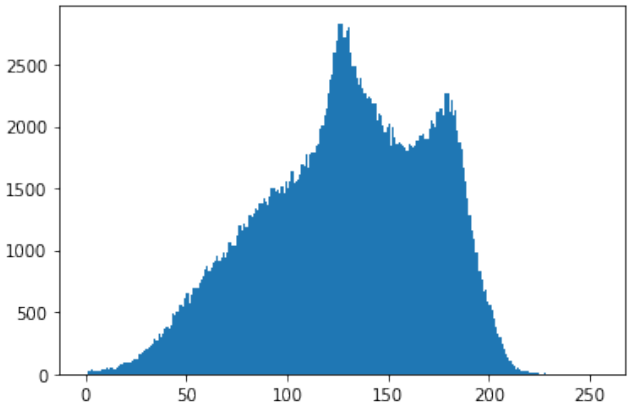

Figure 3. Histograma da Imagem de entrada baboon.pgm

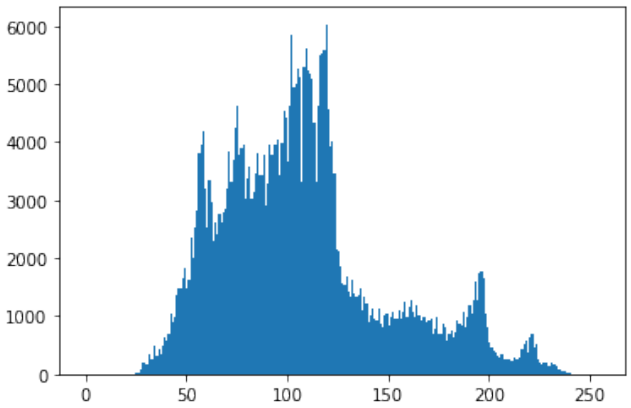

Figure 4. Histograma da Imagem de entrada monarch.pgm

- **img.ravel():** Função geradora da biblioteca numpy que retorna uma matriz 1-D, contendo os elementos da entrada.
- **range = (0, 255):** Intervalo da Imagem, nesse caso x.min() e x.max();
- **bins = 255:** Dividi o histograma inteiro em 16 sub-partes, e o valor de cada sub-parte é a soma de toda a contagem de pixels nele. Essa sub-parte é chamada de ”**BIN**”. BINS é representado pelo termo *histSize* na documentação do OpenCV.
- O histograma apresentará informações como **Amount of pixels** x **Gray Level**

3.2. Resultado do Método Otsu

Na forma mais simples, o algoritmo retorna um limite de intensidade única que separa os pixels em duas classes, primeiro e segundo plano. Esse limiar é determinado minimizando a variação de intensidade intra-classe ou, equivalentemente, maximizando a variação entre classes

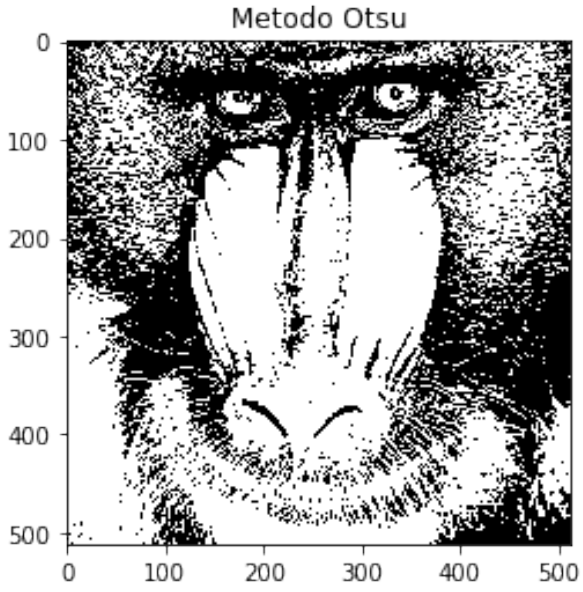

Figure 5. Baboon Otsu Global

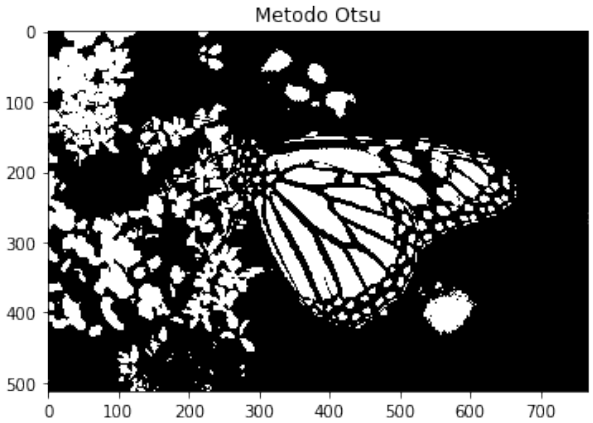

Figure 6. Monarch Otsu Global

- Tempo de Execução do Algoritmo Otsu [Baboon]: **0.014848988997982815**
- Quantidade Total de Pixels[Baboon]: **262144**
- Quantidade de Pixels Brancos[Baboon]: **0.9999847412109375**
- Quantidade de Pixels Pretos[Baboon]: **1.52587890625e-05**

3.3 Resultado do Método Bernsen

Na implementação, se o contraste local(max-min) for superior ou igual ao limite de contraste , o limite será definido no valor médio de cinza local (a média dos valores mínimo e máximo de cinza na janela local). Se o contraste local estiver abaixo do limite de contraste, a vizinhança é considerada composta apenas por uma classe e o pixel é definido como objeto ou plano de fundo, dependendo do valor do meio-cinza.

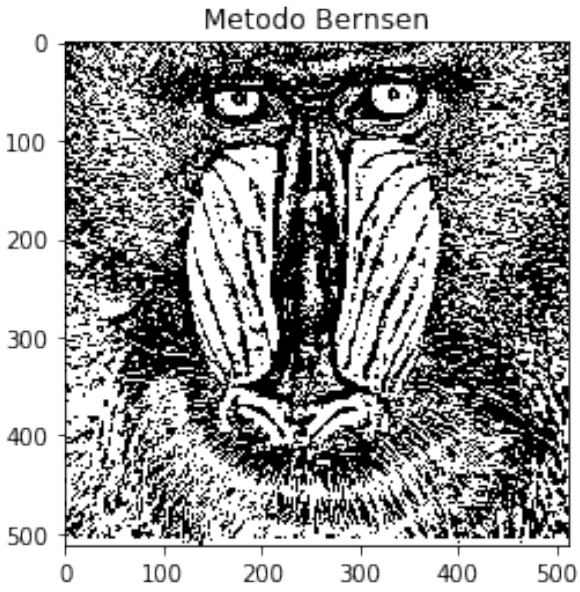

Figure 7. Baboon Bernsen

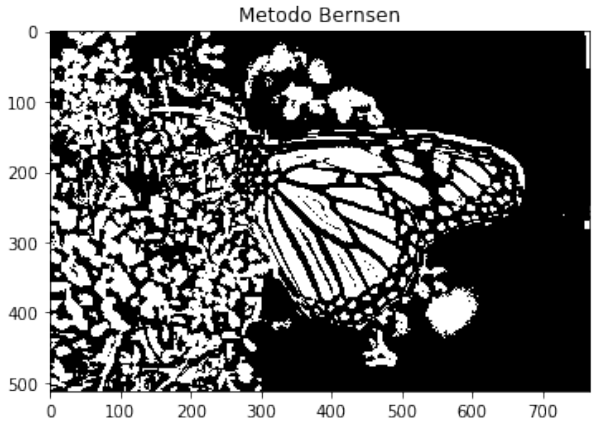

Figure 8. Monarch Bernsen

- Tempo de Execução do Algoritmo Bernsen[Baboon]: **0.296950703006587**
- Quantidade de Pixels Brancos[Baboon]: **0.872281901041**
- Quantidade de Pixels Pretos[Baboon]: **0.12771809895**

3.4 Resultado do Método Niblack

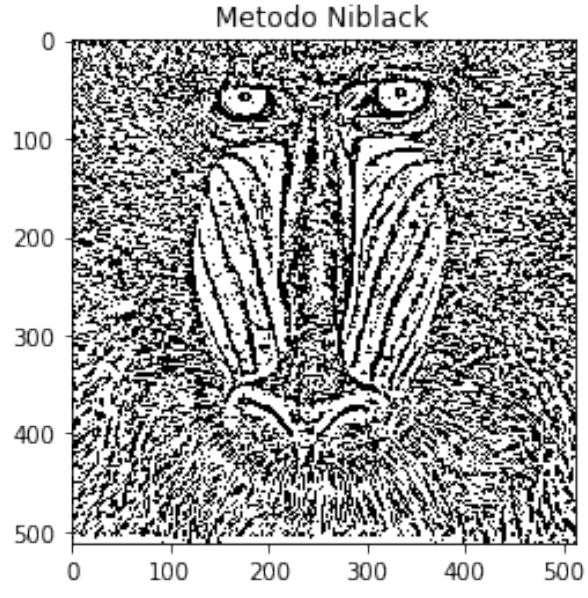

Figure 9. Baboon Niblack

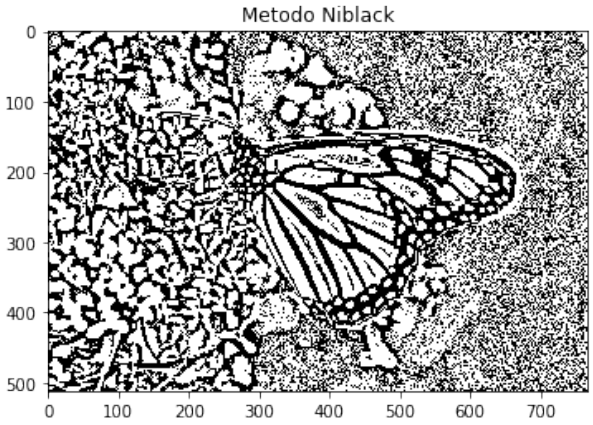

Figure 10. Monarch Niblack

- Tempo de Execução do Algoritmo Niblack[Baboon]: **0.03449465200537816**
- Quantidade de Pixels Brancos[Baboon]: **0.475390625**
- Quantidade de Pixels Pretos[Baboon]: **0.524609375**

3.5 Resultado do Método Sauvola

Implementa o método de limiar de Sauvola, que é uma variação do método de Niblack.

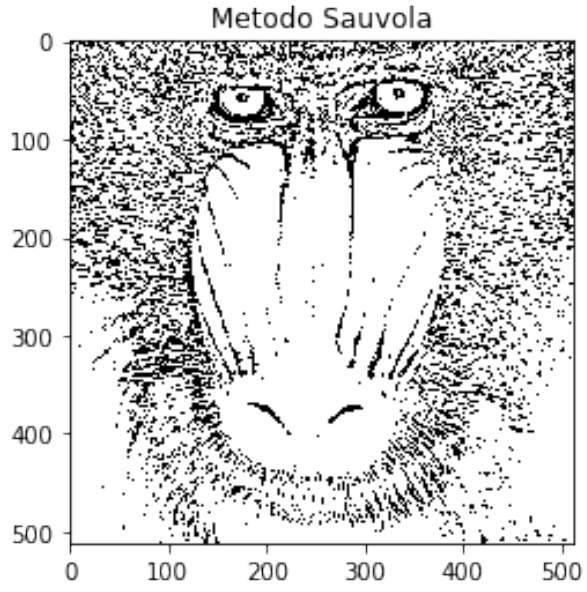

Figure 11. Baboon Sauvola

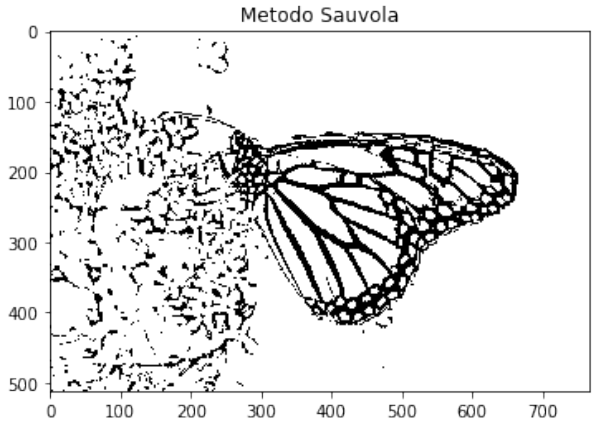

Figure 12. Monarch Sauvola

- Tempo de Execução do Algoritmo de Sauvola[Baboon]: **0.04442485600884538**
- Quantidade de Pixels Brancos[Baboon]: **0.2234375**
- Quantidade de Pixels Pretos[Baboon]: **0.7765625**

3.6 Resultado do Método Phansalskar

O Algoritmo é uma modificação do método de Sauvola para lidar com imagens de baixo contraste.

- Tempo de Execução do Algoritmo de Phansalkar: **0.00627090799389407**
- Quantidade de Pixels Brancos: **0.2234375**
- Quantidade de Pixels Pretos: **0.7765625**

Figure 13. Baboon Phansalskar

Figure 14. Monarch Phansalskar

3.7 Resultado do Método Contraste

Define o valor do pixel como branco (255) ou preto (0), dependendo se o valor atual estiver mais próximo do máximo ou do mínimo local, respectivamente.

- Tempo de Execução do Algoritmo de Contraste: **0.2799548059992958**
- Quantidade de Pixels Brancos[Baboon]: **0.5255859375**
- Quantidade de Pixels Pretos[Baboon]: **0.4744140625**

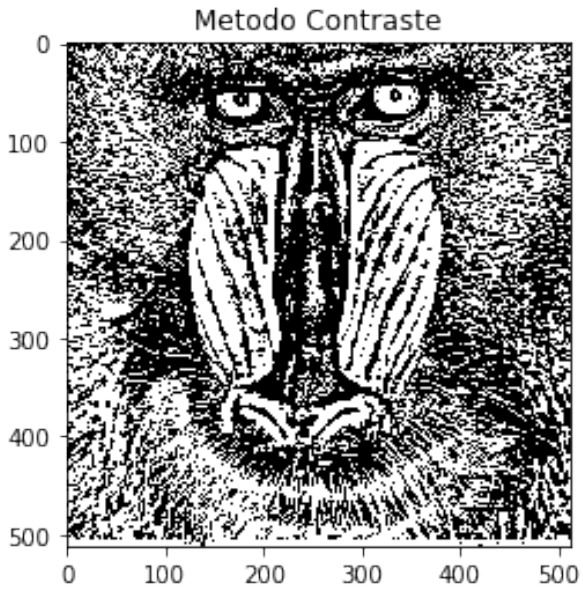

Figure 15. Baboon Contraste

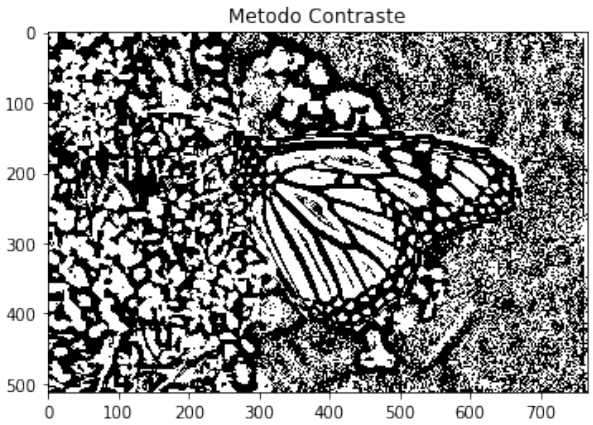

Figure 16. Monarch Contraste

3.8 Resultado do Método Média

- Tempo de Execução do Algoritmo de Média[Baboon]: **0.028969647974008694**
- Quantidade de Pixels Brancos[Baboon]: **0.5815559895833333**
- Quantidade de Pixels Pretos[Baboon]: **0.4184440104166667**

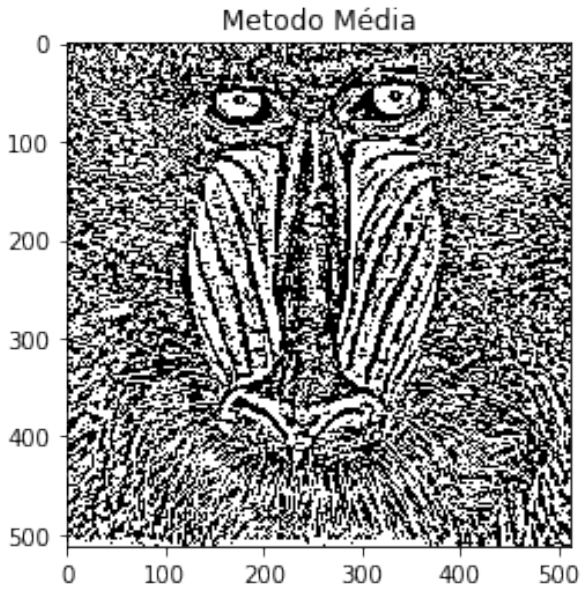

Figure 17. Baboon Média

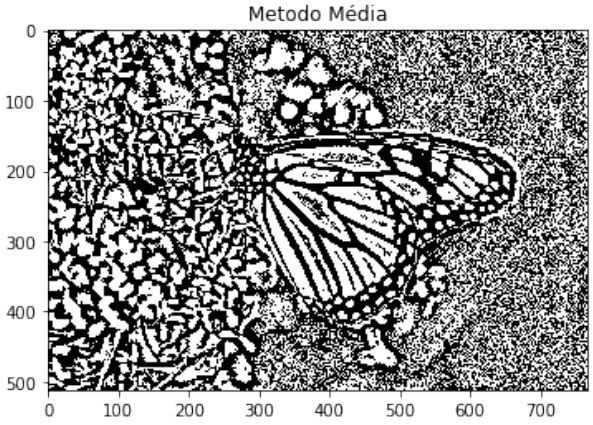

Figure 18. Monarch Média

3.9 Resultado do Método Mediana

Seleciona o limite como a mediana da distribuição em escala de cinza local.

- Tempo de Execução do Algoritmo de Mediana[Baboon]: **0.04570262398920022**
- Pixels Brancos[Baboon]: **0.9999847412109375**
- Pixels Pretos[Baboon]: **1.5258789062555505**

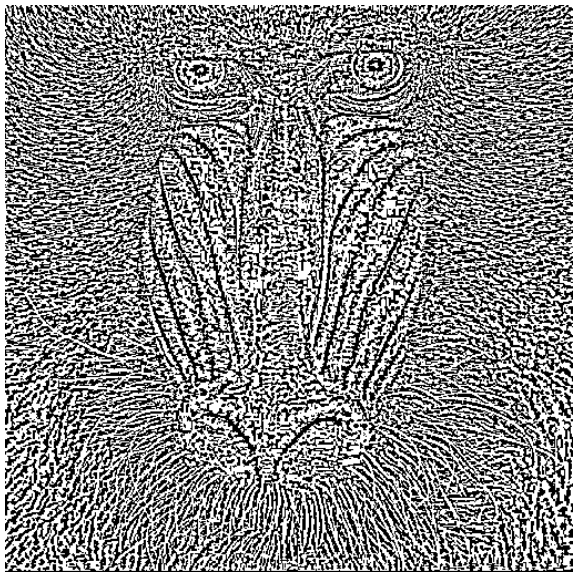

Figure 19. Baboon Mediana

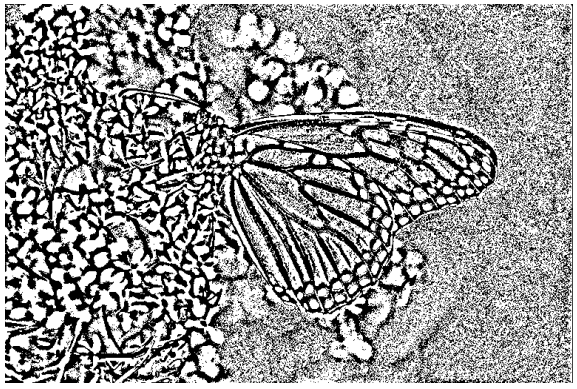

Figure 20. Monarch Mediana

4. Dificuldades Encontradas

- Quanto maior a janela, maior o tempo de processamento, então o melhor resultando em um tempo considerável, foi o tamanho da janela 15. A dificuldade maior foi só processar imagens em janelas maiores, algumas vezes meu notebook chegou a superaquecer e travou tudo, perdi até alguns testes que estava fazendo.
- Notei depois que tinha algumas coisas prontas na scikit-image e que teria facilitado o desenvolvimento se eu tivesse visto no inicio, mas de fato é notório o avanço nesse trabalho, próxima vez irei olhar com mais atenção a documentação do scikit image para ganhar tempo.

5. Conclusão

Todas as imagens que foram geradas nesse Trabalho2 foram anexas no diretório Output. O arquivo completo, com todos os arquivos, imagens processadas e relatorios irá compactado na extensão .zip, com nome **trabalho2_ra262878.zip**.

- **Output:**
  - **output:** Imagens processadas com cada método citado;
  - **pdf:** A execução do Jupyter Notebook e seus comentários;
  - **script:** Todos os algoritmos de todos os métodos, sejam eles Global ou Local, do trabalho 2, na linguagem Python;

Dentro do diretório principal também estará contido este relatório em PDF, a execução do Jupyter Notebook em PDF, o script .ipynb, e as imagens que foram usadas como entrada e os resultados da saída do seu processamento.