Processamento Digital de Imagens: aplicação de filtros de suavização na redução de ruído sal e pimenta

Borges, C. D. B.

Resumo A existência de ruído em imagens é um fator prejudicial durante sua análise, seja através da visão humana, seja por processamento computacional. Sua presença dificulta a visualização e extração de características relevantes, o que causa decréscimo de performance na execução de tarefas dependentes da análise de imagens. Técnicas para redução ou eliminação de ruído são, portanto, extremamente importantes. O presente trabalho estuda o comportamento de duas técnicas de suavização, os filtros média e mediana, sobre um tipo específico de ruído, denominando sal e pimenta.

1 Introdução

O problema da redução de ruído em imagens pode ser tratado no domínio espacial através de filtros de suavização. Esses filtros são categorizados como passa-baixa, pois eliminam informações de alta frequência, comumente associadas a ruído, enquanto mantêm informações de baixa frequência [1]. A filtragem linear no domínio espacial consiste na convolução da imagem com uma matriz denominada *kernel* (Equação 1) [2].

$$I' = I * \kappa \tag{1}$$

Na Equação 1, I é a imagem de entrada, I' a imagem de saída, $*$ é o operador de convolução 2D e κ é denominado *kernel* do filtro bidimensional. O *kernel* define a natureza e os atributos da transformação aplicada à imagem. Exemplos de *kernels* são mostrados nas Equações 2, 3, 4 e 5, possíveis variantes do filtro média, usado para suavização.

No entanto, nem todos os filtros de suavização no domínio do espaço podem ser expressos como uma convolução. Esse é o caso do filtro mediana, um operador não-linear que realiza, para cada *pixel* da imagem, uma ordenação dos *pixels* em sua vizinhança e subsequente seleção do elemento da mediana para substituir o valor do *pixel* central [2].

Neste trabalho, os desempenhos dos filtros média e mediana foram examinados sobre quatro imagens afetadas por ruído sal e pimenta, efeito causado por perturbações abruptas no sinal da imagem e percebido como ocorrências dispersas de *pixels* pretos e brancos [2]. Um total de sete variantes de filtros de suavização foram testadas.

Quatro variantes do filtro média foram avaliadas; seus *kernels* são mostrados nas Equações 2, 3, 4 e 5. Três variantes do filtro mediana foram testadas, operando sob janelas de tamanho 3×3, 5×5 e 7×7.

$$\kappa_1 = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2}$$

$$\kappa_2 = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{3}$$

$$\kappa_3 = \frac{1}{49}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{4}$$

$$\kappa_4 = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{5}$$

2 Metodologia

O primeiro passo nesse experimento foi a normalização das imagens da base de dados para um formato comum. Todas as imagens foram convertidas para JPEG. Em

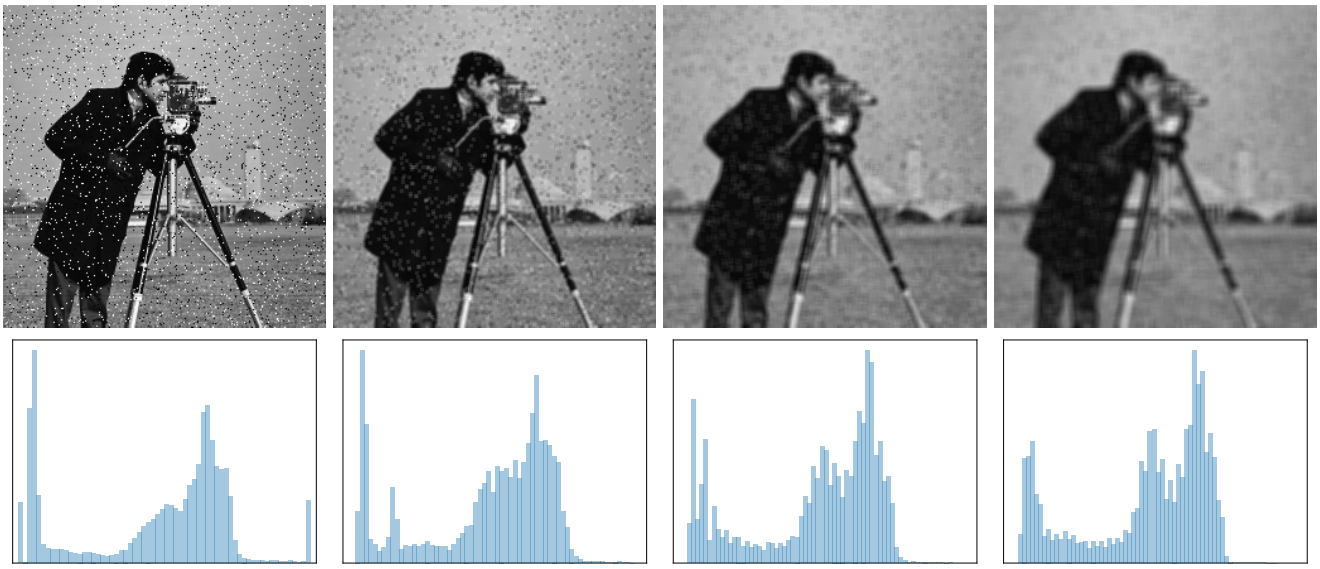

Figura 1 Resultados obtidos com o filtro média sobre a imagem *cameraman* e seus histogramas. Da esquerda para a direita: imagem com ruído, filtragem com κ_1, κ_2 e κ_3

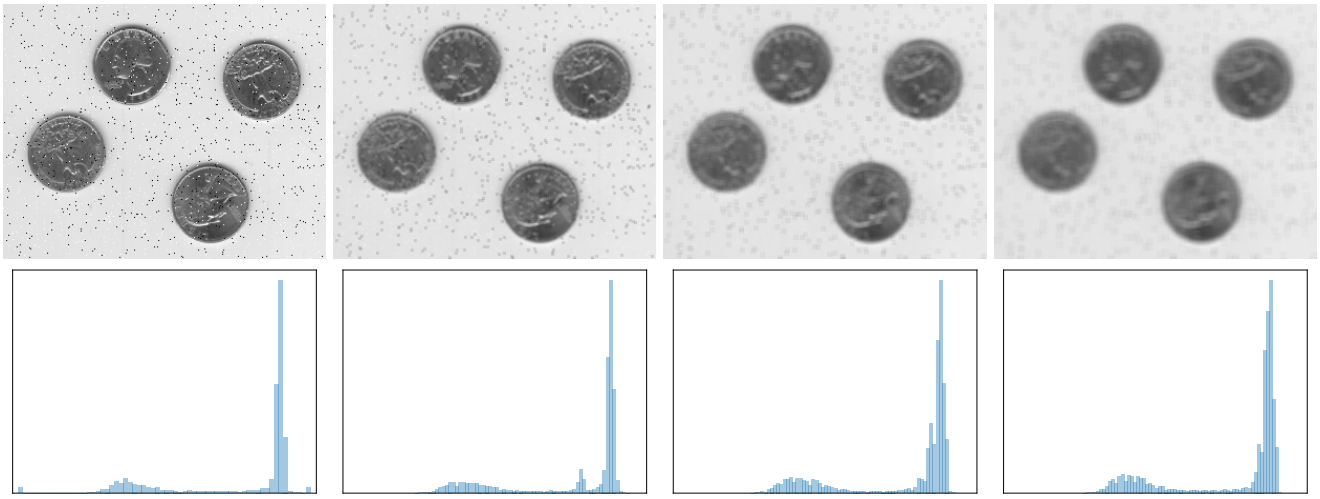

Figura 2 Resultados obtidos com o filtro média sobre a imagem *coins* e seus histogramas. Da esquerda para a direita: imagem com ruído, filtragem com κ_1, κ_2 e κ_3

seguida, um script na linguagem Python 3.7 [3] foi produzido para fazer a leitura das imagens e aplicação dos filtros através do pacote OpenCV [4]. O script também é responsável por gerar os histogramas de cada imagem, utilizando Matplotlib [5] e Seaborn [6]. O código utilizado no experimento encontra-se no Apêndice A.

3 Resultados do filtro média

Nas Figuras 1, 2, 3 e 4 são exibidos os resultados obtidos para as imagens *cameraman*, *coins*, *peppers* e *lena*, respectivamente. Também são mostrados os histogramas das intensidades imediatamente abaixo de suas imagens correspondentes.

Nos histogramas das imagens originais, pode-se observar o efeito do ruído sal e pimenta através de concentrações de *pixels* relativamente altas em ambos os extremos de intensidade. Esse efeito é mais pronunciado nas imagens *peppers* e *lena*, embora seja perceptível também em *cameraman* e *coins*. Esse efeito é substancialmente reduzido com a aplicação dos filtros. Nota-se que, independentemente do tamanho do *kernel*, as altas concentrações em ambos os extremos, observadas nos histogramas das imagens ruidosas, não estão presentes nos histogramas das imagens filtradas.

Vê-se que o ruído sal e pimenta é removido com o filtro média, no entanto, nota-se nas imagens filtradas a produção de artefatos indesejáveis, *pixels* de intensida-

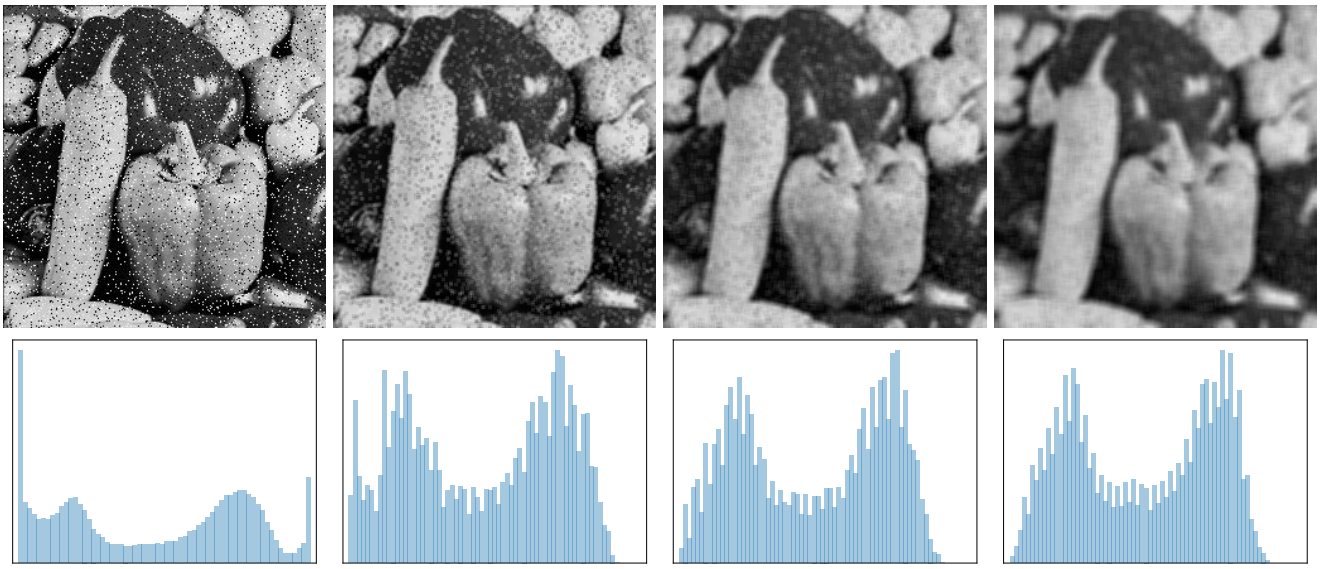

Figura 3 Resultados obtidos com o filtro média sobre a imagem *peppers* e seus histogramas. Da esquerda para a direita: imagem com ruído, filtragem com κ_1, κ_2 e κ_3

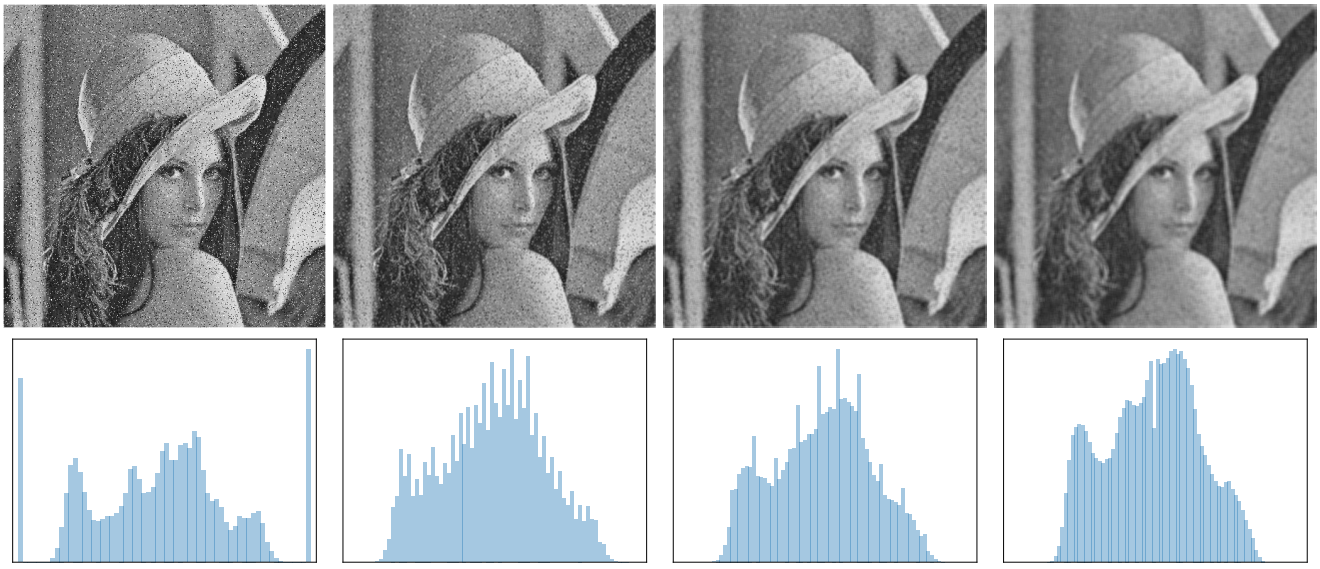

Figura 4 Resultados obtidos com o filtro média sobre a imagem *lena* e seus histogramas. Da esquerda para a direita: imagem com ruído, filtragem com κ_1, κ_2 e κ_3

des anormalmente altas ou baixas, mas não necessariamente extremas, remanescentes do ruído original. Esses artefatos são reduzidos conforme o aumento das dimensões do *kernel*, contudo, ocorre também uma progressiva perda da riqueza de detalhes da imagem, especialmente nas regiões de borda. Como exemplo, as imagens gravadas nas moedas da figura *coins* são reconhecíveis na imagem ruidosa e nas imagens filtradas com *kernels* 3×3 e 5×5, mas tornam-se praticamente irreconhecíveis após aplicação do filtro com *kernel* 7×7.

Já nas Figuras 5, 6, 7 e 8, são comparados os resultados dos filtros média aplicados com *kernels* κ_1 e κ_4, ambos 3×3. Em relação à quantidade de artefatos remanescentes do ruído original, não se observa muita diferença. Todavia, o filtro média aplicado com *kernel* κ_4 sofre menos com o efeito da perda de detalhes e borramento em regiões de borda. Nota-se por exemplo, que o rosto do homem na imagem *cameraman* é muito mais nítido com a aplicação de κ_4 do que com o uso do *kernel* κ_1. O mesmo pode ser observado nas figuras *coins* e *peppers*. Já na imagem *lena*, esse efeito é menos perceptível.

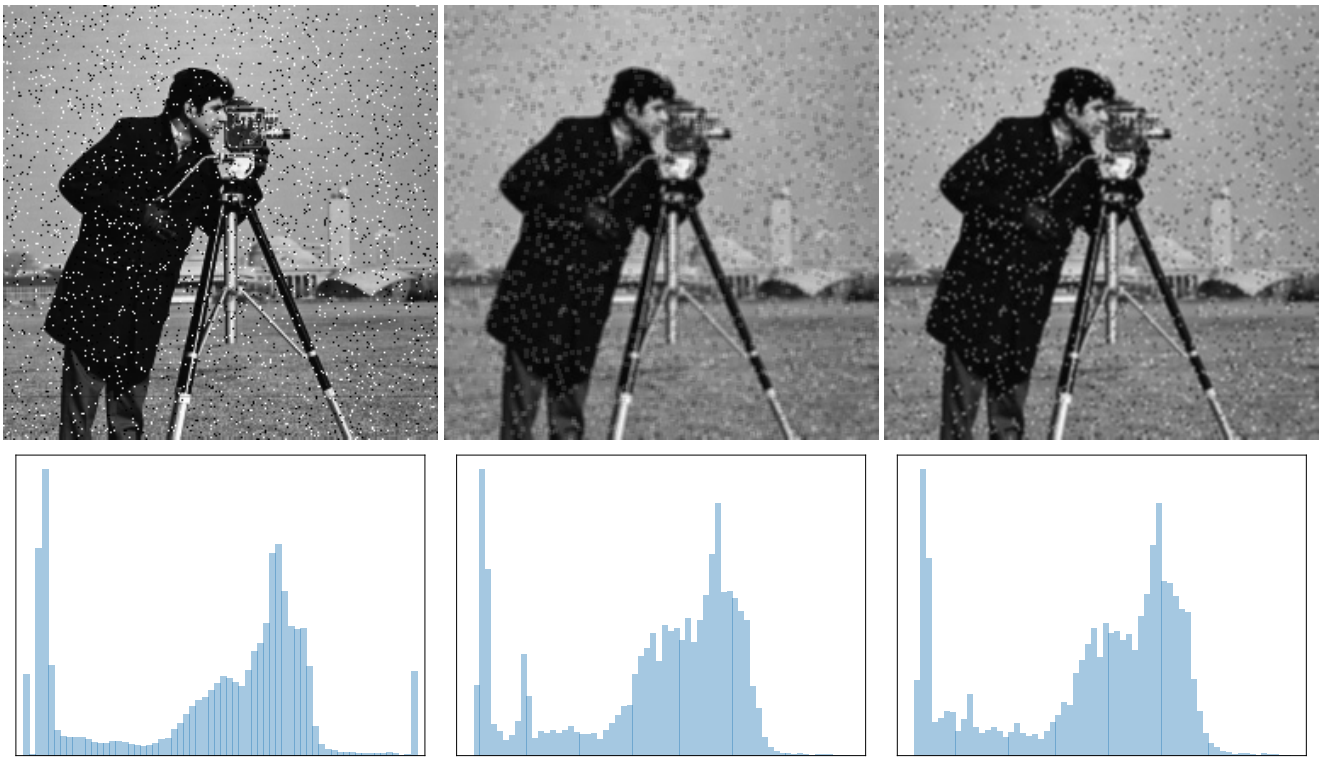

Figura 5 Resultados obtidos com o filtro média customizado sobre a imagem *cameraman* e seus histogramas. Da esquerda para a direita: imagem com ruído, filtragem com κ_2 e κ_4

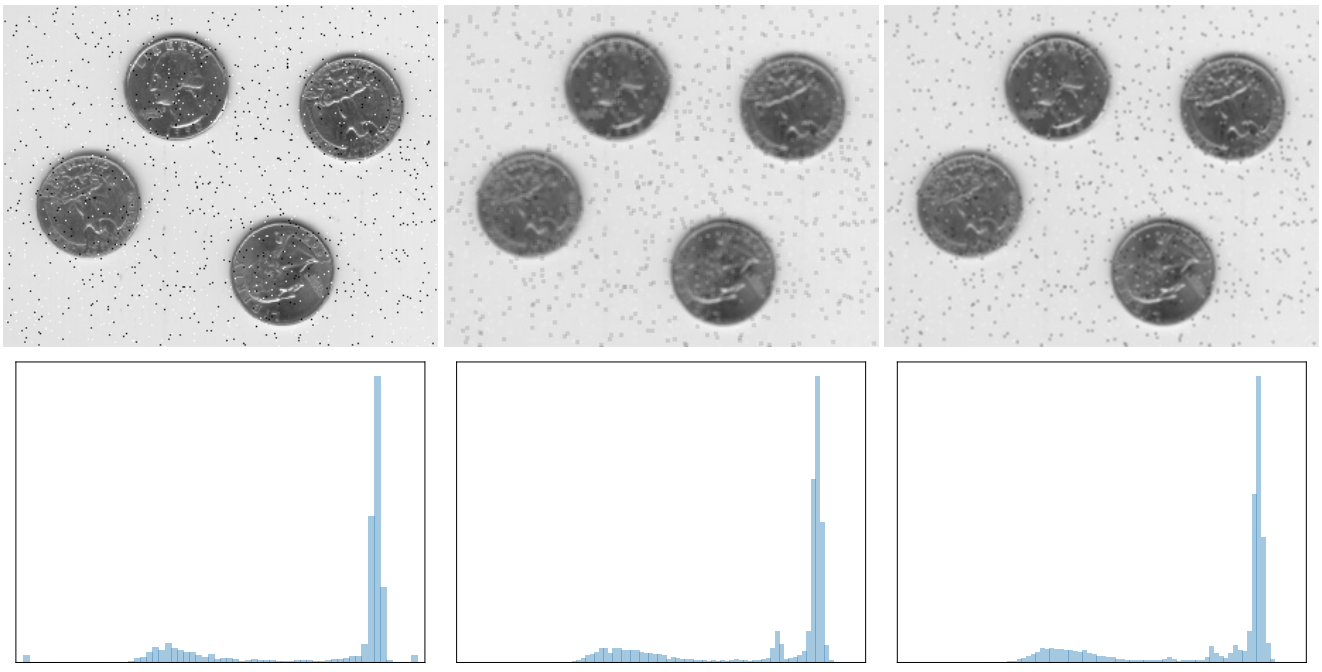

Figura 6 Resultados obtidos com o filtro média customizado sobre a imagem *coins* e seus histogramas. Da esquerda para a direita: imagem com ruído, filtragem com κ_2 e κ_4

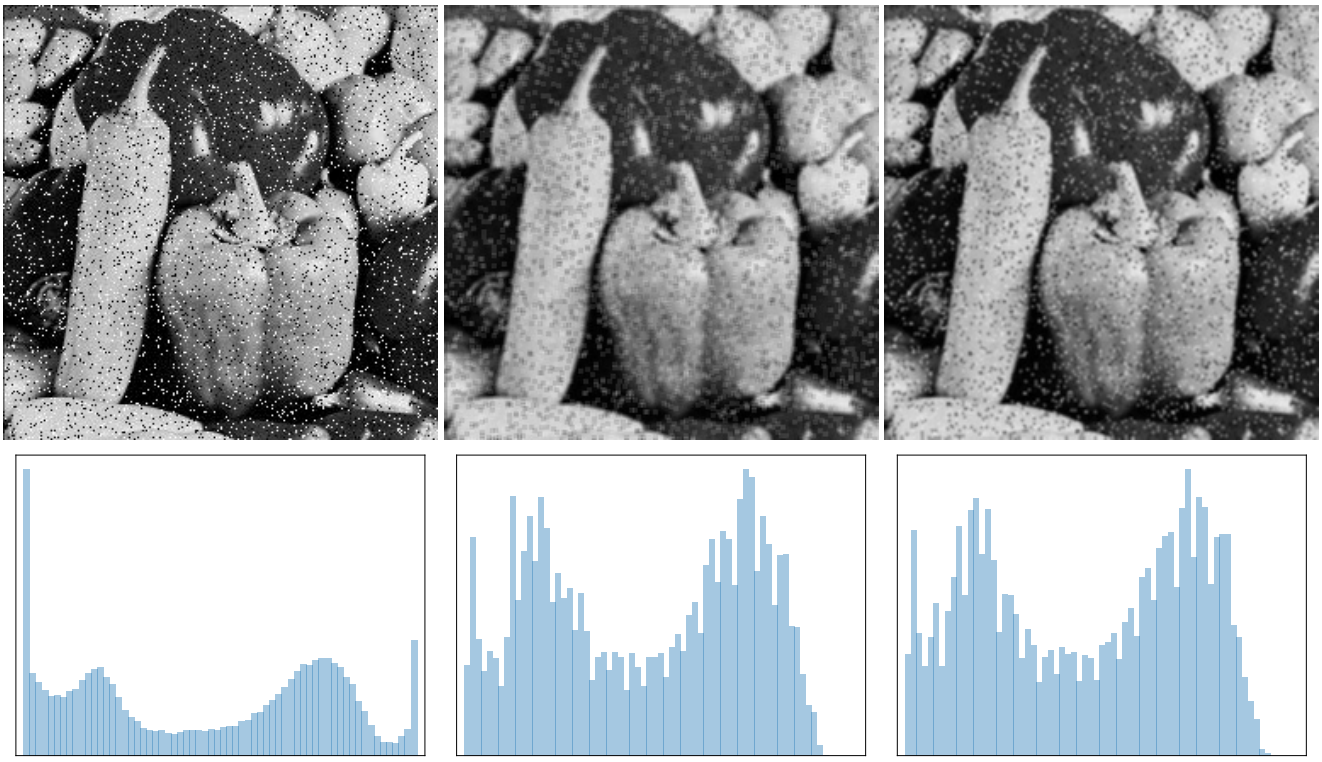

Figura 7 Resultados obtidos com o filtro média customizado sobre a imagem *peppers* e seus histogramas. Da esquerda para a direita: imagem com ruído, filtragem com κ_2 e κ_4

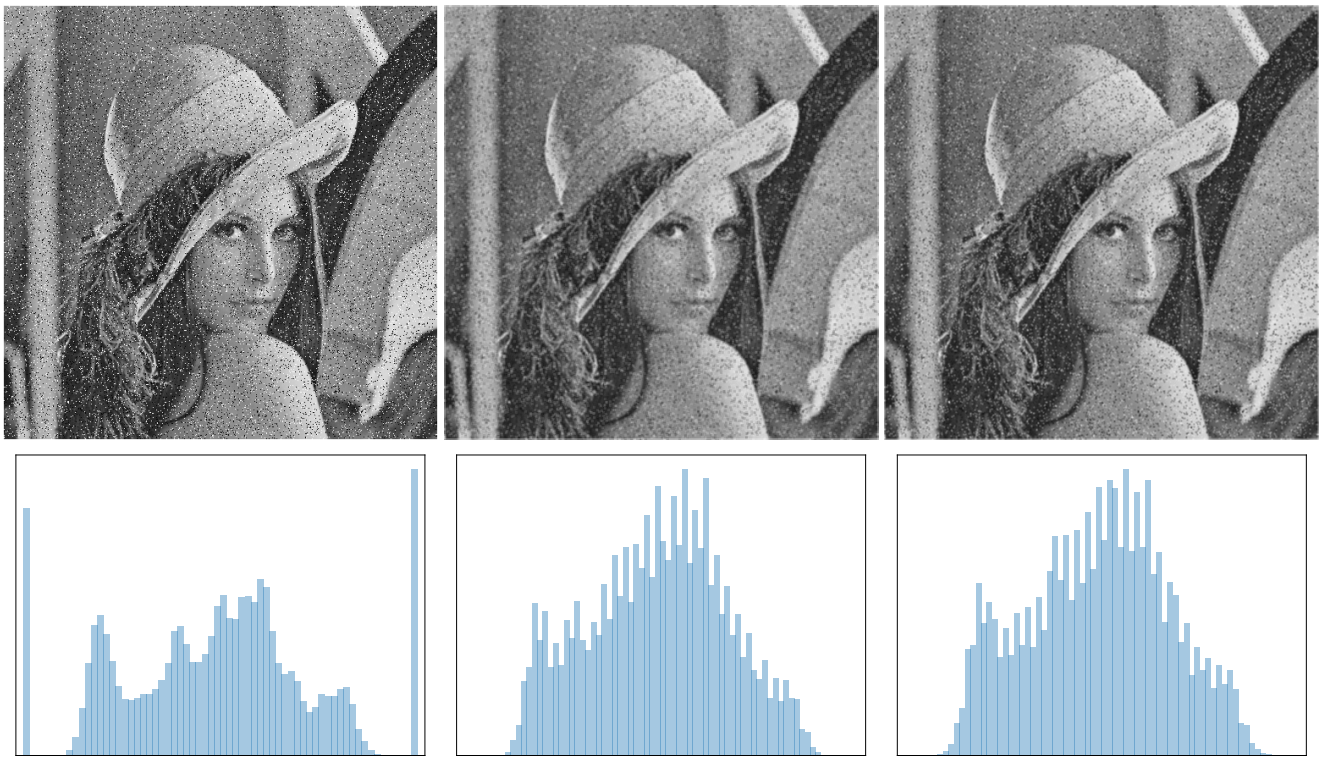

Figura 8 Resultados obtidos com o filtro média customizado sobre a imagem *lena* e seus histogramas. Da esquerda para a direita: imagem com ruído, filtragem com κ_2 e κ_4

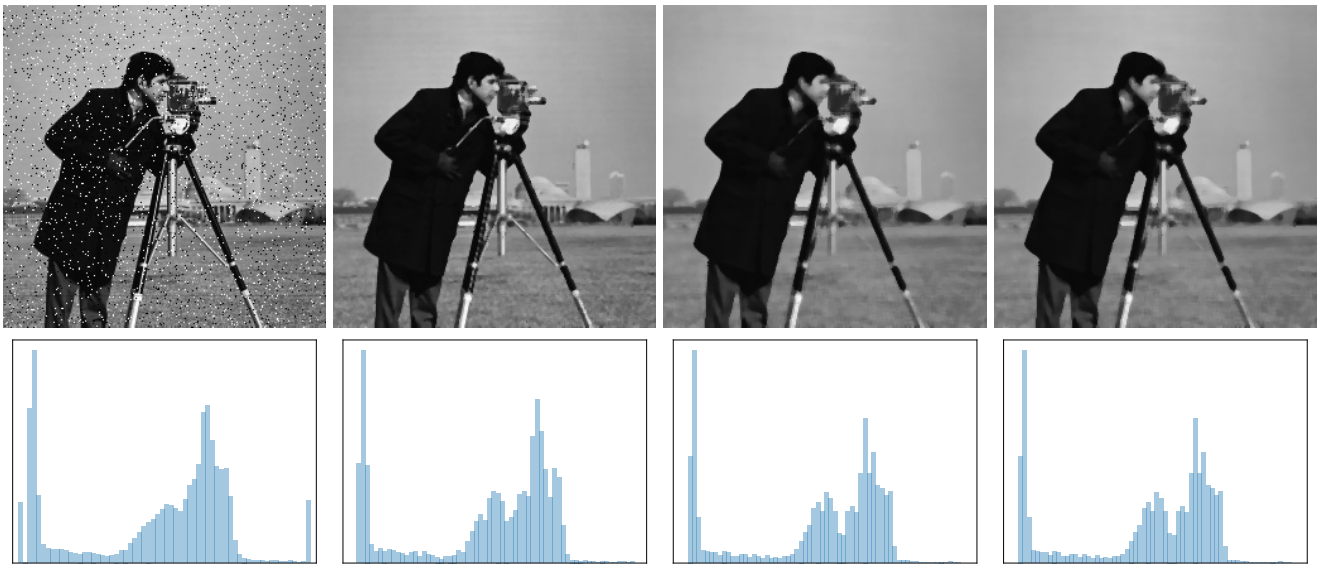

Figura 9 Resultados obtidos com o filtro mediana sobre a imagem *cameraman* e seus histogramas. Da esquerda para a direita: imagem com ruído e aplicações do filtro mediana com janelas 3×3, 5×5 e 7×7

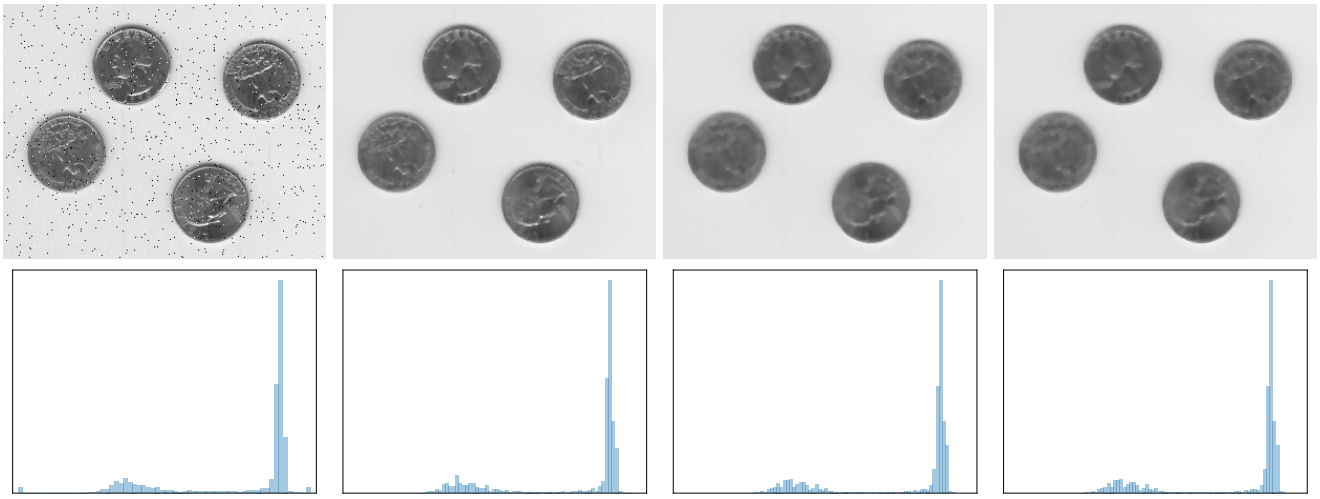

Figura 10 Resultados obtidos com o filtro mediana sobre a imagem *coins* e seus histogramas. Da esquerda para a direita: imagem com ruído e aplicações do filtro mediana com janelas 3×3, 5×5 e 7×7

4 Resultados do filtro mediana

Nas Figuras 9, 10, 11 e 12 são exibidos os resultados obtidos com o filtro mediana para as imagens *cameraman*, *coins*, *peppers* e *lena*, respectivamente. Os histogramas correspondentes são exibidos abaixo de cada imagem.

Através da mesma lógica utilizada anteriormente, nota-se o desaparecimento da assinatura do ruído sal e pimenta nos histogramas das imagens filtradas, independentemente das dimensões das janelas de processamento. Portanto, o filtro mediana efetivamente elimina o ruído sal e pimenta. Adicionalmente, observa-se nas imagens filtradas a quase inexistência de artefatos remanescentes do ruído original. Nota-se nas imagens filtradas que o aumento do tamanho da janela de processamento causa uma redução ainda mais forte dos artefatos, mas também uma progressiva perda de detalhes. Essa perda é perceptível em regiões como o rosto do homem na figura *cameraman*, as imagens nas faces das moedas em *coins* e no enfeite do chapéu de *lena*.

5 Conclusões

O presente trabalho fez uma análise acerca da filtragem de ruído sal e pimenta através de duas técnicas de suavização no domínio espacial: os filtros média e mediana. Sete variantes desses filtros foram aplicadas a quatro imagens. Os resultados foram examinados visualmente de acordo com níveis de eliminação do ruído,

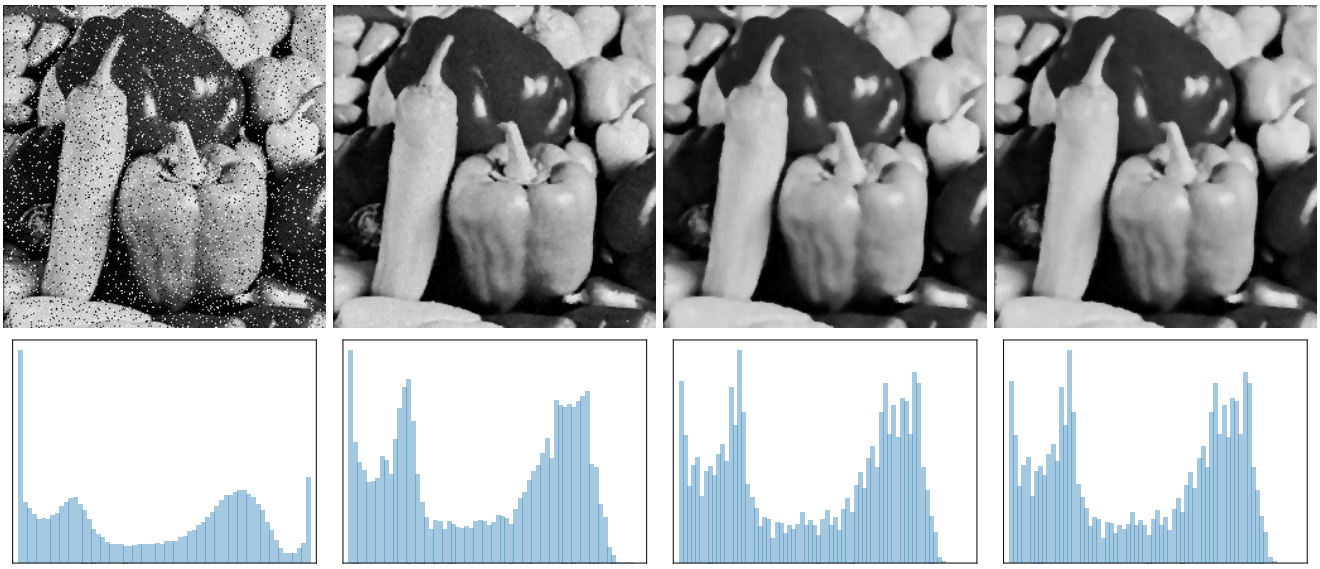

Figura 11 Resultados obtidos com o filtro mediana sobre a imagem *peppers* e seus histogramas. Da esquerda para a direita: imagem com ruído e aplicações do filtro mediana com janelas 3×3, 5×5 e 7×7

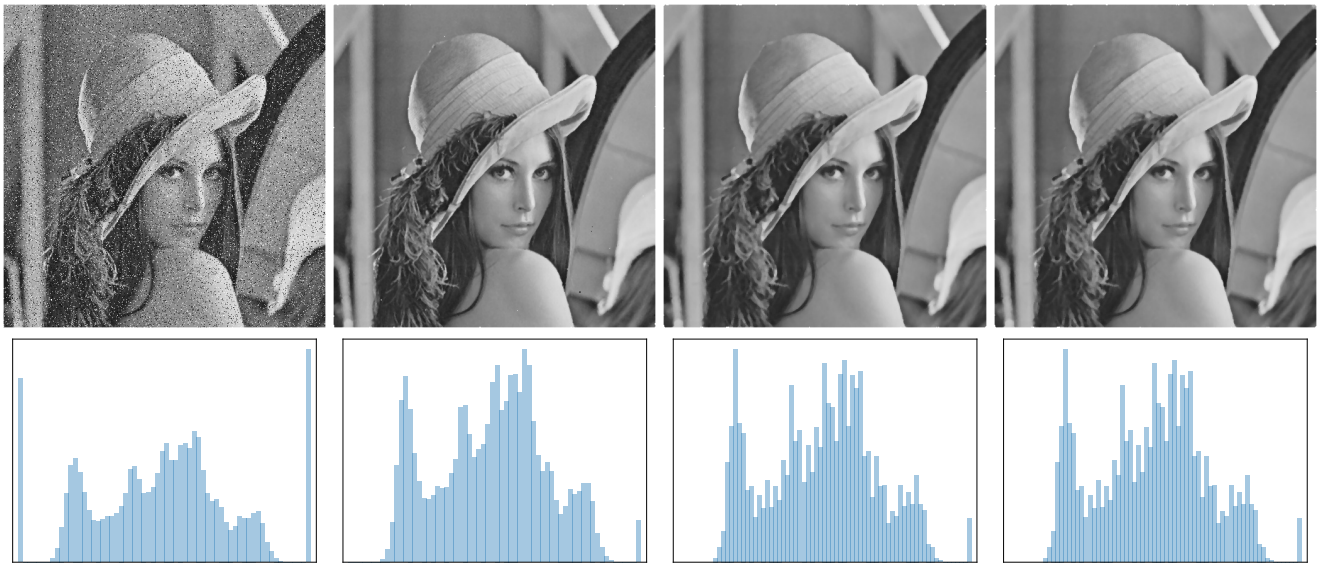

Figura 12 Resultados obtidos com o filtro mediana sobre a imagem *lena* e seus histogramas. Da esquerda para a direita: imagem com ruído e aplicações do filtro mediana com janelas 3×3, 5×5 e 7×7

produção de artefatos, borramento e a presença da assinatura do ruído sal e pimenta nos histogramas das imagens. A análise feita permite dizer que o filtro média reduz o ruído sal e pimenta, no entanto, artefatos remanescentes do ruído original não são efetivamente eliminados. Ocorre redução dos artefatos conforme o tamanho do *kernel* do filtro aumenta, mas isso acontece simultaneamente a um progressivo borramento de detalhes e regiões de bordas. O uso de um *kernel* modificado, com pesos maiores no centro e menores nas margens, reduz de forma sutil o borramento. A mesma análise sobre o filtro mediana mostra que esse é capaz de eliminar o ruído sal e pimenta, enquanto não produz artefatos remanescentes. O aumento das dimensões da janela de processamento também gera um progressivo borramento de detalhes, no entanto esse efeito é nitidamente menor que o observado no filtro média. O estudo dos resultados obtidos leva à conclusão de que o filtro mediana 3×3 obteve o melhor desempenho na eliminação de ruído sal e pimenta. Essa afirmativa é corroborada pela eliminação da assinatura do ruído sal e pimenta nos histogramas, a quase inexistência de artefatos remanescentes e a melhor conservação de detalhes em relação às imagens filtradas com os outros métodos.

Referências

1. R. Gonzalez, R. Woods, ***Processamento Digital de Imagens***, 3rd edn. (Pearson Prentice Hall, São Paulo, 2010)
2. D. Forsyth, J. Ponce, ***Computer Vision: A Modern Approach***, 2nd edn. (Pearson, New Jersey, 2011)
3. Python (2018). URL https://www.python.org/
4. OpenCV (2018). URL https://opencv.org/
5. Matplotlib (2018). URL https://matplotlib.org/
6. Seaborn (2018). URL https://seaborn.pydata.org/

Apêndice A

```
import os
import cv2
import numpy
import matplotlib.pyplot as plt
import seaborn

import warnings
warnings.filterwarnings("ignore")

input_path = './images'
output_path = './output'

if not os.path.exists(output_path):
    os.mkdir(output_path)

def meanfilter_3x3(image):
    return cv2.blur(image, (3,3))

def meanfilter_5x5(image):
    return cv2.blur(image, (5,5))

def meanfilter_7x7(image):
    return cv2.blur(image, (7,7))

def meanfilter_custom_3x3(image):
    k = (1/16)*numpy.array([[1, 2, 1], [2, 4, 2], [1, 2, 1]])
    return cv2.filter2D(image, -1, k)

def medianfilter_3x3(image):
    return cv2.medianBlur(image, 3)

def medianfilter_5x5(image):
    return cv2.medianBlur(image, 5)

def medianfilter_7x7(image):
    return cv2.medianBlur(image, 5)

def histogram(path, image):
    x = numpy.array([p for row in image for p in row])
    seaborn.distplot(x, bins=64, kde=False, norm_hist=True)
    plt.xlim([-5, 260])
    plt.yticks([])
    plt.xticks([])
    plt.tight_layout()
    plt.savefig(path)
    plt.clf()

dataset = dict()

for image_name in os.listdir(input_path):
    id_ = image_name.split('.')[0]
    dataset[id_] = cv2.imread(os.path.join(input_path, image_name), cv2.IMREAD_GRAYSCALE)
    histogram(os.path.join(output_path, '_'.join([id_, 'original']))+'.pdf', dataset[id_])

functions = [meanfilter_3x3, meanfilter_5x5, meanfilter_7x7, meanfilter_custom_3x3,
                medianfilter_3x3, medianfilter_5x5, medianfilter_7x7]

for id_ in dataset:
    for f in functions:
        output = f(dataset[id_])
        cv2.imwrite(os.path.join(output_path, '_'.join([id_, f.__name__]))+'.jpg', output)
        histogram(os.path.join(output_path, '_'.join([id_, f.__name__]))+'.pdf', output)
```