INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SÃO PAULO Campus Birigui	**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA** **Campus Birigui** **Bacharelado em Engenharia de Computação**
Disciplina: Processamento Digital de Imagens	Trabalho sobre alteração de imagens utilizando variadas bibiliotecas
Professor: Prof. Dr. Murilo Varges da Silva	**Data:** 25/08/2023
Nome do aluno: Leonardo Reneres dos Santos	**Prontuário:** BI3009131

Questões:

Para todos os exercícios utilizar as imagens lena, cameraman e house. Estão disponíveis no Moodle.

1. **[OPERAÇÃO PONTO A PONTO]:**
    1. Calcular o negativo das imagens;
    2. Diminuir pela metade a intensidade dos pixels;
    3. Incluir 4 quadrados brancos 10 x 10 pixels em cada canto das imagens;
    4. Incluir 1 quadrado preto 15X15 no centro das imagens
2. **[OPERAÇÃO POR VIZINHANÇA]:**

    Utilizar kernel 3x3 pixels e desconsiderar pixels das extremidades. Para cada filtro implementar utilizando apenas numpy, utilizando pillow, utilizando opencv e utilizando scipy.

    1. Calcular o filtro da média;
    2. Calcular o filtro da mediana;
3. **[TRANSFORMAÇÕES GEOMÉTRICAS]:**

    Para cada filtro implementar utilizando apenas numpy, utilizando pillow, utilizando opencv e utilizando scipy.

    1. Escala: Redução em 1.5x e aumentar em 2.5x;
    2. Rotação em 45º, 90º e 100º;
    3. Translação utilizar os parâmetros que quiser nas coordenadas x e y;
    4. Translação em 35 pixel no eixo X, 45 eixo Y;

Respostas:

- **[OPERAÇÃO PONTO A PONTO]:**

```
from PIL import Image

import numpy as np

import matplotlib.pyplot as plt

from numpy import asarray



def main():



    im = np.array(Image.open('cameraman.tif'))

    im2 = np.array(Image.open('house.tif'))

    im3 = np.array(Image.open('lena_gray_512.tif'))



    #recebendo as imagens



    im_neg = im.copy()

    im2_neg = im2.copy()

    im3_neg = im3.copy()

    #copiando as imagens para fazer o negativo as copias



    im_neg =255-im_neg

    im2_neg =255-im2_neg

    im3_neg =255-im3_neg

    #gerando o negativo
```

```
    im_dark = im.copy()

    im_dark = im_dark/2

    im2_dark = im2.copy()

    im2_dark = im2_dark/2

    im3_dark = im3.copy()

    im3_dark = im3_dark/2

    #copiando as imagens para fazer o escurecimento e
escurecendo



    im_4qB = im.copy()

    im2_4qB = im2.copy()

    im3_4qB = im3.copy()

    #copiando as imagens para fazer os 4 quadrados brancos



    im_4qB[0:10,0:10] = 255

    im_4qB[0:10,-10:] = 255

    im_4qB[-10:,0:10] = 255

    im_4qB[-10:,-10:] = 255


    im2_4qB[0:10,0:10] = 255

    im2_4qB[0:10,-10:] = 255

    im2_4qB[-10:,0:10] = 255

    im2_4qB[-10:,-10:] = 255



    im3_4qB[0:10,0:10] = 255
```

```
im3_4qB[0:10,-10:] = 255

im3_4qB[-10:,0:10] = 255

im3_4qB[-10:,-10:] = 255

#fazendo os 4 quadrados brancos



im_qP = im.copy()

im2_qP = im2.copy()

im3_qP = im3.copy()

#copiando as imagens para fazer o quadrado preto



c1 = im_qP.shape[1]

c2 = im2_qP.shape[1]

c3 = im3_qP.shape[1]

l1 = im_qP.shape[0]

l2 = im2_qP.shape[0]

l3 = im3_qP.shape[0]

#setando as variaveis para o tamanho das imagens



l1 = l1//2

l2 = l2//2

l3 = l3//2

c1 = c1//2

c2 = c2//2

c3 = c3//2
```

```
    #setando as variaveis para representarem o meio das
imagens



    im_qP[l1-7:l1+7,c1-7:c1+7] = 0

    im2_qP[l2-7:l2+7,c2-7:c2+7] = 0

    im3_qP[l3-7:l3+7,c3-7:c3+7] = 0

    #fazendo o quadrado preto



    fig = plt.figure()

    plt1 = plt.subplot(5,3,1)

    plt2 = plt.subplot(5,3,2)

    plt3 = plt.subplot(5,3,3)

    plt4 = plt.subplot(5,3,4)

    plt5 = plt.subplot(5,3,5)

    plt6 = plt.subplot(5,3,6)

    plt7 = plt.subplot(5,3,7)

    plt8 = plt.subplot(5,3,8)

    plt9 = plt.subplot(5,3,9)

    plt10 = plt.subplot(5,3,10)

    plt11 = plt.subplot(5,3,11)

    plt12 = plt.subplot(5,3,12)

    plt13= plt.subplot(5,3,13)

    plt14 = plt.subplot(5,3,14)

    plt15 = plt.subplot(5,3,15)

    #criando o plot
```

```
plt1.title.set_text('original')

plt2.title.set_text('original')

plt3.title.set_text('original')


plt4.title.set_text('negativo')

plt5.title.set_text('negativo')

plt6.title.set_text('negativo')




plt7.title.set_text('Escurecido')

plt8.title.set_text('Escurecido')

plt9.title.set_text('Escurecido')


plt10.title.set_text('4 Q P')

plt11.title.set_text('4 Q P')

plt12.title.set_text('4 Q P')


plt13.title.set_text(' Q P')

plt14.title.set_text(' Q P')

plt15.title.set_text(' Q P')

#setando os titulos


plt1.imshow(im, cmap='gray')
```

```
    plt2.imshow(im2, cmap='gray')
    plt3.imshow(im3,cmap= 'gray')

    plt4.imshow(im_neg, cmap='gray', vmin=0, vmax=255)
    plt5.imshow(im2_neg, cmap='gray', vmin=0, vmax=255)
    plt6.imshow(im3_neg, cmap='gray', vmin=0, vmax=255)

    plt7.imshow(im_dark, cmap='gray', vmin=0, vmax=255)
    plt8.imshow(im2_dark, cmap='gray', vmin=0, vmax=255)
    plt9.imshow(im3_dark, cmap='gray', vmin=0, vmax=255)

    plt10.imshow(im_4qB, cmap='gray', vmin=0, vmax=255)
    plt11.imshow(im2_4qB, cmap='gray', vmin=0, vmax=255)
    plt12.imshow(im3_4qB,cmap='gray', vmin=0, vmax=255)

    plt13.imshow(im_qP, cmap='gray', vmin=0, vmax=255)
    plt14.imshow(im2_qP, cmap='gray', vmin=0, vmax=255)
    plt15.imshow(im3_qP,cmap='gray', vmin=0, vmax=255)
    #plotando as imagens
    plt.show()

if __name__ == "__main__":
    main()
```

- **[OPERAÇÃO POR VIZINHANÇA]:**
  - Numpy

```
from PIL import Image

import numpy as np

import matplotlib.pyplot as plt




def main():



    im = Image.open('lena_gray_512_salt_pepper.tif')

    im2 = Image.open('house.tif')

    im3 = Image.open('cameraman.tif')



    f_m2_nd = np.array(im2)

    f_m3_nd = np.array(im3)



    g_m2_nd = np.zeros(f_m2_nd.shape)#criar imagem de
zeros do tamanho de f_m_nd

    g_m3_nd = np.zeros(f_m3_nd.shape)#criar imagem de
zeros do tamanho de f_m_nd



    h_m2_nd = np.zeros(f_m2_nd.shape)#criar imagem de
zeros do tamanho de f_m_nd

    h_m3_nd = np.zeros(f_m3_nd.shape)#criar imagem de
zeros do tamanho de f_m_nd



    f_m_nd = np.array(im)
```

```
    g_m_nd = np.zeros(f_m_nd.shape)#criar imagem de
zeros do tamanho de f_m_nd

    h_m_nd = np.zeros(f_m_nd.shape)#criar imagem de
zeros do tamanho de f_m_nd

    #i_image_nd = np.zeros(f_m_nd.shape)#criar imagem
de zeros do tamanho de f_m_nd

    #plt.imshow(f_m_nd, cmap='gray')

    #im.show()



    #Operação em vizinhança

    l = f_m_nd.shape[0]#recupera as linhas da imagem

    c = f_m_nd.shape[1]#recupera as colunas da imagem

    k = 1#tamanho do kernel 1 = 3x3, 2 = 5x5, 3 = 7x7,
4 = 9x9, 5 = 11x11



    l2 = f_m2_nd.shape[0]#recupera as linhas da imagem

    c2 = f_m2_nd.shape[1]#recupera as colunas da
imagem



    l3 = f_m3_nd.shape[0]#recupera as linhas da imagem

    c3 = f_m3_nd.shape[1]#recupera as colunas da
imagem



    for x in range(k, l2-k):

        for y in range(k, c2-k):

            s_xy = f_m2_nd[x-k:x+k+1,
y-k:y+k+1]#recupera a vizinhança
```

```
            g_m2_nd[x,y] = np.mean(s_xy)#calcula a
média da vizinhança e atribui ao pixel central

            h_m2_nd[x,y] = np.median(s_xy)#calcula a
mediana da vizinhança e atribui ao pixel central

            #i_image_nd[x,y] = np.max(s_xy)#calcula o
maximo da vizinhança e atribui ao pixel central



    for x in range(k, l3-k):

        for y in range(k, c3-k):

            s_xy = f_m3_nd[x-k:x+k+1, y-k:y+k+1]

            g_m3_nd[x,y] = np.mean(s_xy)

            h_m3_nd[x,y] = np.median(s_xy)




    for x in range(k, l-k):

        for y in range(k, c-k):

            s_xy = f_m_nd[x-k:x+k+1,
y-k:y+k+1]#recupera a vizinhança

            g_m_nd[x,y] = np.mean(s_xy)#calcula a
média da vizinhança e atribui ao pixel central

            h_m_nd[x,y] = np.median(s_xy)#calcula a
mediana da vizinhança e atribui ao pixel central

            #i_image_nd[x,y] = np.max(s_xy)#calcula o
maximo da vizinhança e atribui ao pixel central



    #create two colums plot

    fig = plt.figure()

    plt1 = plt.subplot(3,3,1)
```

```
plt2 = plt.subplot(3,3,2)

plt3 = plt.subplot(3,3,3)

plt4 = plt.subplot(3,3,4)

plt5 = plt.subplot(3,3,5)

plt6 = plt.subplot(3,3,6)

plt7 = plt.subplot(3,3,7)

plt8 = plt.subplot(3,3,8)

plt9 = plt.subplot(3,3,9)




plt1.title.set_text('original')

plt2.title.set_text('mean filter')

plt3.title.set_text('median filter')

plt4.title.set_text('original')

plt5.title.set_text('mean filter')

plt6.title.set_text('median filter')

plt7.title.set_text('original')

plt8.title.set_text('mean filter')

plt9.title.set_text('median filter')



plt1.imshow(f_m_nd, cmap='gray')

plt2.imshow(g_m_nd, cmap='gray', vmin=0, vmax=255)

plt3.imshow(h_m_nd, cmap='gray', vmin=0, vmax=255)

plt4.imshow(f_m2_nd, cmap='gray')
```

```
    plt5.imshow(g_m2_nd, cmap='gray', vmin=0,
vmax=255)

    plt6.imshow(h_m2_nd, cmap='gray', vmin=0,
vmax=255)

    plt7.imshow(f_m3_nd, cmap='gray')

    plt8.imshow(g_m3_nd, cmap='gray', vmin=0,
vmax=255)

    plt9.imshow(h_m3_nd, cmap='gray', vmin=0,
vmax=255)



    plt.show()









if __name__ == "__main__":

    main()
```

- Opencv

```
from PIL import Image, ImageFilter

import matplotlib.pyplot as plt

import cv2




def main():

```

```
    im = cv2.imread('lena_gray_512_salt_pepper.tif')

    im2 = cv2.imread('house.tif')

    im3 = cv2.imread('cameraman.tif')



    imm = cv2.blur(im,(3,3))

    imme = cv2.medianBlur(im,3)



    imm2 = cv2.blur(im2,(3,3))

    imme2 = cv2.medianBlur(im2,3)



    imm3 = cv2.blur(im3,(3,3))

    imme3 = cv2.medianBlur(im3,3)









    #create two colums plot

    fig = plt.figure()

    plt1 = plt.subplot(3,3,1)

    plt2 = plt.subplot(3,3,2)

    plt3 = plt.subplot(3,3,3)

    plt4 = plt.subplot(3,3,4)

    plt5 = plt.subplot(3,3,5)
```

```
    plt6 = plt.subplot(3,3,6)
    plt7 = plt.subplot(3,3,7)
    plt8 = plt.subplot(3,3,8)
    plt9 = plt.subplot(3,3,9)



    plt1.title.set_text('original')
    plt2.title.set_text('mean filter')
    plt3.title.set_text('median filter')
    plt4.title.set_text('original')
    plt5.title.set_text('mean filter')
    plt6.title.set_text('median filter')
    plt7.title.set_text('original')
    plt8.title.set_text('mean filter')
    plt9.title.set_text('median filter')


    plt1.imshow(im, cmap='gray')
    plt2.imshow(imm, cmap='gray', vmin=0, vmax=255)
    plt3.imshow(imme, cmap='gray', vmin=0, vmax=255)
    plt4.imshow(im2, cmap='gray')
    plt5.imshow(imm2, cmap='gray', vmin=0, vmax=255)
    plt6.imshow(imme2, cmap='gray', vmin=0, vmax=255)
```

```
    plt7.imshow(im3, cmap='gray')
    plt8.imshow(imm3, cmap='gray', vmin=0, vmax=255)
    plt9.imshow(imme3, cmap='gray', vmin=0, vmax=255)

    plt.show()



if __name__ == "__main__":
    main()
```

- Pillow

```
from PIL import Image, ImageFilter
import matplotlib.pyplot as plt



def main():

    im = Image.open('lena_gray_512_salt_pepper.tif')
    im2 = Image.open('house.tif')
    im3 = Image.open('cameraman.tif')

    imm = im.filter(ImageFilter.BoxBlur(radius=3))
    imme = im.filter(ImageFilter.MedianFilter(size=3))
```

```
    imm2 = im2.filter(ImageFilter.BoxBlur(radius=3))

    imme2 =
im2.filter(ImageFilter.MedianFilter(size=3))



    imm3 = im3.filter(ImageFilter.BoxBlur(radius=3))

    imme3 =
im3.filter(ImageFilter.MedianFilter(size=3))



    #create two colums plot

    fig = plt.figure()

    plt1 = plt.subplot(3,3,1)

    plt2 = plt.subplot(3,3,2)

    plt3 = plt.subplot(3,3,3)

    plt4 = plt.subplot(3,3,4)

    plt5 = plt.subplot(3,3,5)

    plt6 = plt.subplot(3,3,6)

    plt7 = plt.subplot(3,3,7)

    plt8 = plt.subplot(3,3,8)

    plt9 = plt.subplot(3,3,9)




    plt1.title.set_text('original')

    plt2.title.set_text('mean filter')

    plt3.title.set_text('median filter')

    plt4.title.set_text('original')

    plt5.title.set_text('mean filter')
```

```
    plt6.title.set_text('median filter')
    plt7.title.set_text('original')
    plt8.title.set_text('mean filter')
    plt9.title.set_text('median filter')


    plt1.imshow(im, cmap='gray')
    plt2.imshow(imm, cmap='gray', vmin=0, vmax=255)
    plt3.imshow(imme, cmap='gray', vmin=0, vmax=255)
    plt4.imshow(im2, cmap='gray')
    plt5.imshow(imm2, cmap='gray', vmin=0, vmax=255)
    plt6.imshow(imme2, cmap='gray', vmin=0, vmax=255)
    plt7.imshow(im3, cmap='gray')
    plt8.imshow(imm3, cmap='gray', vmin=0, vmax=255)
    plt9.imshow(imme3, cmap='gray', vmin=0, vmax=255)


    plt.show()
if __name__ == "__main__":
    main()
```

- Scipy

```
import matplotlib.pyplot as plt
import imageio
import numpy as np
from imageio import imread
from scipy import ndimage
```

```
def main():

    im =
imageio.imread('lena_gray_512_salt_pepper.tif')
    im2 = imageio.imread('house.tif')
    im3 = imageio.imread('cameraman.tif')

    k = np.ones((3,3),np.float32)/9

    imm = ndimage.convolve(im, k)
    imme = ndimage.median_filter(im, size=3)

    imm2 = ndimage.convolve(im2, k)
    imme2 = ndimage.median_filter(im2, size=3)

    imm3 = ndimage.convolve(im3, k)
    imme3 = ndimage.median_filter(im3, size=3)

    #create two colums plot
    fig = plt.figure()
    plt1 = plt.subplot(3,3,1)
    plt2 = plt.subplot(3,3,2)
    plt3 = plt.subplot(3,3,3)
```

```
plt4 = plt.subplot(3,3,4)

plt5 = plt.subplot(3,3,5)

plt6 = plt.subplot(3,3,6)

plt7 = plt.subplot(3,3,7)

plt8 = plt.subplot(3,3,8)

plt9 = plt.subplot(3,3,9)




plt1.title.set_text('original')

plt2.title.set_text('mean filter')

plt3.title.set_text('median filter')

plt4.title.set_text('original')

plt5.title.set_text('mean filter')

plt6.title.set_text('median filter')

plt7.title.set_text('original')

plt8.title.set_text('mean filter')

plt9.title.set_text('median filter')


plt1.imshow(im, cmap='gray')

plt2.imshow(imm, cmap='gray', vmin=0, vmax=255)

plt3.imshow(imme, cmap='gray', vmin=0, vmax=255)

plt4.imshow(im2, cmap='gray')

plt5.imshow(imm2, cmap='gray', vmin=0, vmax=255)

plt6.imshow(imme2, cmap='gray', vmin=0, vmax=255)

plt7.imshow(im3, cmap='gray')
```

```
    plt8.imshow(imm3, cmap='gray', vmin=0, vmax=255)
    plt9.imshow(imme3, cmap='gray', vmin=0, vmax=255)


    plt.show()
if __name__ == "__main__":
    main()
```

- **[TRANSFORMAÇÕES GEOMÉTRICAS]:**

  Serão apresentadas apenas as funções de alteração, pois os códigos são cumpridos, e estão no git hub

  - Numpy

```
def rotate(image, angle):
    an = angle * np.pi / 180


    # Calculando a matriz de transformação de rotação
    cos_angle = np.cos(an)
    sin_angle = np.sin(an)
    rotation_matrix = np.array([[cos_angle,
-sin_angle], [sin_angle, cos_angle]])


    # Calculando as coordenadas dos pixels
rotacionados
    x, y = np.meshgrid(range(image.shape[1]),
range(image.shape[0]))
    coordinates = np.vstack([x.ravel(), y.ravel()])
```

```
    rotated_coordinates = np.dot(rotation_matrix,
coordinates).astype(int)



    # Criando um array para armazenar a imagem
rotacionada

    imRotate = np.zeros_like(image)



    # Copiando os pixels da imagem original para a
imagem rotacionada

    for i in range(coordinates.shape[1]):

        x, y = coordinates[:, i]

        new_x, new_y = rotated_coordinates[:, i]

        if 0 <= new_x < imRotate.shape[1] and 0 <=
new_y < imRotate.shape[0]:

            imRotate[new_y, new_x] = image[y, x]



    return imRotate



def tamanho(imagem, fator):



    # Calculando o novo tamanho da imagem

    new_shape = tuple(int(dim * fator) for dim in
imagem.shape)



    # Criando um array para armazenar a imagem
reduzida

    image_alterada = np.zeros(new_shape)
```

```
    # Selecionando uma submatriz da imagem
    x_ratio = imagem.shape[0] / new_shape[0]
    y_ratio = imagem.shape[1] / new_shape[1]
    for i in range(new_shape[0]):
        for j in range(new_shape[1]):
            image_alterada[i, j] = imagem[int(i * x_ratio), int(j * y_ratio)]

    return image_alterada

def translacao(image, x, y):

    # Definindo o deslocamento da translação
    x_trava = x
    y_trava = y

    # Criando um array para armazenar a imagem transladada
    imTras = np.zeros_like(image)

    # Copiando os pixels da imagem original para a imagem transladada
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            new_i = i + y_trava
            new_j = j + x_trava
```

```
            if 0 <= new_i < image.shape[0] and 0 <=
new_j < image.shape[1]:

                imTras[new_i, new_j] = image[i, j]



    return imTras
```

- Opencv

```
def reduzir (im, scale):

    width = int(im.shape[1] * scale / 100)

    height = int(im.shape[0] * scale / 100)

    dim = (width, height)

    imRed = cv2.resize(im, dim, interpolation =
cv2.INTER_AREA)



    return imRed



def aumentar (im, scale):

    width = int(im.shape[1] * scale / 100)

    height = int(im.shape[0] * scale / 100)

    dim = (width, height)

    imAum = cv2.resize(im, dim, interpolation =
cv2.INTER_CUBIC)



    return imAum

def rotaciona (im, angle):
```

```
    rows, cols = im.shape[:2]

    M =
cv2.getRotationMatrix2D((cols/2,rows/2),angle,1)

    imRot = cv2.warpAffine(im,M,(cols,rows))



    return imRot



def translacao (im, x, y):

    M = np.float32([[1,0,x],[0,1,y]])

    imTras =
cv2.warpAffine(im,M,(im.shape[1],im.shape[0]))



    return imTras
```

- Pillow

```
def reduzir (im, scale):

    width = int(im.width / scale)

    height = int(im.height / scale )



    imRed = im.resize((width, height), resample =
Image.LANCZOS)



    return imRed
```

```
def aumentar (im, scale):

    width = int(im.width * scale)

    height = int(im.height * scale)



    imAum = im.resize((width, height), resample =
Image.LANCZOS)



    return imAum
```

```
imRot90 = im.rotate(90)

    im2Rot90 = im2.rotate(90)

    im3Rot90 = im3.rotate(90)



    imRot45 = im.rotate(45)

    im2Rot45 = im2.rotate(45)

    im3Rot45 = im3.rotate(45)



    imRot100= im.rotate(100)

    im2Rot100 = im2.rotate(100)

    im3Rot100 = im3.rotate(100)



    imTras = ImageChops.offset(im, 50, 50)

    im2Tras = ImageChops.offset(im2, 50, 50)

    im3Tras = ImageChops.offset(im3, 50, 50)



    imTras3545 = ImageChops.offset(im, 35, 45)
```

```
    im2Tras3545 = ImageChops.offset(im2, 35, 45)

    im3Tras3545 = ImageChops.offset(im3, 35, 45)
```

- Scipy

```
def reduzir(im, scale):



    scale = 1/ scale

    imr = interpolation.zoom(im,zoom = (scale, scale, 1))



    return imr




def aumentar(im, scale):



    imr = interpolation.zoom(im,zoom = (scale, scale, 1))



    return imr



def translacao(im, x, y):

    deslocamento = [x,y]

    tmatrix = np.eye(3)

    tmatrix[0,2] = deslocamento[0]

    tmatrix[1,2] = deslocamento[1]
```

```
    imT = ndimage.affine_transform(im, tmatrix)



    return imT

imRot90 = ndimage.rotate(im,90)

    im2Rot90 = ndimage.rotate(im2,90)

    im3Rot90 = ndimage.rotate(im3,90)



    imRot45 = ndimage.rotate(im,45)

    im2Rot45 = ndimage.rotate(im2,45)

    im3Rot45 = ndimage.rotate(im3,45)



    imRot100= ndimage.rotate(im,100)

    im2Rot100 = ndimage.rotate(im2,100)

    im3Rot100 = ndimage.rotate(im3,100)
```

Conclusão:

Cada uma das ferramentas apresenta particularidades, todavia, Opencv, Pillow e Scipy apresentam maior quantidade de funções prontas, facilitando sua utilização

Link Git:

https://github.com/Reneress/PDI.git