import numpy as np

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

*# leitura da imagem*

imagemOrignal = mpimg.imread('../image\_(1).jpg')

*# Como proceder nesse caso? são postos 3 coisinhas pra cada original pixel, por isso o*

*# np array reclama que tem uma sequancia na lista sendo que não devia ter, caso tenha*

*# não tem como aplicar o uint8, mas como sempre são iguais acredito que funciona como esta*

def **noise**(imagemOriginal, prob1=0.10 ,prob2=0.10 , option1=0, option2=255):

imagemReturn = []

*# loop percorre conjuntos de pixels nomeando-os como line*

for line in imagemOriginal :

imagemReturn += [[]]

*# loop percorre pixels em line*

for originalPixel in line:

*# cada pixel original tem uma analise probabilistica se deve ou não ser substituido por ruido*

option = choose(prob1, prob2, option1, option2)

*# caso a opção em questão seja -1 então o pixel deve se manter como originalmente informado*

if(option<0):

pixel = originalPixel[0]

else:

*# caso contratio deve ser substituido pelo valor de cor atribuido a option*

pixel = option

*# adiciona o pixel lido ao fim da lista*

pos = (imagemReturn.**\_\_len\_\_**()-1)

imagemReturn[pos] = imagemReturn[pos] + [pixel]

return np.array(imagemReturn, dtype=np.uint8)

def **gaussianNoise**(imagemOriginal, media = 10, desvio = 30):

imagemReturn = []

*# loop percorre conjuntos de pixels nomeando-os como line*

for line in imagemOriginal :

imagemReturn += [[]]

*# loop percorre pixels em line*

for originalPixel in line:

pixel = gaussianAux(originalPixel[0], media, desvio)

*# adiciona o pixel lido ao fim da lista*

pos = (imagemReturn.**\_\_len\_\_**()-1)

**print**(pixel)

imagemReturn[pos] = imagemReturn[pos] + [pixel]

return np.array(imagemReturn, dtype=np.uint8)

*# retorna probabilidades*

def **prob**(prob1,prob2):

return [prob1, prob2, 1-(prob1+prob2)]

*# retorna opções*

def **options**(option1,option2):

return [option1, option2, -1]

*# retorna a opção selecionada de acordo com a probabilidade*

def **choose**(prob1,prob2, option1, option2):

return np.random.choice(options(option1,option2), p = prob(prob1, prob2))

*# retorna o valor retirado da gaussiana*

def **gaussianAux**(z, media, desvio):

*# (e^(-((Z-Zm)^2)/(2D^2)))/(D\*((2pi)^0.5)))*

return np.exp(-((z-media)\*\*2)/(2\*(desvio\*\*2)))/(((2\*np.pi\*desvio)\*\*0.5))

*# ploting*

*# (fig, (ax1, ax2, ax3)) = plt.subplots(1,3)*

*# ax1.imshow(imagemOrignal, cmap='gray')*

*# ax2.imshow(gaussianNoise(imagemOrignal), cmap='gray')*

*# ax3.imshow(noise(imagemOrignal), cmap='gray')*

*# plt.show()*

*# saving*

*# plt.imshow(noise(imagemOrignal), cmap='gray')*

plt.imsave('./questao1/Resultado\_(1) - noise.jpg', gaussianNoise(imagemOrignal), cmap='gray')