import cv2

import matplotlib.pyplot as plt

import numpy as np

from scipy.signal import convolve2d as conv2

from skimage import color, data, restoration

from math import sqrt

from sklearn.metrics import mean\_squared\_error

from skimage.morphology import disk, square, diamond

from skimage.filters import median

path = "C:/Users/DELL 3468/Desktop/TY Shit/Python/Image Restoration Tool/"

imgpath1 = path + "KAS McLeod Damaged.jpg"

imgpath2 = path + "Girl 1.png"

imgpath3 = path + "Girl 2.png"

img1 = cv2.imread(imgpath1, 1)

img1 = cv2.cvtColor(img1, cv2.COLOR\_BGR2RGB)

img2 = cv2.imread(imgpath2, 1)

img2 = cv2.cvtColor(img2, cv2.COLOR\_BGR2RGB)

img3 = cv2.imread(imgpath3, 1)

img3 = cv2.cvtColor(img3, cv2.COLOR\_BGR2RGB)

caller = getattr(data, 'rocket')

img4 = caller()

img4 = color.rgb2gray(img4)

output = [img1, img2, img3, img4]

titles = ['1. McLeod', '2. Girl 1', '3. Girl 2', '4. Rocket']

print("Welcome. This is our Image Restoration Tool")

print("MENU")

print('Which image would you want restored? ')

c = input()

for i in range(4):

plt.subplot(2, 2, i + 1)

#if i == 1 or i == 4:

plt.imshow(output[i], cmap='gray')

#else:

#plt.imshow(output[i])

plt.title(titles[i])

plt.xticks([])

plt.yticks([])

plt.show()

if c == '1':

img = img1

height = np.size(img, 0)

width = np.size(img, 1)

createdMask = np.zeros([width, height])

grayimg = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# temp = np.asarray(PIL.Image.open('test.jpg'))

# for j in grayimg:

# iii = np.asarray([[i[0], i[1]] for i in j])

for i in range(1, width - 1):

for j in range(1, height - 1):

# ii = iii[i,j]

if grayimg[i, j] > 0:

createdMask[i, j] = 0

else:

createdMask[i, j] = 255

gb\_kernel = cv2.getGaborKernel((8, 8), 1.2, np.pi, 28.0, 0.0, 0, ktype=cv2.CV\_32F)

gcreatedMask = cv2.filter2D(createdMask, cv2.CV\_8U, gb\_kernel.transpose())

# saves the merged image to a file

# cv2.imwrite("rgb.jpg", img)

output1 = cv2.inpaint(img, gcreatedMask, 5, cv2.INPAINT\_TELEA)

output2 = cv2.inpaint(img, gcreatedMask, 5, cv2.INPAINT\_NS)

output = [img, gcreatedMask, output1, output2]

titles = ['Damaged Image', 'Mask', 'TELEA', 'NS']

for i in range(4):

plt.subplot(2, 2, i + 1)

if i == 1:

plt.imshow(output[i], cmap='gray')

else:

plt.imshow(output[i])

plt.title(titles[i])

plt.xticks([])

plt.yticks([])

plt.show()

elif c == '2':

img = img2

denoised\_gray = cv2.fastNlMeansDenoising(img, None, 5, 13)

height = np.size(img, 0)

width = np.size(img, 1)

mask = np.zeros([width, height])

th, mask = cv2.threshold(denoised\_gray, 250, 255, cv2.THRESH\_BINARY)

kernel = cv2.getStructuringElement(cv2.MORPH\_RECT, (6, 6))

morph\_mask = cv2.dilate(mask, kernel, None, (2, 2), iterations=4)

result = cv2.inpaint(denoised\_gray, morph\_mask, 7, cv2.INPAINT\_TELEA)

th, mask = cv2.threshold(denoised\_gray, 210, 255, cv2.THRESH\_BINARY)

kernel = cv2.getStructuringElement(cv2.MORPH\_RECT, (6, 6))

morph\_mask = cv2.dilate(mask, kernel, None, (2, 2), iterations=4)

result = cv2.inpaint(denoised\_gray, morph\_mask, 7, cv2.INPAINT\_TELEA)

result = cv2.medianBlur(result, 3, None)

output = [img, result]

titles = ['Damaged Image', 'Result']

for i in range(2):

plt.subplot(1, 2, i + 1)

plt.imshow(output[i], cmap='gray')

plt.title(titles[i])

plt.xticks([])

plt.yticks([])

plt.show()

elif c == '3':

print('ALL 3')

elif c == '4':

img = img4

psf = np.ones((3, 3)) / 9

img = conv2(img, psf, 'same')

# Add Noise to Image

img\_noisy = img.copy()

img\_noisy += (np.random.poisson(lam=25, size=img.shape) - 20) / 255

# Restore Image using Richardson-Lucy algorithm

deconvolved\_RL = restoration.richardson\_lucy(img\_noisy, psf, iterations=30)

fig, ax = plt.subplots(nrows=1, ncols=4, figsize=(8, 5))

plt.gray()

for a in (ax[0], ax[1], ax[2], ax[3]):

a.axis('off')

result = median(deconvolved\_RL, disk(3))

#ax[0].imshow(img)

#ax[0].set\_title('Original Data')

ax[0].imshow(img\_noisy)

ax[0].set\_title('Noisy data')

ax[1].imshow(deconvolved\_RL)

ax[1].set\_title('Restoration using\nRichardson-Lucy')

ax[2].imshow(result)

ax[2].set\_title('Median on RL')

fig.subplots\_adjust(wspace=0.02, hspace=0.2, top=0.9, bottom=0.05, left=0, right=1)

plt.show()