Filtering_NIM.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

1. Mean Filter

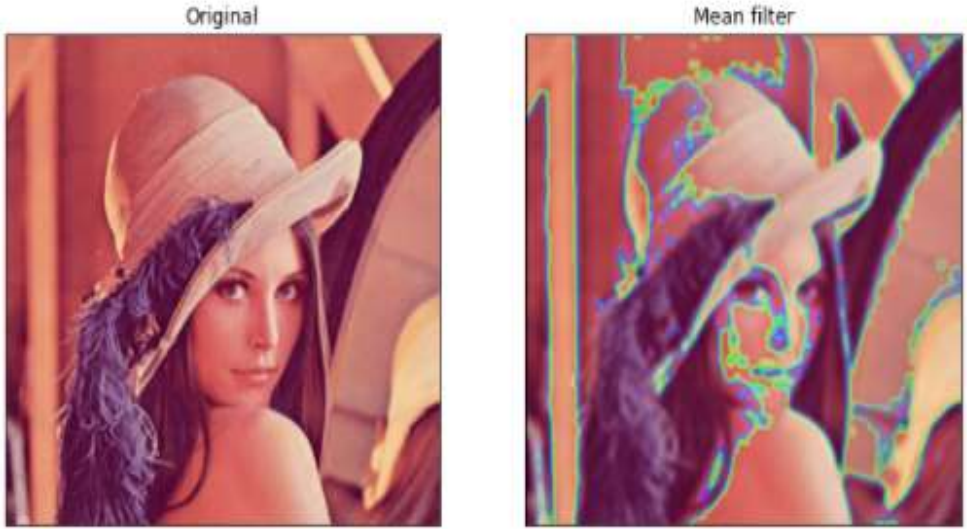
[1]
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from skimage import data, color
from skimage.io import imread, imshow
from matplotlib import pyplot as plt
from PIL import Image, ImageFilter
from google.colab import files
uploaded = files.upload()

```

Choose Files lenna.JPG

- lenna.JPG(image/jpeg) - 51779 bytes, last modified: 4/6/2021 - 100% done

```
Saving lenna.JPG to lenna.JPG
```

[2]
```
%matplotlib inline
image = cv2.imread('lenna.JPG') # reads the image
image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV) # convert to HSV
figure_size = 9 # the dimension of the x and y axis of the kernal.
new_image = cv2.blur(image,(figure_size, figure_size))
plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(cv2.cvtColor(image, cv2.COLOR_HSV2RGB)),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_HSV2RGB)),plt.title('Mean filter')
plt.xticks([]), plt.yticks([])
plt.show()
```



[3]
```
# The image will first be converted to grayscale
image2 = cv2.cvtColor(image, cv2.COLOR_HSV2BGR)
image2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)
figure_size = 9
new_image = cv2.blur(image2,(figure_size, figure_size))
plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(image2, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(new_image, cmap='gray'),plt.title('Mean filter')
plt.xticks([]), plt.yticks([])
plt.show()
```

2. Gaussian Filter

```
[4]  1 new_image = cv2.GaussianBlur(image, (figure_size, figure_size),0)
     2 plt.figure(figsize=(11,6))
     3 plt.subplot(121), plt.imshow(cv2.cvtColor(image, cv2.COLOR_HSV2RGB)),plt.title('Original')
     4 plt.xticks([]), plt.yticks([])
     5 plt.subplot(122), plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_HSV2RGB)),plt.title('Gaussian Filter')
     6 plt.xticks([]), plt.yticks([])
     7 plt.show()
```


```
[5]  1 new_image_gauss = cv2.GaussianBlur(image2, (figure_size, figure_size),0)
     2 plt.figure(figsize=(11,6))
     3 plt.subplot(121), plt.imshow(image2, cmap='gray'),plt.title('Original')
     4 plt.xticks([]), plt.yticks([])
     5 plt.subplot(122), plt.imshow(new_image_gauss, cmap='gray'),plt.title('Gaussian Filter')
     6 plt.xticks([]), plt.yticks([])
     7 plt.show()
```

3. Median Filter

[6]
```
new_image = cv2.medianBlur(image, figure_size)
plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(cv2.cvtColor(image, cv2.COLOR_HSV2RGB)),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_HSV2RGB)),plt.title('Median Filter')
plt.xticks([]), plt.yticks([])
plt.show()
```

[7]
```
new_image = cv2.medianBlur(image2, figure_size)
plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(image2, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(new_image, cmap='gray'),plt.title('Median Filter')
plt.xticks([]), plt.yticks([])
plt.show()
```

4. Laplacian Filter

```
[9]  1 new_image = cv2.Laplacian(image2,cv2.CV_64F)
     2 plt.figure(figsize=(11,6))
     3 plt.subplot(131), plt.imshow(image2, cmap='gray'),plt.title('Original')
     4 plt.xticks([]), plt.yticks([])
     5 plt.subplot(132), plt.imshow(new_image, cmap='gray'),plt.title('Laplacian')
     6 plt.xticks([]), plt.yticks([])
     7 plt.subplot(133), plt.imshow(image2 + new_image, cmap='gray'),plt.title('Resulting image')
     8 plt.xticks([]), plt.yticks([])
     9 plt.show()
```

5.Frequency Filter

```
[10]  1 dft = cv2.dft(np.float32(image2),flags = cv2.DFT_COMPLEX_OUTPUT)
      2 # shift the zero-frequncy component to the center of the spectrum
      3 dft_shift = np.fft.fftshift(dft)
      4 # save image of the image in the fourier domain.
      5 magnitude_spectrum = 20*np.log(cv2.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))
      6 # plot both images
      7 plt.figure(figsize=(11,6))
      8 plt.subplot(121),plt.imshow(image2, cmap = 'gray')
      9 plt.title('Input Image'), plt.xticks([]), plt.yticks([])
     10 plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
     11 plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
     12 plt.show()
```

[14]
```
rows, cols = image2.shape
crow,ccol = rows//2 , cols//2
# create a mask first, center square is 1, remaining all zeros
mask = np.zeros((rows,cols,2),np.uint8)
mask[crow-30:crow+30, ccol-30:ccol+30] = 1
# apply mask and inverse DFT
fshift = dft_shift*mask
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:,:,0],img_back[:,:,1])
# plot both images
plt.figure(figsize=(11,6))
plt.subplot(121),plt.imshow(image2, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
plt.title('Low Pass Filter'), plt.xticks([]), plt.yticks([])
plt.show()
```