

```

# 1. Реализуйте высокочастотную фильтрацию на основе ядра Гаусса

import numpy as np
import cv2 as cv
#from random import randint
from matplotlib import pyplot as plt

image = cv.imread('periodic_noise.jpg', cv.IMREAD_GRAYSCALE)

dft = cv.dft(np.float32(image), flags=cv.DFT_COMPLEX_OUTPUT)      #
#преобразование Фурье
dft_shift = np.fft.fftshift(dft)                                  #
#сдвиг нулевых частот в центр массива

# фильтр высоких частот = 1 - низкочастотный фильтр Гаусса

sigma = 30                                                         #
"сигма" : в результирующем изображении отображает сглаживание

# получение сетки x и y изображения
rows, cols = image.shape
x, y = np.meshgrid(np.arange(0, cols, 1), np.arange(0, rows, 1))

gauss_mask = np.exp(-((x - (cols // 2))**2 + (y - (rows // 2))**2) /
(2 * sigma**2))

mask_res = 1 - gauss_mask

print(gauss_mask)

[[2.83635701e-59 4.38693848e-59 6.77765789e-59 ... 6.77765789e-59
 4.38693848e-59 2.83635701e-59]
 [3.94309099e-59 6.09870249e-59 9.42227006e-59 ... 9.42227006e-59
 6.09870249e-59 3.94309099e-59]
 [5.47558032e-59 8.46897406e-59 1.30842521e-58 ... 1.30842521e-58
 8.46897406e-59 5.47558032e-59]
 ...
 [7.59523052e-59 1.17473960e-58 1.81492928e-58 ... 1.81492928e-58
 1.17473960e-58 7.59523052e-59]
 [5.47558032e-59 8.46897406e-59 1.30842521e-58 ... 1.30842521e-58
 8.46897406e-59 5.47558032e-59]
 [3.94309099e-59 6.09870249e-59 9.42227006e-59 ... 9.42227006e-59
 6.09870249e-59 3.94309099e-59]]

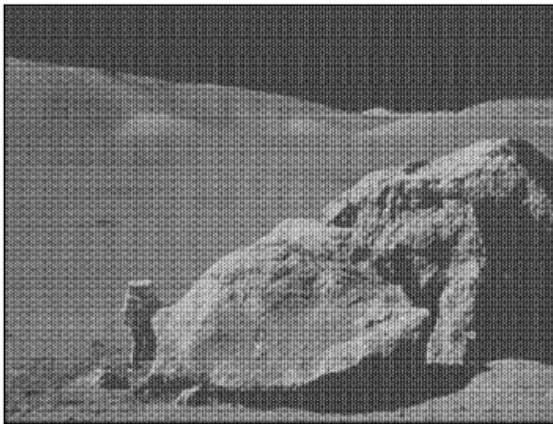
dft_shift = dft_shift * mask_res[:, :, np.newaxis]                #
#применение фильтра (добавить размерность)

image_idft = cv.idft(np.fft.ifftshift(dft_shift))                  #
#обратный сдвиг нулевых частот + обратное преобразование Фурье
image_idft = cv.magnitude(image_idft[:, :, 0], image_idft[:, :, 1])

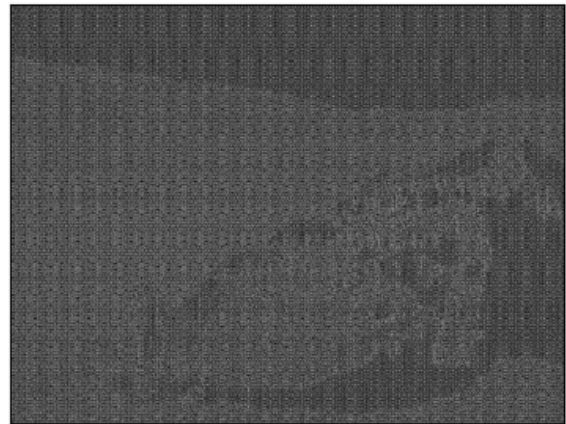
```

```
plt.figure(figsize = (10, 5))
plt.subplot(121),plt.imshow(image, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(image_idft, cmap = 'gray')
plt.title('Filtered image'), plt.xticks([]), plt.yticks([])
plt.show()
```

Input Image



Filtered image



*# 2. Реализуйте удаление периодического шума*

```
from skimage.feature import peak_local_max

dft = cv.dft(np.float32(image), flags=cv.DFT_COMPLEX_OUTPUT) #
# преобразование Фурье
dft_shift = np.fft.fftshift(dft) #
# сдвиг нулевых частот в центр массива

magnitude_spectrum =
20*np.log(cv.magnitude(dft_shift[:, :, 0], dft_shift[:, :, 1]))
local_maxum = peak_local_max(magnitude_spectrum, min_distance=15,
threshold_abs=200)

rows, cols = image.shape
mask = np.copy(dft_shift)
mask[:, :, :] = 1

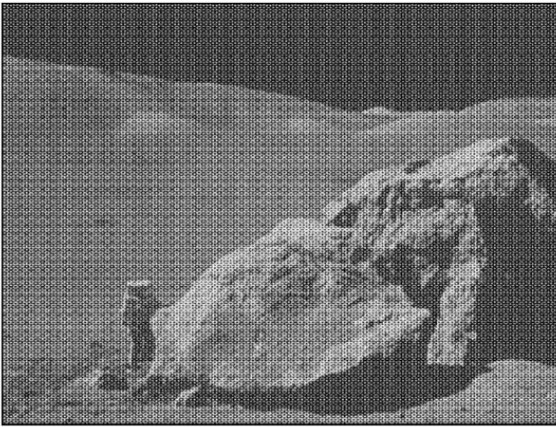
for i in local_maxum:
    r = np.int32(((i[0]-rows//2)**2+(i[1]-cols//2)**2)**(1/2))
    cv.circle(mask, (rows//2, cols//2), r, (0,0))

dft_shift = dft_shift * mask # применение фильтра
# (добавить размерность)

image_idft = cv.idft(np.fft.ifftshift(dft_shift)) #
# обратный сдвиг нулевых частот + обратное преобразование Фурье
image_idft = cv.magnitude(image_idft[:, :, 0], image_idft[:, :, 1])
```

```
plt.figure(figsize = (15, 8))
plt.subplot(121),plt.imshow(image, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(image_idft, cmap = 'gray')
plt.title('Filtered image'), plt.xticks([]), plt.yticks([])
plt.show()
```

Input Image



Filtered image

