Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение  
высшего образования

национальный исследовательский университет итмо

Лабораторная работа №4

«Фильтрация изображений в частотной области»

Выполнил:

Нехорошкин М. Д.

Группа:

Q4110

Проверил:

Лашманов О.Ю.

Санкт-Петербург

2022

**1) Подготовить свой портрет в цифровом формате (полутоновое изображение с разрешением не более 800×800). Получить изображение фурье-спектра.**

Необходимо подгрузить:

import numpy as np  
import cv2  
import matplotlib.pyplot as plt  
import os  
from PIL import Image  
from numpy.fft import fft2, fftshift  
from numpy.fft import ifft2, ifftshift

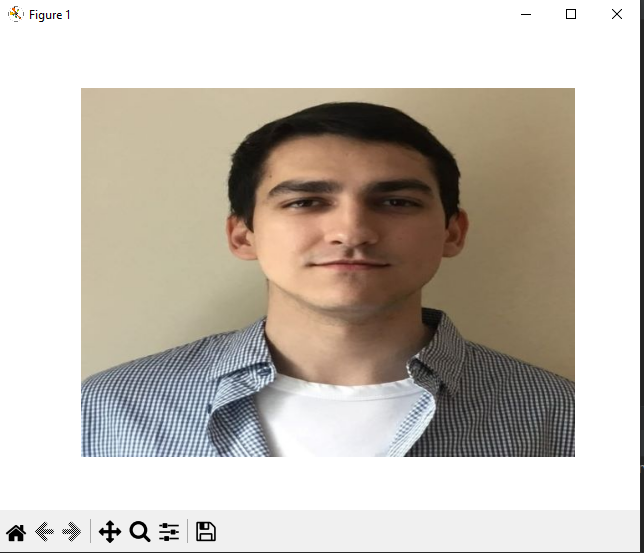
Функция для перевода из RGB в полутоновое:

def rgb\_to\_gray(img):  
 (W, H) = img.size  
 grey\_img = np.array([ [0]\*W for i in range(H) ])  
 pix = img.load()  
 for i in range(W):  
 for j in range(H):  
 grey\_img[j][i] = pix[i, j][0]\*0.3 + pix[i, j][1]\*0.59 + pix[i, j][2]\*0.11  
 return grey\_img

Теперь произведем требуемые операции:

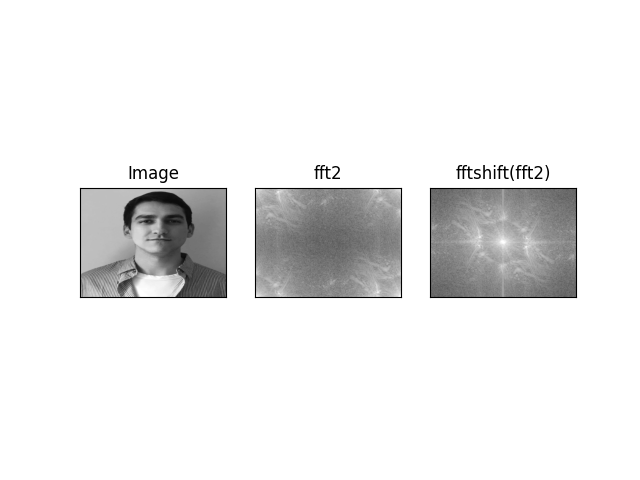
img\_dir = 'D:\\Programms\\PyCharm\\PyCharm\\PyCharm\_projects\\Theme\_7\\image.jpg' # путь к фото  
save\_path = 'D:\\Programms\\PyCharm\\PyCharm\\PyCharm\_projects\\Theme\_7\\DIP\\Lab2\\'  
with Image.open(img\_dir) as img: # открытие файла  
 img.load() #считывание данных  
plt.imshow(img)  
plt.axis('off')  
plt.show()  
  
img\_gray = rgb\_to\_gray(img)  
plt.imshow(img\_gray, cmap='gray')  
plt.axis('off')  
plt.savefig(save\_path + 'img\_gray')  
plt.show()  
  
F = fft2(img\_gray)  
F\_shift = fftshift(F)  
ampl\_spectr = (20\*np.log(1+np.abs(F\_shift))).clip(0,255)  
ifft\_shift = ifftshift(F\_shift)  
fig = plt.figure()  
plt.subplot(131),plt.imshow(img\_gray, cmap = 'gray')  
plt.title('Image'), plt.xticks([]), plt.yticks([])  
plt.subplot(132),plt.imshow((20\*np.log(1+np.abs(F))).clip(0,255), cmap = 'gray')  
plt.title('fft2'), plt.xticks([]), plt.yticks([])  
plt.subplot(133),plt.imshow(ampl\_spectr, cmap = 'gray')  
plt.title('fftshift(fft2)'), plt.xticks([]), plt.yticks([])  
plt.show()

Результат:



Изображение выглядит как текст, в позе, костюм

Автоматически созданное описание



**2) Провести низкочастотную фильтрацию изображения с использованием: идеального фильтра низких частот для D0 = 5, 10, 50, 250; фильтра Баттерворта низких частот для n = 2; и D0 = 5, 10, 50, 250; фильтра Гауса низких частот для D0 = 5, 10, 50, 250.**

Определим функции:

def LowPassFilter(radius, img\_gray):  
 rows, cols = img\_gray.shape  
 crow,ccol = rows//2 , cols//2  
 mask = np.zeros((rows,cols),np.uint8)  
 for i in range(rows):  
 for j in range(cols):  
 if ((i-crow)\*\*2 + (j-ccol)\*\*2)\*\*0.5 <= radius:  
 mask[i,j] = 1  
 return mask

def Butterworth\_Low(radius, img\_gray, n):  
 rows, cols = img\_gray.shape  
 crow,ccol = rows//2 , cols//2  
 mask = np.ones(img\_gray.shape)  
 for i in range(rows):  
 for j in range(cols):  
 D = ((i-crow)\*\*2 + (j-ccol)\*\*2)\*\*0.5  
 mask[i,j] = 1 / (1 + (D/radius)\*\*(2\*n))  
 return mask

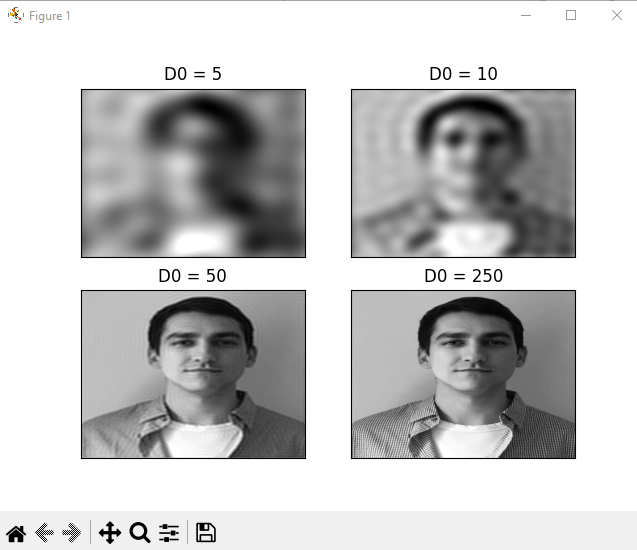
def Gauss\_Low(radius, img\_gray):  
 rows, cols = img\_gray.shape  
 crow,ccol = rows//2 , cols//2  
 mask = np.ones(img\_gray.shape)  
 for i in range(rows):  
 for j in range(cols):  
 D = ((i-crow)\*\*2 + (j-ccol)\*\*2)\*\*0.5  
 mask[i,j] = np.exp((-(D)\*\*2)/(2\*radius\*\*2))  
 return mask

Основной код:

n = 2  
D\_0 = [5, 10, 50, 250]  
res = []  
res1 = []  
res2 = []  
for i in D\_0:  
 maska = LowPassFilter(i, img\_gray)  
 res.append(abs(ifft2(ifftshift(maska \* F\_shift))).clip(0,255))  
 maska1 = Butterworth\_Low(i, img\_gray, n)  
 res1.append(abs(ifft2(ifftshift(maska1 \* F\_shift))).clip(0,255))  
 maska2 = Gauss\_Low(i, img\_gray)  
 res2.append(abs(ifft2(ifftshift(maska2 \* F\_shift))).clip(0, 255))  
res\_all = [res, res1, res2]  
for j in range (3):  
 fig = plt.figure()  
 for i in range(4):  
 plt.subplot(2,2,(i+1)), plt.imshow(res\_all[j][i], cmap='gray')  
 plt.title(f'D0 = {D\_0[i]}'), plt.xticks([]), plt.yticks([])  
 plt.show()

Результат:

Идеальный фильтр низких частот

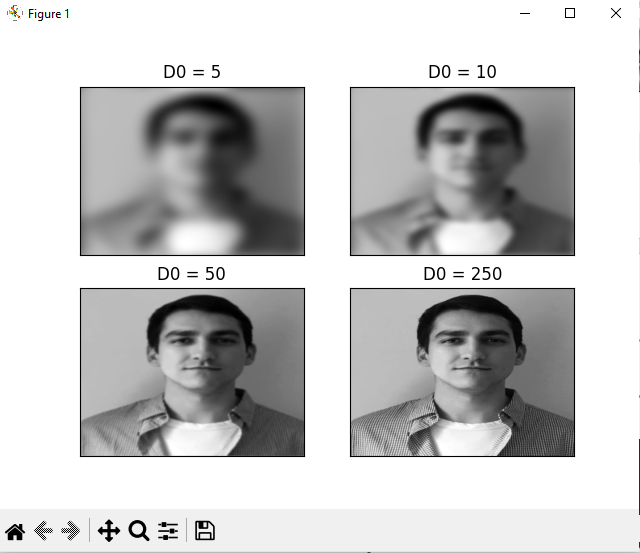


Фильтр Баттерворта низких частот

Изображение выглядит как текст, другой, снимок экрана

Автоматически созданное описание

Фильтр Гауса низких частот



**3) Провести высокочастотную фильтрацию изображения с использованием: идеального фильтра верхних частот для D0 = 5, 10, 50, 250; фильтра Баттерворта верхних частот для n = 2; и D0 = 5, 10, 50, 250; фильтра Гауса верхних частот для D0 = 5, 10, 50, 250.**

Определим функции:

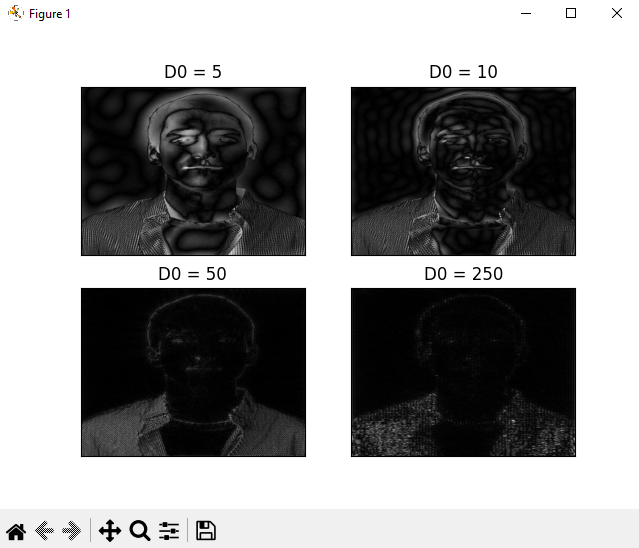
def HighPassFilter(radius, img\_gray):  
 rows, cols = img\_gray.shape  
 crow,ccol = rows//2 , cols//2  
 mask = np.ones((rows,cols),np.uint8)  
 for i in range(rows):  
 for j in range(cols):  
 if ((i-crow)\*\*2 + (j-ccol)\*\*2)\*\*0.5 <= radius:  
 mask[i,j] = 0  
 return mask  
def Butterworth\_High(radius, img\_gray, n):  
 rows, cols = img\_gray.shape  
 crow,ccol = rows//2 , cols//2  
 mask = np.ones(img\_gray.shape)  
 for i in range(rows):  
 for j in range(cols):  
 D = ((i-crow)\*\*2 + (j-ccol)\*\*2)\*\*0.5  
 if D == 0:  
 D = 10\*\*(-10)  
 mask[i,j] = 1 / (1 + (radius/D)\*\*(2\*n))  
 return mask  
def Gauss\_High(radius, img\_gray):  
 rows, cols = img\_gray.shape  
 crow,ccol = rows//2 , cols//2  
 mask = np.ones(img\_gray.shape)  
 for i in range(rows):  
 for j in range(cols):  
 D = ((i-crow)\*\*2 + (j-ccol)\*\*2)\*\*0.5  
 mask[i,j] = 1-np.exp((-(D)\*\*2)/(2\*radius\*\*2))  
 return mask

Основной код:

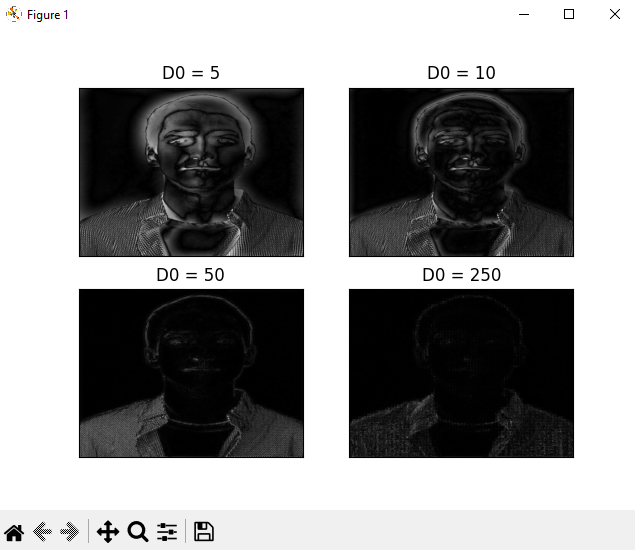
n = 2  
D\_0 = [5, 10, 50, 250]  
res = []  
res1 = []  
res2 = []  
for i in D\_0:  
 maska = HighPassFilter(i, img\_gray)  
 res.append(abs(ifft2(ifftshift(maska \* F\_shift))).clip(0,255))  
 maska1 = Butterworth\_High(i, img\_gray, n)  
 res1.append(abs(ifft2(ifftshift(maska1 \* F\_shift))).clip(0,255))  
 maska2 = Gauss\_High(i, img\_gray)  
 res2.append(abs(ifft2(ifftshift(maska2 \* F\_shift))).clip(0, 255))  
res\_all = [res, res1, res2]  
for j in range (3):  
 fig = plt.figure()  
 for i in range(4):  
 plt.subplot(2,2,(i+1)), plt.imshow(res\_all[j][i], cmap='gray')  
 plt.title(f'D0 = {D\_0[i]}'), plt.xticks([]), plt.yticks([])  
 plt.show()

Результат:

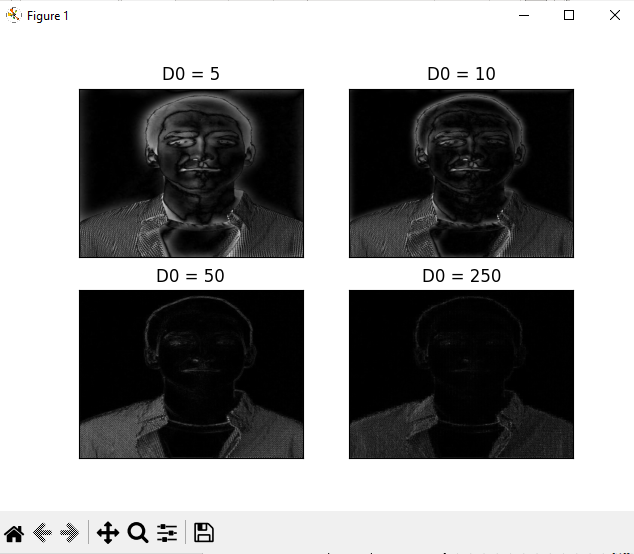
Идеальный фильтр высоких частот



Фильтр Баттерворта высоких частот



Фильтр Гаусса высоких частот



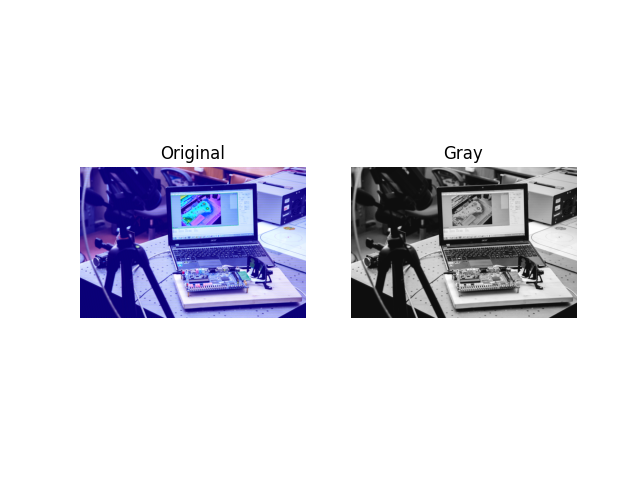
**4) Подобрать и провести избирательную фильтрацию изображения, хранящегося в файле «lab4\_var\_X.jpg», где X – номер индивидуального задания.**

**Я в списке 17, фотографий 11 🡪 вариант 6.**

Считываем изображение и переводим в изображение в оттенках серого. Выводим результат.

img\_dir = 'D:\\Programms\\PyCharm\\PyCharm\\PyCharm\_projects\\Theme\_7\\6.png' # путь к фото  
with Image.open(img\_dir) as img: # открытие файла  
 img.load() #считывание данных  
img\_gray = rgb\_to\_gray(img)  
fig = plt.figure()  
plt.subplot(1,2,1), plt.imshow(img)  
plt.axis('off')  
plt.title('Original')  
plt.subplot(1,2,2), plt.imshow(img\_gray, cmap='gray')  
plt.axis('off')  
plt.title('Gray')  
plt.show()  
plt.figure(figsize=(11, 11))  
plt.imshow(img\_gray, cmap="gray")  
plt.title('Input Image'), plt.xticks([]), plt.yticks([])

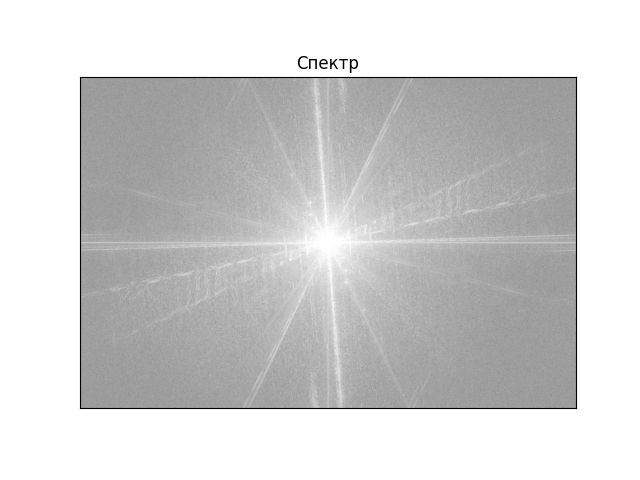
Результат:

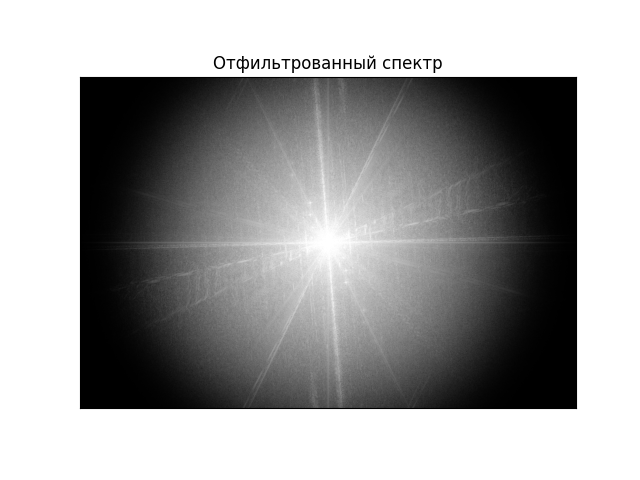




Далее находим спектр, пропускаем через фильтр Гаусса низких частот, делаем обратное преобразование Фурье:

spectrum = fftshift(fft2(img\_gray))  
plt.figure()  
plt.imshow((20\*np.log(1+np.abs(spectrum))).clip(0,255), cmap="gray")  
plt.title('Спектр'), plt.xticks([]), plt.yticks([])  
plt.show()  
filtered = Gauss\_Low(250, img\_gray) \* spectrum  
plt.figure()  
plt.imshow((20\*np.log(1+np.abs(filtered))).clip(0,255), cmap="gray")  
plt.title('Отфильтрованный спектр'), plt.xticks([]), plt.yticks([])  
plt.show()  
img\_back = abs(ifft2(ifftshift(filtered)))  
plt.figure()  
plt.imshow(img\_back, cmap="gray")  
plt.title('Отфильтрованное изображение'), plt.xticks([]), plt.yticks([])  
plt.show()

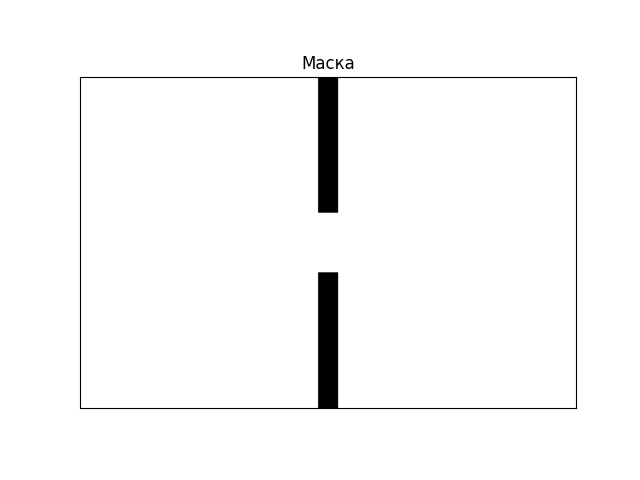






Произведем избирательную фильтрацию (маска как в презентации)

def cross\_filter(wide, img\_gray\_shape):  
 mask = np.ones(img\_gray\_shape)  
 for i in range(img\_gray\_shape[0]):  
 for j in range(img\_gray\_shape[1]):  
 if (img\_gray\_shape[1]//2 - wide <= j <= img\_gray\_shape[1]//2 + wide):  
 mask[i,j] = 0  
 mask[img\_gray\_shape[0]//2 - 3\*wide : img\_gray\_shape[0]//2 + 3\*wide, img\_gray\_shape[1]//2 - 3\*wide : img\_gray\_shape[1]//2 + 3\*wide] = 1  
 return mask  
spectr = cross\_filter(50, img\_gray.shape)  
fig = plt.figure()  
plt.imshow((20\*np.log(1+np.abs(spectr))).clip(0,255), cmap="gray")  
plt.title('Маска'), plt.xticks([]), plt.yticks([])  
plt.show()  
fig = plt.figure()  
plt.imshow(abs(ifft2(ifftshift(spectr\*spectrum))), cmap = 'gray')  
plt.title('Изображение после избирательного фильтра'), plt.xticks([]), plt.yticks([])  
plt.show





Далее черными точками занулим нежелательные места в спектре:

dots = np.array([[1176, 1439], [1034, 1345],[1315, 1528],  
 [977, 1348],[761, 1179],[691, 1169],  
 [632, 1167],[501, 1080],[430, 1080],  
 [728, 1491],[749, 1447],[777, 1398],  
 [886, 1152],[898, 1120],[922, 1063],[942, 1023]])  
dot = np.ones(spectrum.shape)  
for i in range(len(dots)):  
 h = dots[i][0]  
 w = dots[i][1]  
 for j in range(dot.shape[0]):  
 for j2 in range(dot.shape[1]):  
 if (j - h)\*\*2 + (j2 - w)\*\*2 <= 10\*\*2:  
 dot[j,j2] = 0  
plt.figure()  
plt.imshow((20\*np.log(1+np.abs(spectrum\*dot))).clip(0,255), cmap="gray")  
plt.title('Спектр с точками'), plt.xticks([]), plt.yticks([])  
plt.show()  
fig = plt.figure()  
plt.imshow(abs(ifft2(ifftshift(dot\*spectrum))), cmap = 'gray')  
plt.title('Изображение после избирательной фильтрации точками'), plt.xticks([]), plt.yticks([])  
plt.show()  
dot\_paral = np.copy(dot)  
for i in range(dot\_paral.shape[0]):  
 for j in range(dot\_paral.shape[1]):  
 if i in range(954-200,400+1147+1) and j in range(550-200,957+1):  
 dot\_paral[i,j]=0  
plt.figure()  
plt.imshow((20\*np.log(1+np.abs(spectrum\*dot\_paral))).clip(0,255), cmap="gray")  
plt.title('Спектр с точками и прямоугольником'), plt.xticks([]), plt.yticks([])  
plt.show()  
fig = plt.figure()  
plt.imshow(abs(ifft2(ifftshift(dot\_paral\*spectrum))), cmap = 'gray')  
plt.title('Изображение после избирательной фильтрации точками и прямоугольником'), plt.xticks([]), plt.yticks([])  
plt.show()

