Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных средств

**СИСТЕМЫ ОБРАБОТКИ МЕДИАДАННЫХ**

Лабораторная работа № 3

Преобразование яркости изображений и пространственная фильтрация

Вариант № 14

группа № 850702

| Выполнил: | Турко В.Д. |
| --- | --- |
| Проверил | Рыбенков Е.В. |

Минск 2021

**1 ЦЕЛЬ РАБОТЫ**

Изменение яркости, синтез и использование цифровых фильтров в задачах обработки изображений.

**2 ЗАДАНИЕ**

1. Загрузить изображение, рассчитать гистограмму и провести глобальную эквализацию.
2. Загрузить изображение и выполнить локальную гистограммную обработку областями размерами 3х3 и 5х5.
3. Загрузить изображение и провести пространственную фильтрацию с помощью сглаживающего фильтра.
4. Загрузить изображение, выполнить сглаживание и применить медианную фильтрацию.
5. Загрузить изображение и применить фильтрацию с помощью ядра лапласиана.

**3 ХОД РАБОТЫ**

Для выполнения заданий были написаны базовые функции:

1. Загрузка изображения:

def get\_image(name):  
 img = cv.imread('lab3/test\_images/' + name + '.tif')[..., ::-1]  
 return cv.cvtColor(img, cv.COLOR\_BGR2GRAY)

1. Расширение изображения

### def \_\_pad\_image(img, area\_size): area\_r = area\_size // 2 return np.pad(img, (area\_r, area\_r), 'constant', constant\_values=(0, 0))

1. Применение фильтра

def \_\_base\_filter(img, area\_size, filter):

result\_img = np.zeros(img.shape)

padded\_img = \_\_pad\_image(img, area\_size)

h, w = img.shape

area\_r = area\_size // 2

for coord, v in np.ndenumerate(padded\_img[area\_r:h + area\_r, area\_r:w + area\_r]):

i, j = coord

area = padded\_img[i:i + 2 \* area\_r + 1, j:j + 2 \* area\_r + 1]

result\_img[i][j] = filter(area)

return result\_img

**3.1 Глобальная эквализация**

\_\_brightness\_levels = 256

ref = np.arange(\_\_brightness\_levels)

Функция для расчета гистограммы:

def count\_p(img):

probabilities = {}

values, counts = np.unique(img, return\_counts=True)

for index, value in np.ndenumerate(values):

probabilities[value] = counts[index] / img.size

return probabilities

Функция глобальной эквализации:

def equalize(img):

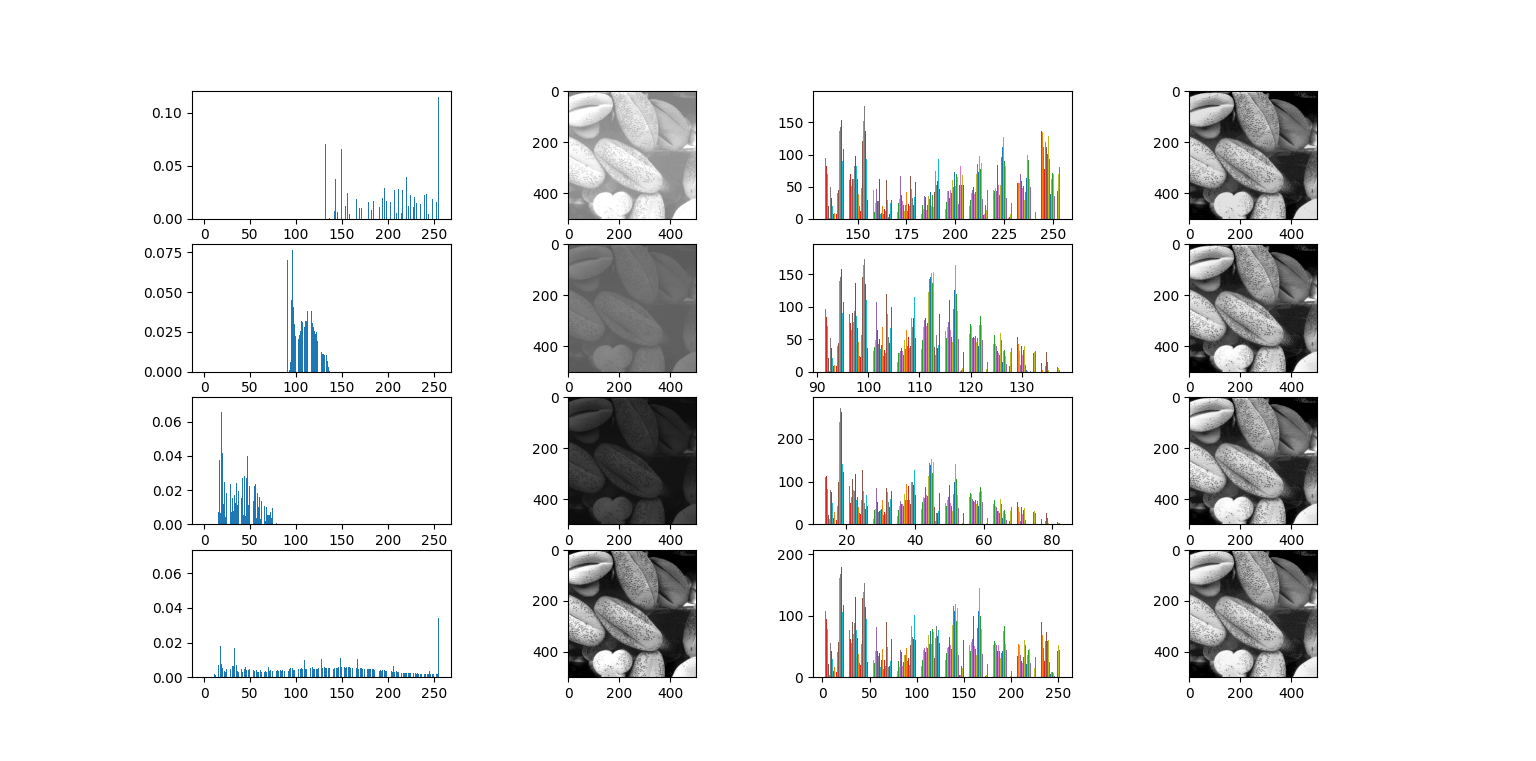
p = count\_p(img)

t = np.cumsum(np.fromiter(p.values(), dtype='float'), 0) \* \

(\_\_brightness\_levels - 1)

new\_p = dict(zip(p.keys(), t))

return [[new\_p[value] for value in row] for row in img]



**3.2 Локальная эквализация**

Функция локальной эквализации - по сути функция глобальной эквализации выбранной области:

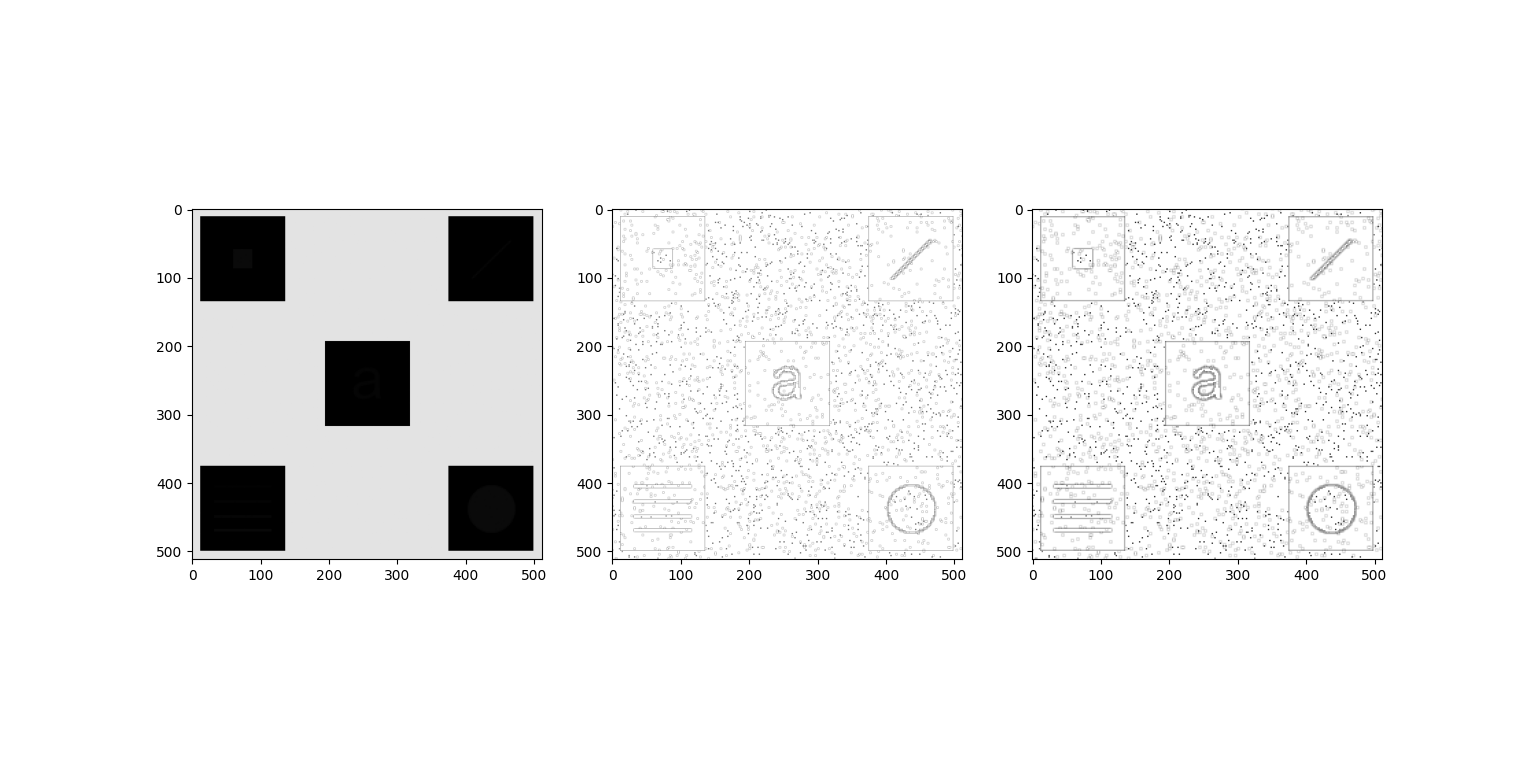
def \_\_equalize\_local(area):

area\_h, area\_w = area.shape

return equalize(area)[area\_h][area\_h]

def equalize\_local(img, area\_size):

return \_\_base\_filter(img, area\_size, \_\_equalize\_local)



**3.3 Пространственная фильтрация**

Функция расчета ядра фильтра:

def \_\_flat\_filter(s, t, m, n):

a = (m - 1)/2

b = (n - 1)/2

if -a <= s <= a and -b <= t <= b:

return 1/(m\*n)

else:

return 0

Фильтр области:

def \_\_convulate(area):

m, n = area.shape

result = 0

for coord, v in np.ndenumerate(area):

s, t = coord

result += \_\_flat\_filter(s, t, m, n) \* area[m - s - 1][n - t - 1]

return result

def my\_2dfilter(img, area\_size):

return \_\_base\_filter(img, area\_size, \_\_convulate)

Функция пороговой обработки изображения:

def threshold\_filter(img, threshold):

result\_img = np.zeros(img.shape)

h, w = img.shape

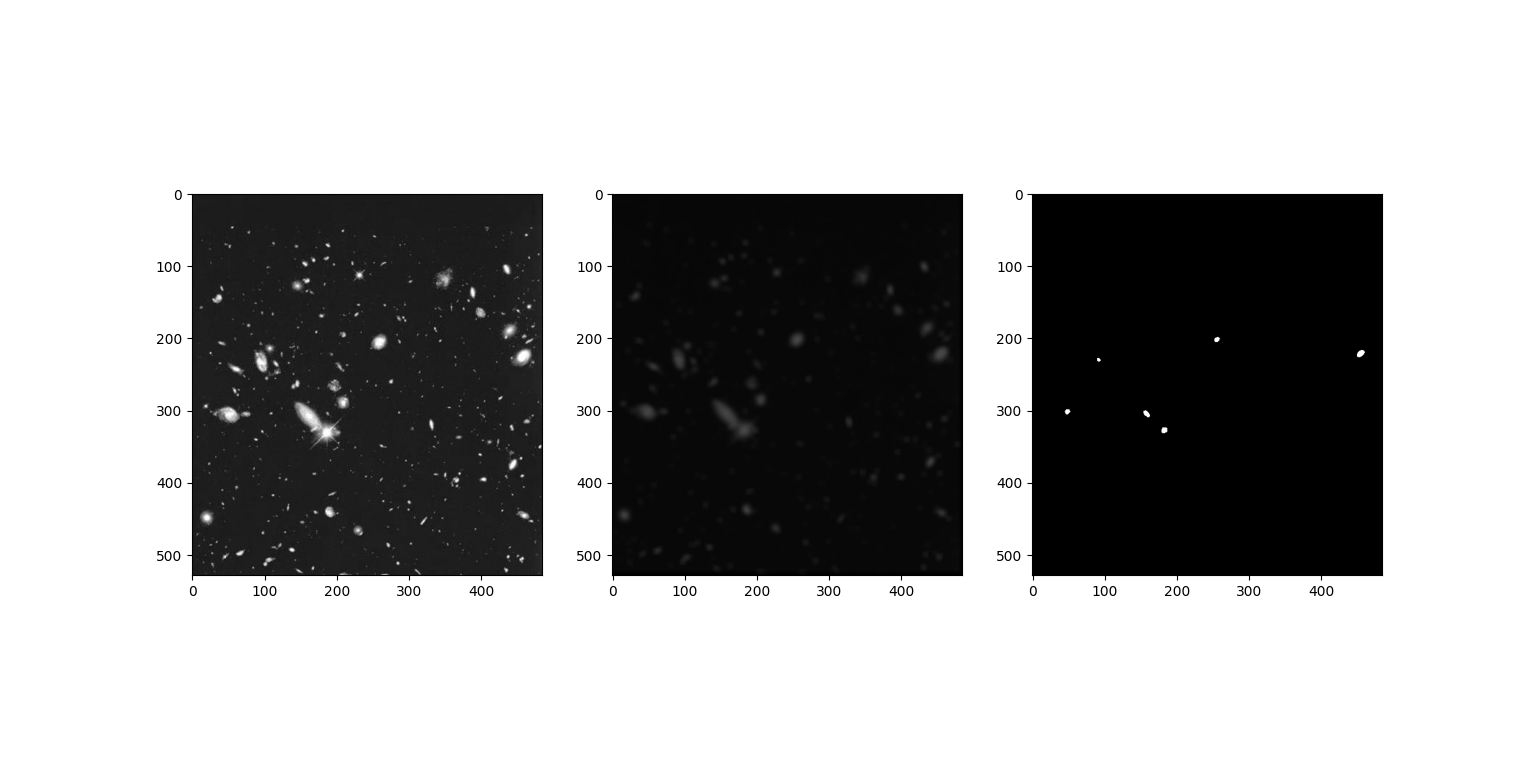
for coord, v in np.ndenumerate(img):

i, j = coord

if (img[i][j] > 64):

result\_img[i][j] = 255

return result\_img



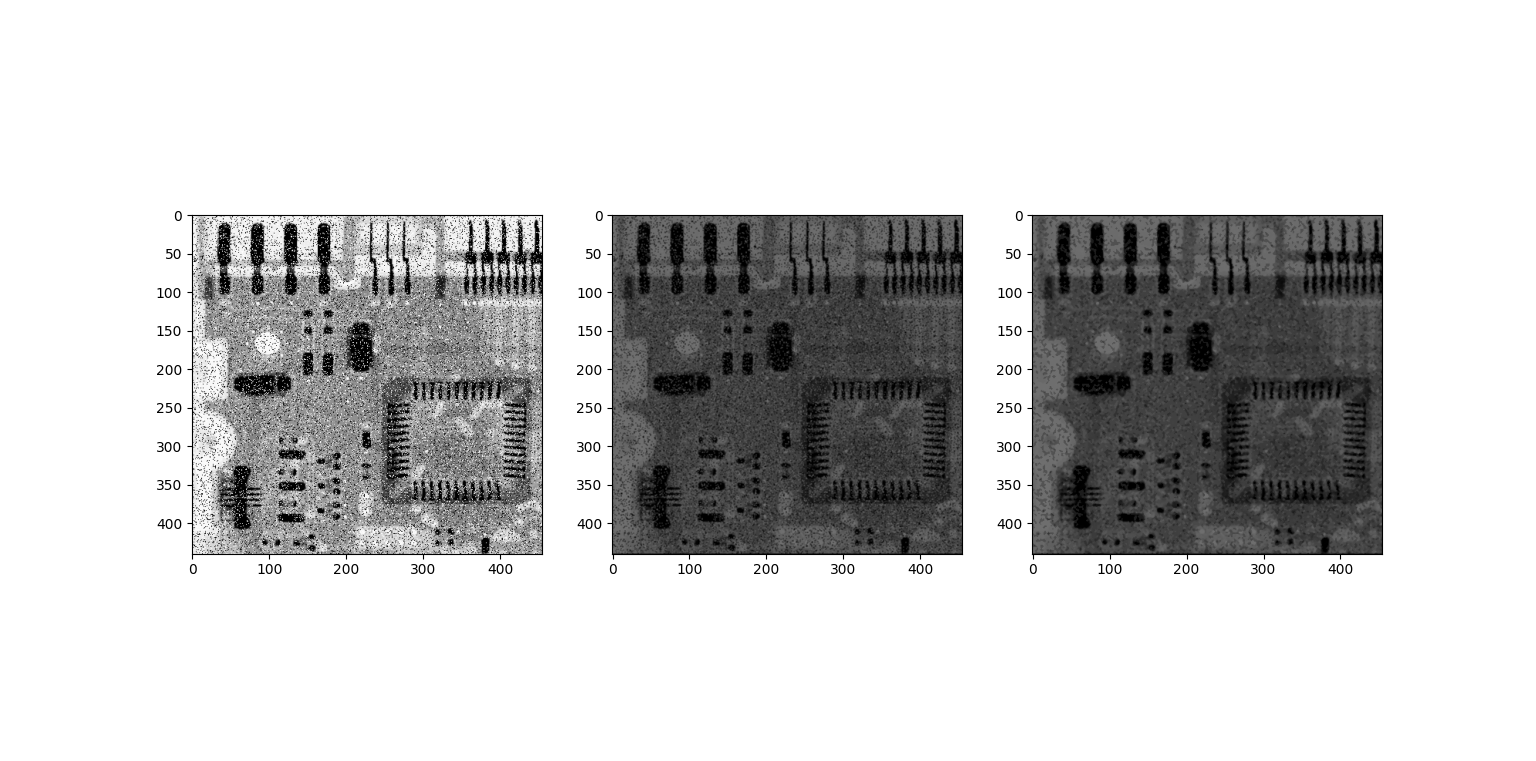
**3.4 Медианная фильтрация**

Функция медианной фильтрации:

def median\_filter(img, area\_size):

median = lambda x: np.median(np.concatenate(x), axis=0)

return \_\_base\_filter(img, area\_size, median)



**3.5 Повышение резкости изображения**

Функции фильтрации с помощью ядра лапласиана:

def \_\_laplasian(area):

\_\_coeffs = [[0, 1, 0], [1, -4, 1], [0, 1, 0]]

return np.sum(np.concatenate(area \* \_\_coeffs), axis=0)

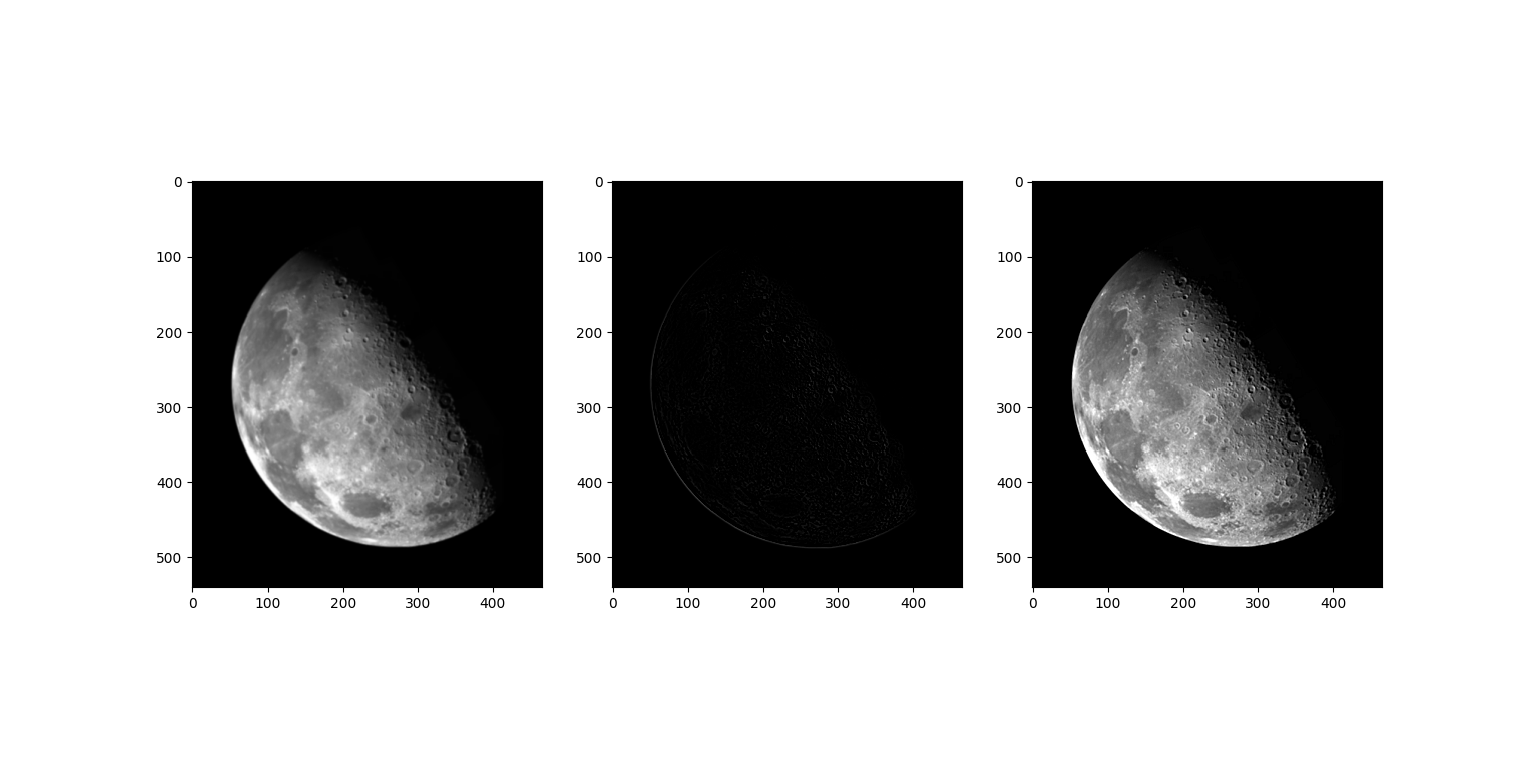
def laplasian\_filter\_3x3(img):

return \_\_base\_filter(img, 3, \_\_laplasian)

Функция для добавления фона изображения:

def remove\_back(img, filtered\_img):

return np.subtract(img, filtered\_img)



**4 ВЫВОД:**

В данной лабораторной работе я научился изменять яркость изображений, а также синтезировать и использовать фильтры для обработки изображений.