Студент гр. РИМ-181226 Бабикова Евгения Витальевна

---

4. Фильтрация изображений

Импорт необходимых библиотек и модулей

In [1]:

```
import numpy as np
from skimage.io import imread, imshow
from skimage.color import rgb2gray
from skimage import img_as_ubyte
from scipy.signal import convolve2d
import matplotlib.pyplot as plt
%matplotlib inline
```

4.1. Операция свертки. Линейные фильтры

Свертка изображения

In [2]:

```
image = np.array([[0, -8, -3, -2],
                  [1, 9, -8, 0],
                  [9, -4, 5, -9],
                  [6, -4, 6, 3]])

kernel = np.array([[4, -5, 4],
                   [-6, -8, -2],
                   [1, 5, 5]])

print(convolve2d(kernel, image, mode='valid'))
```

```
[[  7 -88]
 [ 94  -7]]
```

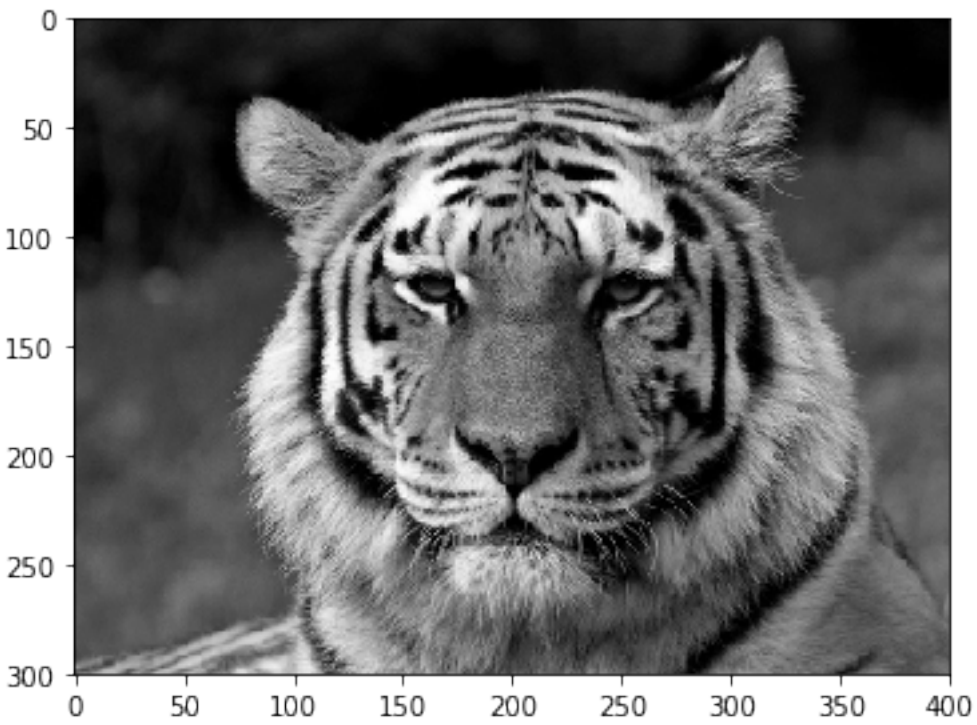
Box-фильтр

Box-фильтр при применении к изображению дает размытое изображение.

Считывание и просмотр изображения.

In [93]:

```
img = imread('tiger-gray-small.png')
imshow(img);
```



Box-фильтрация изображения. Применение свертки с ядром 5*5:

$$\begin{vmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{vmatrix} \cdot$$

Необходимо поделить получившиеся значения на 25, чтобы не изменить яркость изображения.

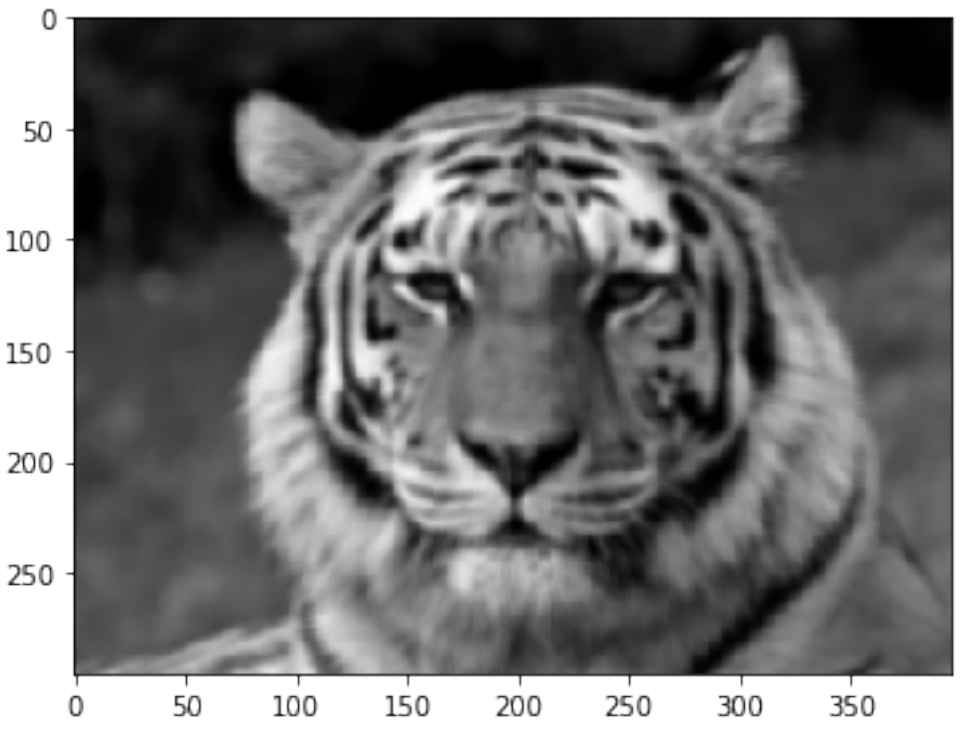
In [97]:

```
kernel_size = 5
img_out = (convolve2d(img, np.ones((kernel_size, kernel_size)), mode = "valid"
) / 25).astype('uint8')
```

Просмотр полученного box-фильтрацией изображения.

In [98]:

```
imshow(img_out, cmap=plt.cm.gray);
```



Сравнение полученного изображения и образца.

In [99]:

```
img_sample=imread('box-tiger.png')
np.array_equal(img_out, img_sample)
```

Out[99]:

```
True
```

Подсчет функции Гаусса

Box-фильтрация производит артефакты (вертикальные линии). Более качественный результат дает гауссовский фильтр.

Функция Гаусса:

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2-y^2}{2\sigma^2}}.$$

Функция принимает максимальное значение в точке (0,0) (центральный пиксель), отдалясь от него значения уменьшаются, но всегда положительны.

Параметр σ - разброс гауссианы. Чем меньше σ, тем более резкий и высокий пик у функции Гаусса. При малом σ просматривается только небольшая область изображения, сглаживание небольшое.

Реализация подсчета функции Гаусса и проверка ее работы.

In [7]:

```
def gauss_function(sigma, x, y):
    return 1/((2*np.pi)*sigma**2)*np.e**((-x**2 - y**2)/(2*sigma**2))
gauss_function(1, 1, 1)
```

Out[7]:

```
0.05854983152431917
```

Ядро Гауссовского фильтра

Вычисление элементов ядра Гауссоовского фильтра и проверка работы функции.

In [65]:

```
def gauss_kernel(sigma):
    k = round(3*sigma) # радиус фильтра (размер ядра)
    # Заполняем ядро значениями функции Гаусса
    kernel = np.array([[gauss_function(sigma, x, y) for x in range(-k, k+1)] f
or y in range(-k, k+1)])
    # Нормируем ядро, чтобы удостовериться, что сумма элементов = 1 и яркость
изображения не изменится
    norm_kernel = kernel / np.sum(kernel, axis=(0,1))
    return norm_kernel
gauss_kernel(0.33)
```

Out[65]:

```
array([[9.87532107e-05, 9.73995858e-03, 9.87532107e-05],
       [9.73995858e-03, 9.60645153e-01, 9.73995858e-03],
       [9.87532107e-05, 9.73995858e-03, 9.87532107e-05]])
```

Гауссовская фильтрация

Реализация фильтрации изображения с помощью Гауссовского фильтра с $\sigma = 0.66$.

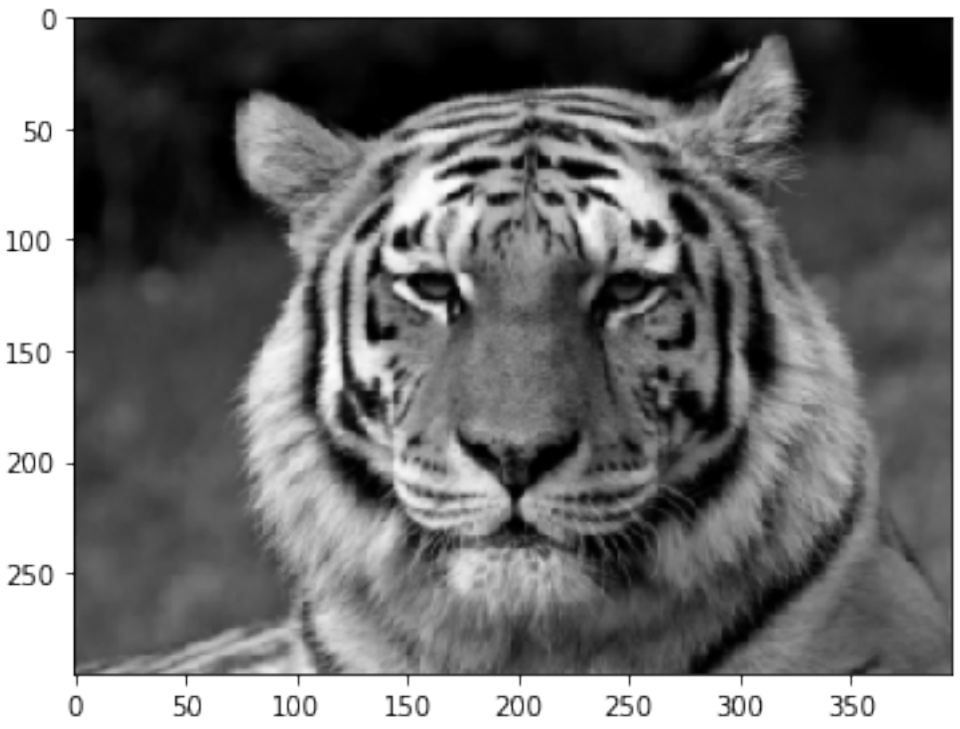
In [11]:

```
kernel = gauss_kernel(0.66)
img_out = convolve2d(img, kernel, mode='valid').astype('uint8')
```

Просмотр полученного гауссовской фильтрацией изображения.

In [12]:

```
imshow(img_out);
```



Сравнение полученного изображения и образца.

In [13]:

```
img_sample=imread('gaussian-tiger.png')
np.array_equal(img_out, img_sample)
```

Out[13]:

```
True
```

Повышение резкости изображения

Реализация повышения четкости изображения путём фильтрации изображения с ядром

$$\frac{1}{10}\begin{bmatrix} -1 & -2 & -1 \\ -2 & 22 & -2 \\ -1 & -2 & -1 \end{bmatrix}.$$

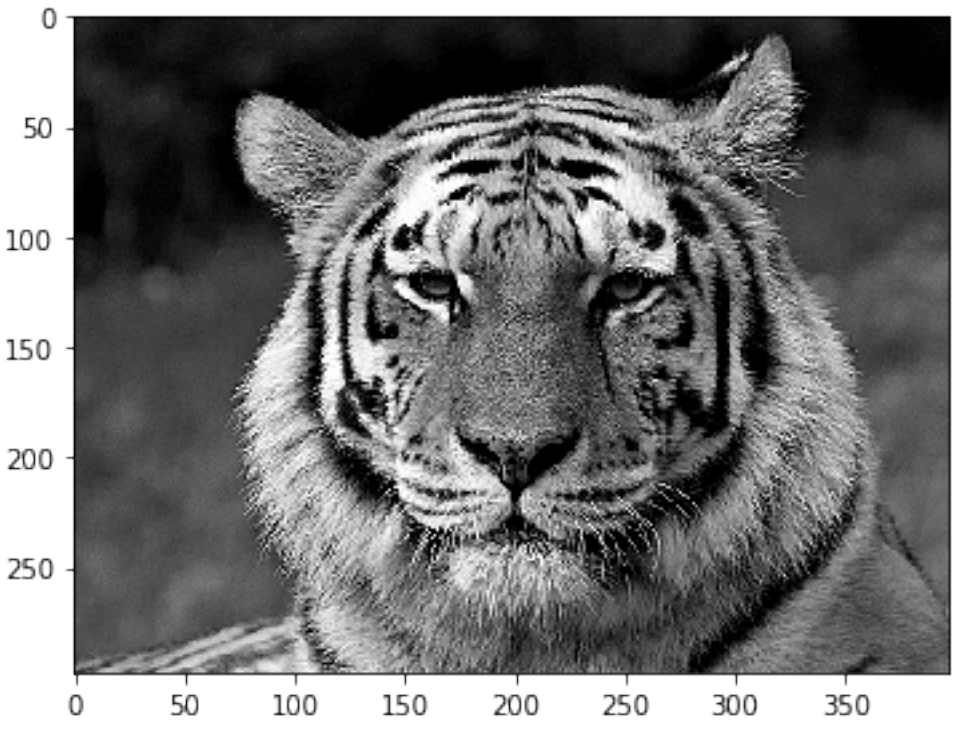
In [47]:

```
kernel = np.array([[-1, -2, -1],
                   [-2, 22, -2],
                   [-1, -2, -1]]) / 10
img_sharping = convolve2d(img, kernel, mode='valid')
img_out = np.clip(img_sharping, 0, 255).astype('uint8')
```

Просмотр изображения с повышенной резкостью.

In [48]:

```
imshow(img_out);
```



4.2. Медианный фильтр

Медианный фильтр позволяет бороться с шумом типа соль и перец (в случайном порядке пиксели изображения становятся белыми или черными). Данный фильтр похож на box-фильтр (тоже рассматривает окрестность изображения), только не имеет собственного ядра. Медианный фильтр берет окрестность вокруг выброса, вытягивает ее в один список, отсортированный по возрастанию, вычисляет медиану и записывает ее в значение центрального пикселя.

Реализация медианного фильтра с окном 7*7.

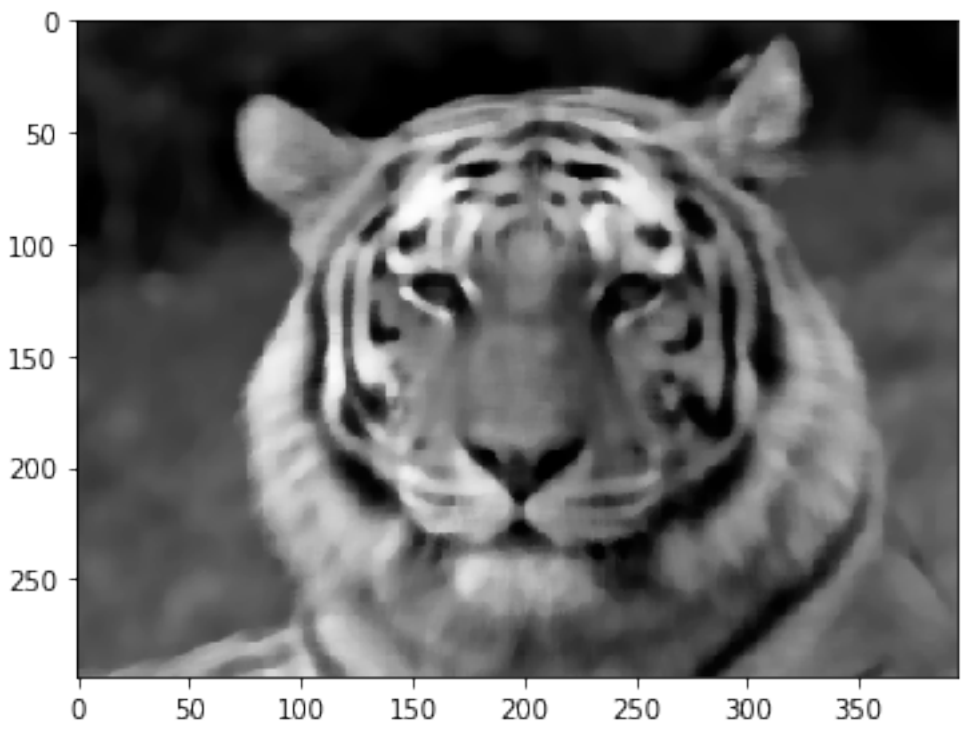
In [100]:

```
img_out = img.copy() # заготовка результата
rows, cols = img.shape
for i in range(3,rows-3):
    for j in range(3,cols-3):
        a = img[i-3:i+3+1, j-3:j+3+1] # выбор окрестности изображения
        m = np.median(a) # определение медианы
        img_out[i][j] = m # записываем медиану
img_out = img_out[3:rows-3, 3:cols-3] # обрезка результата
```

Просмотр отфильтрованного изображения.

In [101]:

```
imshow(img_out);
```



Сравнение полученного изображения с образцом.

In [102]:

```
img_sample=imread('median-tiger.png')
np.array_equal(img_out, img_sample)
```

Out[102]:

```
True
```