

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ

ПРОФЕССИОНАЛЬНОГО

**РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА И ГОСУДАРСТВЕННОЙ СЛУЖБЫ
ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ**

ЛИПЕЦКИЙ ФИЛИАЛ

-
КАФЕДРА ГУМАНИТАРНЫХ И ЕСТЕСТВЕННОНАУЧНЫХ ДИСЦИПЛИН

Лабораторная работа №5

по дисциплине
**«Информационные технологии в государственном и
муниципальном управлении»**

Выполнила:
студентка 1 курса
группы У20-1
Александрова Алина

Проверил:
доцент
Яворский В.М.

```
[1]: import cv2
import numpy as np
from scipy import misc
i = misc.ascent()

[2]: import matplotlib.pyplot as plt
plt.grid(False)
plt.gray()
plt.axis('off')
plt.imshow(i)
plt.show()

[3]: L_transformed = np.copy(i)
size_x = L_transformed.shape[0]
size_y = L_transformed.shape[1]

[4]: #filter = [[0, 1, 0], [1, -4, 1], [0, 1, 0]]
# filter para denosir
#filter = [[1, -1, -2, -1], [0, 0, 0], [1, 2, 1]]
filter = [[1, 0, 1], [0, 0, 1], [1, 0, 1]]

# Если все цифры в фильтре не складываются до значения 0 или 1,
# или скроп этого потребуется никакие условия, чтобы добиться этого,
# то это означает, что веса равны 1,1,1,1,1,1,1,1,1
# или, например, если веса равны 1,1,1,1,0,0,0,0,0
# Они складываются до 10, так что мы бы установили вес .1, если хотели их нормализовать.

weight = 1

[5]: for x in range(size_x-1):
    for y in range(size_y-1):
        convolution = 0.0
```

Создаём новый блокнот по ссылке:
<https://colab.research.google.com/>;
И называем блокнот.

```
[6]: #filter = [[0, 1, 0], [1, -4, 1], [0, 1, 0]]
# filter para denosir
#filter = [[1, -1, -2, -1], [0, 0, 0], [1, 2, 1]]
filter = [[1, 0, 1], [0, 0, 1], [1, 0, 1]]

# Если все цифры в фильтре не складываются до значения 0 или 1,
# или скроп этого потребуется никакие условия, чтобы добиться этого,
# то это означает, что веса равны 1,1,1,1,1,1,1,1,1
# или, например, если веса равны 1,1,1,1,0,0,0,0,0
# Они складываются до 10, так что мы бы установили вес .1, если хотели их нормализовать.

weight = 1

[7]: for x in range(size_x-1):
    for y in range(size_y-1):
        convolution = 0.0
```

Сначала мы загружаем изображение из пакета "Scipy". Для этого мы добавляем новый код:

```
import cv2
import numpy as np
from scipy import misc
i = misc.ascent()
```

```
[8]: L_transformed = np.copy(i)
size_x = L_transformed.shape[0]
size_y = L_transformed.shape[1]

[9]: #filter = [[0, 1, 0], [1, -4, 1], [0, 1, 0]]
# filter para denosir
#filter = [[1, -1, -2, -1], [0, 0, 0], [1, 2, 1]]
filter = [[1, 0, 1], [0, 0, 1], [1, 0, 1]]

# Если все цифры в фильтре не складываются до значения 0 или 1,
# или скроп этого потребуется никакие условия, чтобы добиться этого,
# то это означает, что веса равны 1,1,1,1,1,1,1,1,1
# или, например, если веса равны 1,1,1,1,0,0,0,0,0
# Они складываются до 10, так что мы бы установили вес .1, если хотели их нормализовать.

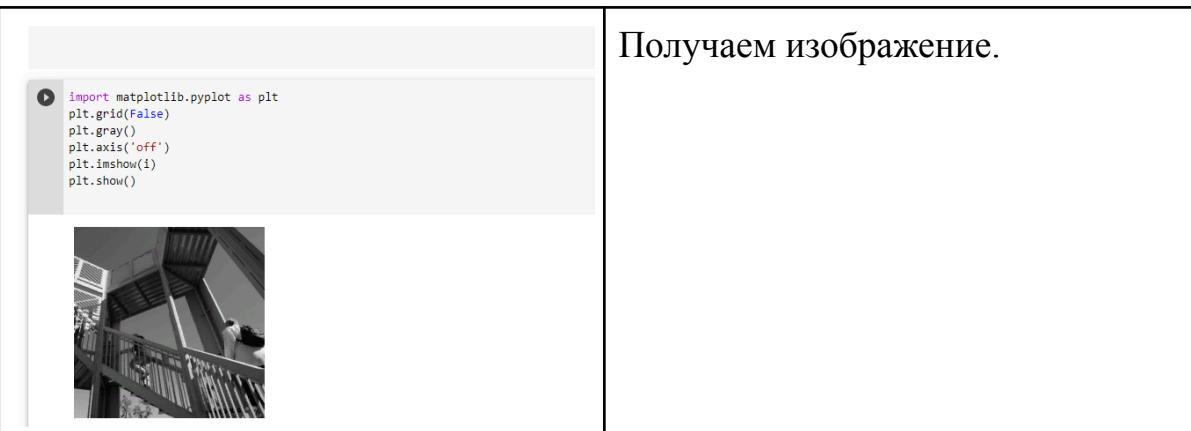
weight = 1

[10]: for x in range(size_x-1):
    for y in range(size_y-1):
        convolution = 0.0
```

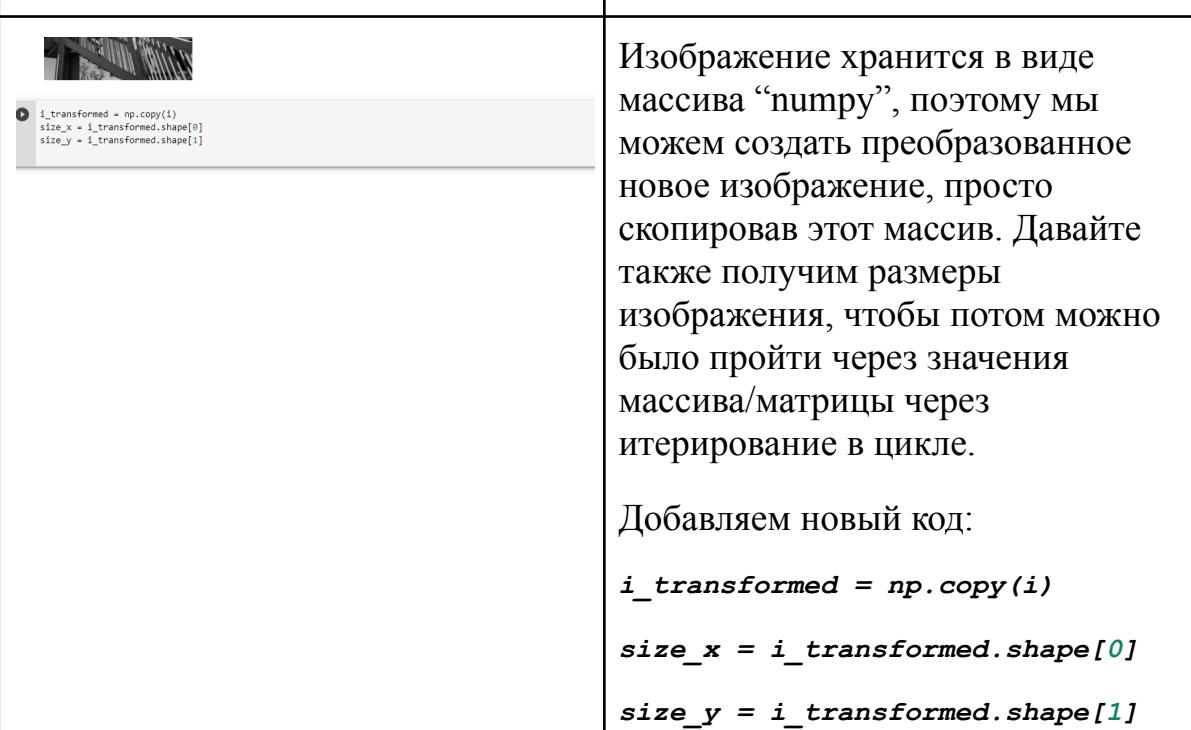
Далее мы используем библиотеку "pyplot" для того, чтобы нарисовать изображение, чтобы мы знали, как оно выглядит.

Добавляем новый код:

```
import matplotlib.pyplot as plt
plt.grid(False)
plt.gray()
plt.axis('off')
plt.imshow(i)
plt.show()
```



Получаем изображение.



Изображение хранится в виде массива “пиптру”, поэтому мы можем создать преобразованное новое изображение, просто скопировав этот массив. Давайте также получим размеры изображения, чтобы потом можно было пройти через значения массива/матрицы через итерирование в цикле.

Добавляем новый код:

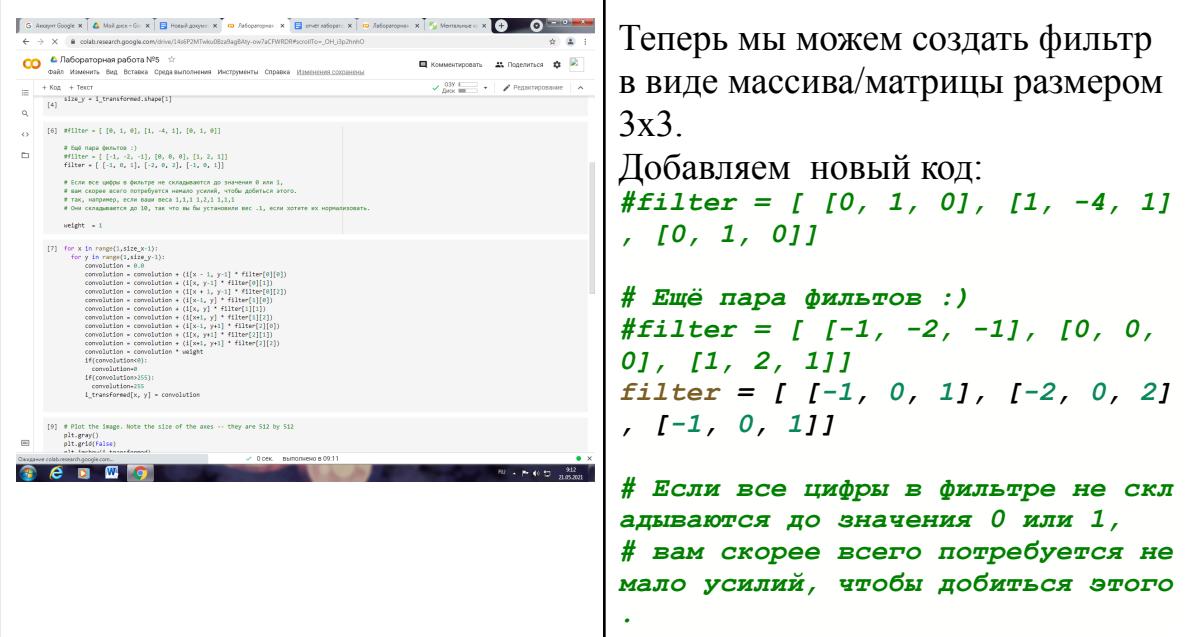
```

i_transformed = np.copy(i)

size_x = i_transformed.shape[0]

size_y = i_transformed.shape[1]

```



Теперь мы можем создать фильтр в виде массива/матрицы размером 3x3.

Добавляем новый код:

```
#filter = [[0, 1, 0], [1, -4, 1], [0, 1, 0]]
```

Ещё пара фильтров :)

```
#filter = [[-1, -2, -1], [0, 0, 0], [1, 2, 1]]
```

```
filter = [[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]]
```

Если все цифры в фильтре не складываются до значения 0 или 1, вам скорее всего потребуется не мало усилий, чтобы добиться этого.

```
# так, например, если ваши веса 1
,1,1 1,2,1 1,1,1
# Они складываются до 10, так что
# вы бы установили вес .1, если хотите их нормализовать.

weight = 1
```

```
[0] # Для каждого элемента:
# filter = [[1, -1, -1], [0, 0, 0], [1, 2, 1]]
# filter = [[1, 1, 0, 1], [1, 0, 2, 1], [1, 0, 0, 1]]
# Если все цифры в фильтре не складываются до значения 0 или 1,
# или содержит всего один элемент, то делится на это.
# Так же, если веса равны 1,1,1,1,1,1,1,1,1
# Они складываются до 10, так что мы установим вес .1, если хотим их нормализовать.
weight = 1

for x in range(size_x-1):
    for y in range(size_y-1):
        convolution = 0.0
        convolution = convolution + (i[x-1, y-1] * filter[0][0])
        convolution = convolution + (i[x, y-1] * filter[0][1])
        convolution = convolution + (i[x+1, y-1] * filter[0][2])
        convolution = convolution + (i[x-1, y] * filter[1][0])
        convolution = convolution + (i[x, y] * filter[1][1])
        convolution = convolution + (i[x+1, y] * filter[1][2])
        convolution = convolution + (i[x-1, y+1] * filter[2][0])
        convolution = convolution + (i[x, y+1] * filter[2][1])
        convolution = convolution + (i[x+1, y+1] * filter[2][2])
        convolution = convolution / weight
        if(convolution>0):
            convolution = convolution
        if(convolution<0):
            convolution = 0.0
        L_transformed[x, y] = convolution

# Plot the image. Note the size of the axes -- they are 512 by 512
plt.gray()
plt.imshow(L)
plt.imshow(L_transformed)
plt.axis('off')
plt.show()
```

Теперь создаём конволюцию. Выполним итерацию по изображению, оставив поле в 1 пиксел, и умножим каждого из соседей текущего пикселя на значение, заданное в фильтре.

Добавляем новый код:

```
for x in range(1, size_x-1):

    for y in range(1, size_y-1):

        convolution = 0.0

        convolution = convolution + (i[x-1, y-1] * filter[0][0])

        convolution = convolution + (i[x, y-1] * filter[0][1])

        convolution = convolution + (i[x+1, y-1] * filter[0][2])

        convolution = convolution + (i[x-1, y] * filter[1][0])

        convolution = convolution + (i[x, y] * filter[1][1])

        convolution = convolution + (i[x+1, y] * filter[1][2])

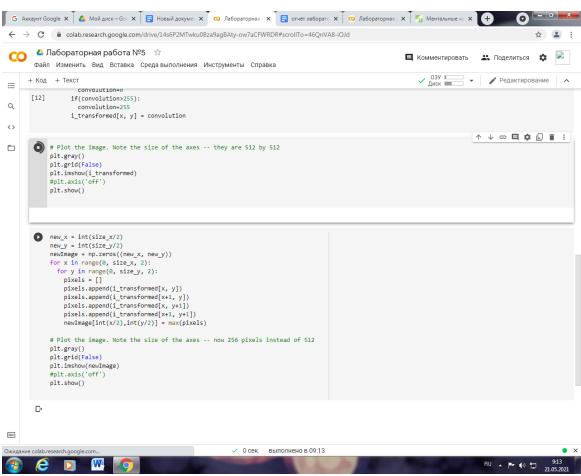
        convolution = convolution + (i[x-1, y+1] * filter[2][0])

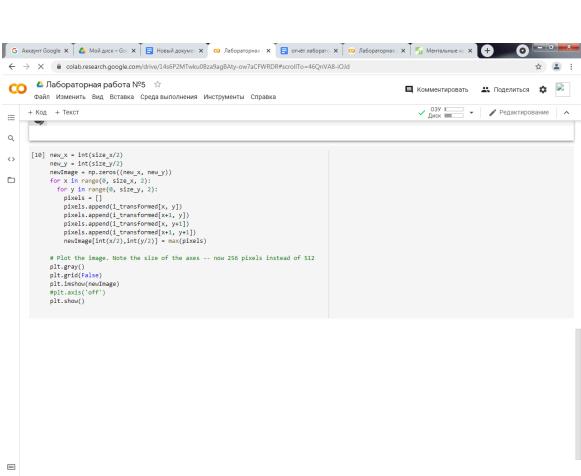
        convolution = convolution + (i[x, y+1] * filter[2][1])

        convolution = convolution + (i[x+1, y+1] * filter[2][2])

        convolution = convolution * weight
```

| | |
|--|--|
| | <pre> <i>if</i>(convolution<0): convolution=0 <i>if</i>(convolution>255): convolution=255 <i>i_transformed[x, y] = convolution</i> </pre> |
|--|--|

| | |
|--|---|
|  | <p>Шаг №8. Теперь мы можем построить изображение, чтобы увидеть эффект свёртки. Добавляем новый код:</p> <pre> <i># Plot the image. Note the size of the axes -- they are 512 by 512</i> plt.gray() plt.grid(False) plt.imshow(i_transformed) plt.axis('off') plt.show() </pre> <p>И получаем изображение</p> |
|--|---|

| | |
|---|---|
|  | <p>Шаг №9. Данный код покажет $(2, 2)$ операцию подвыборки</p> <pre> new_x = int(size_x/2) new_y = int(size_y/2) newImage = np.zeros((new_x, new_y)) for x in range(0, size_x, 2): for y in range(0, size_y, 2): pixels = [] for i in range(x, size_x, 2): for j in range(y, size_y, 2): pixels.append(i_transformed[i, j]) pixels.append(i_transformed[i+1, j]) pixels.append(i_transformed[i, j+1]) pixels.append(i_transformed[i+1, j+1]) rectangle[int(x/2),int(y/2)] = max(pixels) </pre> |
|---|---|

```
pixels = []

pixels.append(i_transformed[x,
y])

pixels.append(i_transformed[x+1,
y])

pixels.append(i_transformed[x,
y+1])

pixels.append(i_transformed[x+1,
y+1])

newImage[int(x/2),int(y/2)] =
max(pixels)

# Plot the image. Note the size
of the axes -- now 256 pixels
instead of 512

plt.gray()

plt.grid(False)

plt.imshow(newImage)

#plt.axis('off')

plt.show()
```

После чего получаем изображение