

# Лабораторная работа №3

## Обработка изображений с использованием библиотеки PIL (Python Imaging Library).

### 1. Смешивание двух изображений с использованием библиотеки PIL в Python

#### Синтаксис:

```
from PIL import Image

img = Image.blend(im1, im2, alpha)
```

#### Параметры:

- `im1` - первое изображение/картинка. Должна быть в режиме `RGBA`.
- `im2` - второе изображение/картинка. Должна иметь тот же режим и размер, что и первое изображение.
- `alpha` - альфа-фактор интерполяции. Если альфа равна 0,0, то возвращается копия первого изображения. Если альфа равна 1,0, то возвращается копия второго изображения. Ограничений по альфа-значению нет. При необходимости результат обрезается, чтобы соответствовать допустимому выходному диапазону.

#### Возвращаемое значение:

- Объект изображения `Image`.

#### Описание:

Функция `Image.blend()` модуля `Pillow` создает новое изображение путем интерполяции между двумя входными изображениями с использованием постоянной альфы.

Интерполяция происходит по формуле:

```
out_image = image1 * (1.0 - alpha) + image2 * alpha
```

`alpha` - фактор интерполяции. Если альфа равна 0,0, то возвращается копия первого изображения. Если альфа равна 1,0, то возвращается копия второго изображения. Ограничений по альфа-значению нет. При необходимости результат обрезается, чтобы соответствовать допустимому выходному диапазону.

- Если картинки разного размера, то одну из них можно подогнать под другую методом `Image.resize()`.
- Если картинки имеют разный режим, то одну из них можно конвертировать методом `Image.convert()`.

2.

## **Размытие изображений**

Классический – и очень полезный – пример свертки изображения – *гауссово размытие*. Идея в том, что (полутоновое) изображение  $I$  сворачивается с гауссовым ядром, в результате чего получается размытое изображение:

$$I_{\sigma} = I * G_{\sigma}.$$

Здесь  $*$  обозначает операцию свертки,  $G_{\sigma}$  – двумерное гауссово ядро со стандартным отклонением  $\sigma$ :

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

Гауссово размытие применяется для интерполяции, вычисления особых точек изображения и во многих других приложениях.

В состав SciPy входит модуль фильтрации `scipy.ndimage.filters`, который позволяет вычислять подобные свертки с помощью быстрого метода разделения переменных. Нам нужно только написать:

```
from PIL import Image
from numpy import *
from scipy.ndimage import filters

im = array(Image.open('empire.jpg').convert('L'))
im2 = filters.gaussian_filter(im, 5)
```

Второй параметр функции `gaussian_filter()` задает стандартное отклонение.

На рис. 1 показаны результаты размытия изображения с увеличивающимся  $\sigma$ . Чем больше стандартное отклонение, тем меньше остается деталей. Для размытия цветных изображений нужно применить гауссово размытие к каждому цветовому каналу:

```

im = array(Image.open('empire.jpg'))
im2 = zeros(im.shape)
for i in range(3):
    im2[:, :, i] = filters.gaussian_filter(im[:, :, i], 5)
im2 = uint8(im2)

```

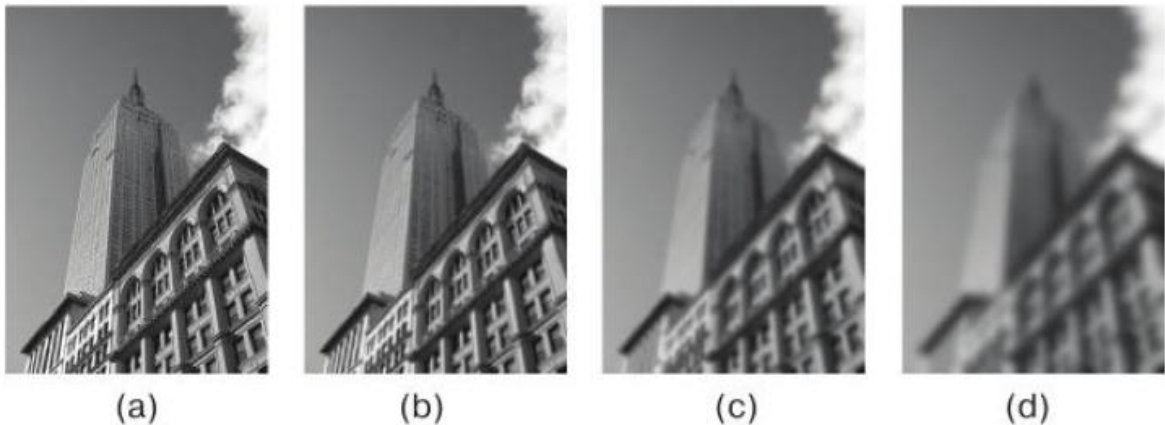
Последнее преобразование к типу uint8 не всегда обязательно, оно просто дает 8-разрядное представление пикселей. Того же эффекта можно было бы добиться, написав:

```

im2 = array(im2, 'uint8')

```

Дополнительные сведения об этом модуле и различных параметрах можно найти в документации по SciPy на странице <http://docs.scipy.org/doc/scipy/reference/ndimage.html>.



**Рис. 1.** Пример гауссова размытия с применением модуля `scipy.ndimage.filters`: (a) исходное полутоновое изображение; (b) фильтра Гаусса с  $\sigma = 2$ ; (c) с  $\sigma = 5$ ; (d) с  $\sigma = 10$

**Задание 1.** Загрузить два JPEG-изображения, пересохранить их в PNG-формат. Осуществить объединение (смешивание) изображений с прозрачностью. Осуществить преобразование с Гауссовым размытием первого изображения.

Пример:

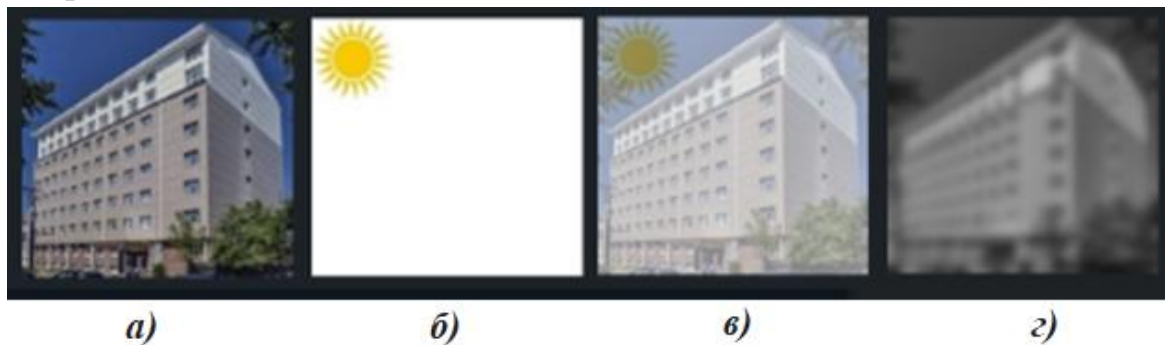


Рис. 2. а) – первое изображение, б) – второе изображение, в) – объединение изображений, г) – Гауссово размытие

myCompiler

Enter a title...

Python ⓘ

```
1 from PIL import Image
2 import binascii
3 from numpy import*
4 from scipy.ndimage import filters
5
6 # Данные1 в формате hex с пробелами
7 hex_data1 = "FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 60 00 60 00 00 FF DB 00 43 00 02 01 01 02 01
8
9 # Удаляем все пробелы из строки
10 hex_data1 = hex_data1.replace(" ", "")
11
12 # Преобразуем hex строку в байты
13 binary_data = binascii.unhexlify(hex_data1)
14
15 # Сохраняем байты в файл
16 with open("hgu.jpg", "wb") as file:
17     file.write(binary_data)
18
19 # Данные2 в формате hex с пробелами
20 hex_data2 = "FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 60 00 60 00 00 FF E1 00 22 45 78 69 66 00 00
21
22 # Удаляем все пробелы из строки
23 hex_data2 = hex_data2.replace(" ", "")
24
25 # Преобразуем hex строку в байты
26 binary_data = binascii.unhexlify(hex_data2)
27
28 # Сохраняем байты в файл
29 with open("sun.jpg", "wb") as file:
30     file.write(binary_data)
31
32
33 #Пересохранение изображений в форматах png
34
35 pil_im1=Image.open("hgu.jpg")
36 pil_im1.save('hgu.png')
37
38 pil_im2=Image.open("sun.jpg")
39 pil_im2.save('sun.png')
40
41 #Объединение с прозрачностью
42
43
44 pil_im3 = Image.blend(pil_im1, pil_im2, 0.5)
45 pil_im3.save('output_image.png')
46
47 #Преобразование с Гауссовым размытием
48 im=array(pil_im1.convert("L"))
49 im2=filters.gaussian_filter(im,3)
50 pil_im1=Image.fromarray(im2)
51 pil_im1.save('hgu gaussian.jpg')
```

Рис. 3. Листинг кода Python (PIL) к примеру преобразований изображений



Рис. 4. Пример вывода результата (myCompiler)

**Задание 2\*.** Загрузить изображение и написать программу, которая проходит по каждому пикселю изображения и получает для него значение цвета RGB-нотации. Переприсвоить значения (R заменить на G, G заменить на B, B заменить на R). Сохранить новое изображение.