Санкт-Петербургский Национальный Исследовательский Университет
Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий

ИТОГОВЫЙ ПРОЕКТ

по дисциплине «Алгоритмы и структуры данных»

на тему: «Преобразование Фурье. Склейка изображений»

Выполнила:

Боброва М.И.

Проверил:

Мусаев А.А.

Санкт-Петербург,

2023

Введение

Цель:

Научиться склеивать изображения и обосновывать работоспособность программы с помощью преобразования Фурье.

Задачи:

1. Подготовить одно изображение для экспериментов с гауссовской и лапласовской пирамидой.

2. Построить гауссовскую пирамиду изображения из не менее чем пяти слоев. Визуализировать полученные изображения и амплитуды частот изображений пирамиды и убедиться, что на каждом слое диапазон частот сужается. Построить пирамиду для трех различных значения сигмы гауссовского ядра.

3. Провести аналогичные эксперименты с лапласовской пирамидой.

4. На основе функций построения гауссовской и лапласовской пирамиды написать функцию склейки двух изображений на основе маски. Функция должна возвращать склеенные изображения и промежуточные результаты — склеенные изображения разных частот (т.е. лапласовскую пирамиду совмещенного изображения).

5. Посмотреть, как ведет себя склейка при изменении sigma и при изменении количества слоев.

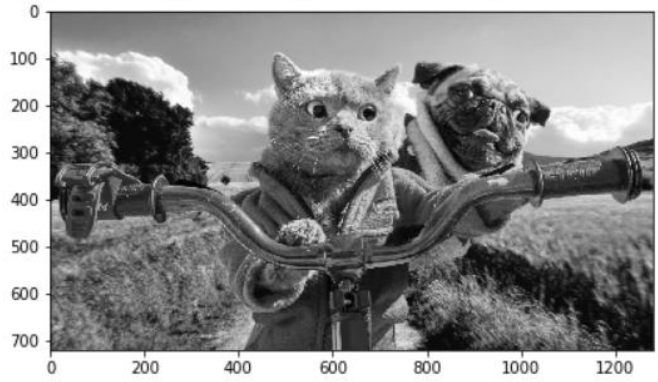
Выполнение

1. Импорт библиотек и подготовка изображения

```
import numpy as np
from math import exp, pi
from numpy.fft import fft2, fftshift
from random import randint
from scipy.signal import convolve2d
import skimage
from skimage import img_as_float, img_as_ubyte
from skimage.io import imread, imshow
from skimage.color import rgb2gray
import cv2
```

```
img = imread('https://i.ytimg.com/vi/o1YA_6tXs5E/maxresdefault.jpg')
img = skimage.color.rgb2gray(img)
imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f7149e67110>
```



2. Построение гауссовской пирамиды

Функция для вычисления ядра гауссовского фильтра

```
def gauss_filter(g, x, y):
  result = 1/(2*pi*g**2) * exp((-x**2-y**2)/(2*g**2))
  return result

def get_kernel(g):
  r = round(3*g)
  k = int(round(g*6)+1)
  kernel = np.array([[0.0]*k]*k)
  sum = 0
  for x in range(len(kernel)):
    for y in range(len(kernel)):
      kernel[x, y] = gauss_filter(g, x-len(kernel)//2, y-len(kernel)//2)
      sum += kernel[x, y]
  for x in range(len(kernel)):
    for y in range(len(kernel)):
      kernel[x, y] /= sum
  return k, kernel
```

Ниже представлена функция для построения гауссовской пирамиды. Операция свертки выполняется с помощью функции convolve2d (режим 'same' используется для сохранения размера исходного изображения).

```
def get_gauss_pyramid(img, sigma, n_layers):
  k, kernel = get_kernel(sigma)
  new_images = []
  layer = img_as_float(img)
  for _ in range(n_layers):
    temp = convolve2d(layer, kernel, mode='same')
    new_images.append(temp)
    layer = new_images[-1]
  return new_images
```

Для удобства экспериментирования определяем отдельную функцию построения гауссовской пирамиды с параметрами img (изображение, по которому строится пирамида), sigma (параметр гауссовского ядра), n_layers (количество слоев пирамиды), возвращающую списки необходимых изображений.

Функция для представления амплитуды частот изображений пирамиды

```
def freq(img):
    imshow(np.round(np.log(1 + np.abs(fftshift(fft2(img))))).astype('uint8'), cmap='gray')
```

Посмотрим на результаты работы функции с различными значениями σ. Для начала возьмём σ со значением 1.

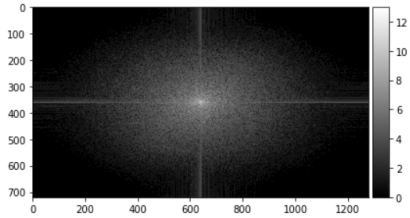
Первый слой:

```
result = get_gauss_pyramid(img, 1, 5)
imshow(result[0])
```

```
<matplotlib.image.AxesImage at 0x7f7149e2a310>
```


```
freq(result[0])
```

```
/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```



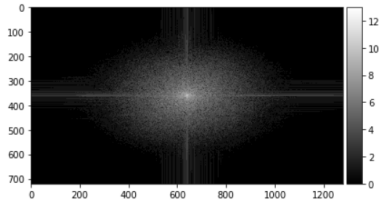
Второй слой:

```
imshow(result[1])
```

```
<matplotlib.image.AxesImage at 0x7f7149b85e50>
```


```
freq(result[1])
```

```
/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```



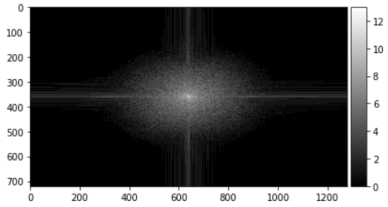
Третий слой:

```
imshow(result[2])
```

```
<matplotlib.image.AxesImage at 0x7f7149af9ad0>
```


```
freq(result[2])
```

```
/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```



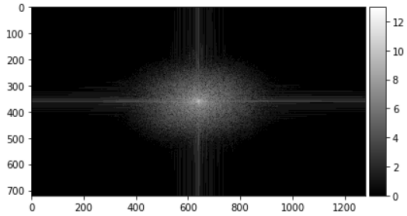
Четвертый слой:

```
imshow(result[3])
```

```
<matplotlib.image.AxesImage at 0x7f7149c1c250>
```


```
freq(result[3])
```

```
/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```



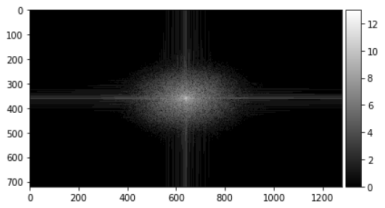
Пятый слой:

```
imshow(result[4])
```

```
<matplotlib.image.AxesImage at 0x7f7149ca95d0>
```


```
freq(result[4])
```

```
/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```



Как видно, с каждым слоем картинка все более размыта.

На карте частот в центре карты – коэффициенты при низких частотах, на окраинах – коэффициенты при высоких частотах. Высокие частоты имеют коэффициенты, близкие к 0, так как на окраинах изображение черное. Так происходит из-за того, что преобразование Фурье от гауссовского фильтра - это гауссиана, которая, в основном, имеет низкие частоты и, следовательно, при свёртке её с некоторым изображении в результирующем изображении останутся, в основном, низкие частоты из исходного изображения.

Посмотрим на изображения при σ = 2.

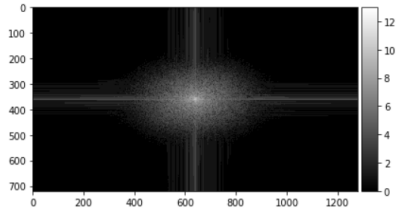
Первый слой:

```
result2 = get_gauss_pyramid(img, 2, 5)
imshow(result2[0])
```

```
<matplotlib.image.AxesImage at 0x7f714a0164d0>
```


```
freq(result2[0])
```

```
/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```



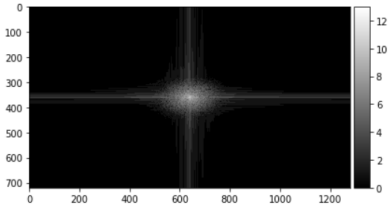
Теперь пятый слой:

```
imshow(result2[4])
```

```
<matplotlib.image.AxesImage at 0x7f71498ba110>
```


```
freq(result2[4])
```

```
/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```



Посмотрим на изображения при σ = 3.

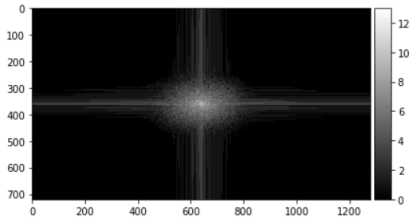
Первый слой:

```
result3 = get_gauss_pyramid(img, 3, 5)
imshow(result3[0])
```

```
<matplotlib.image.AxesImage at 0x7f71497ebe50>
```


```
freq(result3[0])
```

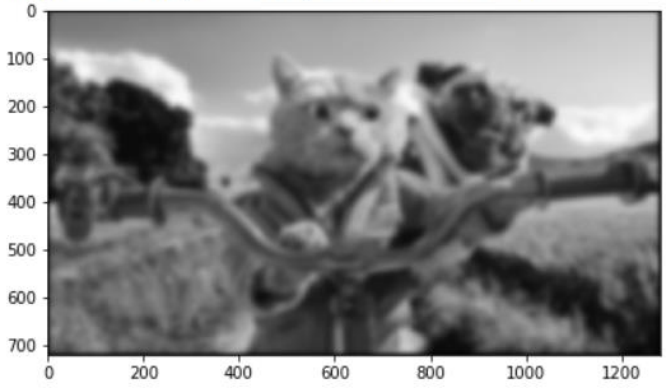
```
/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```



Пятый слой:

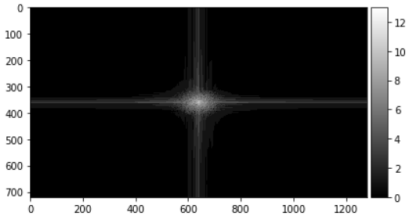
```
imshow(result3[4])
```

```
<matplotlib.image.AxesImage at 0x7f71496a9450>
```




```
freq(result3[4])
```

```
/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```



При увеличении значения sigma в фильтре Гаусса происходит увеличение радиуса ядра фильтра, что приводит к увеличению области размытия и снижению резкости границ объектов на изображении.

Это происходит потому, что фильтр Гаусса используется для размытия изображения путем сглаживания высокочастотных компонентов изображения. Высокочастотные компоненты связаны с резкими переходами между яркостными значениями на изображении (границы объектов), а низкочастотные компоненты представляют собой более плавные переходы.

Увеличение значения sigma приводит к более сильному сглаживанию высокочастотных компонентов и сохранению только низкочастотных компонентов. Это приводит к уменьшению диапазона частот, которые могут быть представлены на изображении, и, следовательно, к сужению диапазона яркостных значений на изображении.

3. Построение лапласовской пирамиды

Предположим, у нас есть два изображения одного и того же размера. Допустим, мы хотим склеить эти два изображения по некоторой маске. Маской мы назовём изображение того же размера, что и исходные, но состоящее из единиц и нулей, где 1 означает, что нужно брать пиксель из первого изображения, а 0 означает - что из второго.

Задача состоит в том, чтобы склеить два изображения по маске - создать коллаж.

Если мы напрямую возьмём пиксели по маске и составим из них новое изображение, то в результате переход между изображениями в нём будет очень резким.

Нам хотелось бы в этом коллаже получить более плавный переход между двумя изображениями. Такой переход можно получить с помощью лапласовской пирамиды. Посмотрим, как она строится:

```python
def get_laplace_pyramid(img, sigma, n_layers):
  gauss_images = get_gauss_pyramid(img, sigma, n_layers)
  new_images = []
  temp = img_as_float(img)
  layer = gauss_images[0]
  for i in range(len(gauss_images)-1):
    new_images.append(temp-layer)
    temp = layer
    layer = gauss_images[i+1]
  new_images.append(gauss_images[-1])
  return new_images
```

Пусть у нас есть исходное изображение I. Сначала будем строить из него гауссовскую пирамиду. Гауссовской пирамидой называется последовательность изображений - I1, I2, I3, ... - которые получаются при размытии исходного изображения с помощью гауссовского фильтра.

Теперь из гауссовской пирамиды получим лапласовскую пирамиду. Она строится следующим образом.

1. Из изображения I вычитаем изображение I1 - получаем новое изображение, которое назовём L1.

2. Из изображения I1 вычитаем изображение I2 - получаем изображение L2.

3. Из изображения I2 вычитаем изображение I3 - получаем изображение L3 и т.д.

Последовательность этих изображений - L1, L2, L3, ... - называется лапласовской пирамидой.

Когда мы для построения изображения L1 лапласовской пирамиды вычитаем из изображения I изображение I1, вычитание в пространстве изображений приводит к такому же вычитанию в пространстве частот. Таким образом, если из амплитуды для I, где есть все частоты, мы вычитаем амплитуду для I1, где нет части высоких частот, мы получаем только те высокие частоты, которые были выброшены из I при преобразовании его в I1. Так что амплитуда частот для L1 будет выглядеть как амплитуда для I, но с большим выброшенным кружком в середине. Затем из амплитуды изображения для I1

мы вычитаем амплитуду для I2 и получаем концентрическую окружность - некоторую полосу из средних частот. Затем, когда из амплитуды для I2 мы вычитаем амплитуду для I3, мы снова получаем концентрическую окружность, но уже меньшего радиуса - некоторую полосу частот, но уже более низких.

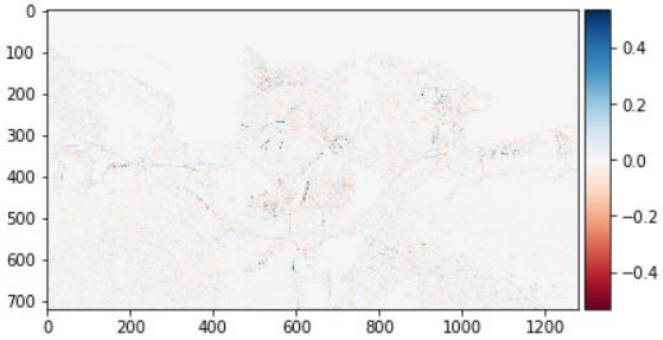
Добавим к лапласовской пирамиде самое последнее изображение - в данном случае, I3. Получается, что если мы сложим изображения I3, L3, L2 и L1, то получим исходное изображение. Таким образом, лапласовская пирамида - это разбиение изображения на непересекающиеся полосы частот.

Посмотрим на результаты работы функции с различными значениями σ. Для начала возьмём σ=1:

Первый слой:

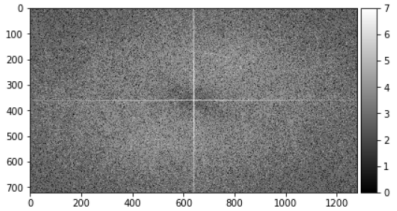
```
res = get_laplace_pyramid(img, 1, 5)
imshow(res[0])
```

```
<matplotlib.image.AxesImage at 0x7f714956c110>
```




```
freq(res[0])
```

```
/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```



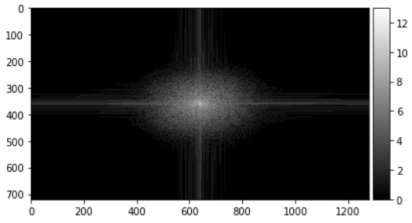
Пятый слой:

```
imshow(res[4])
```

```
<matplotlib.image.AxesImage at 0x7f71498140d0>
```

```
freq(res[4])
```

```
/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```

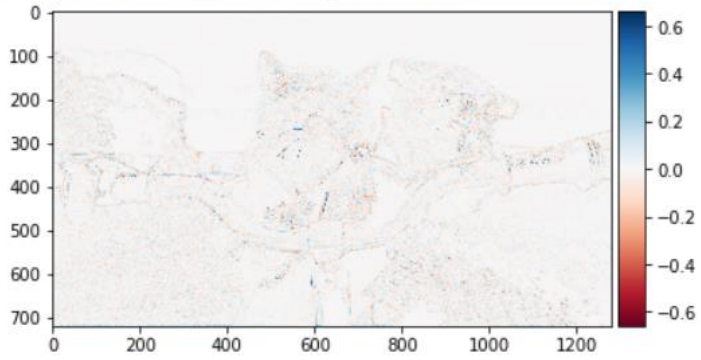


Возьмем σ = 2.

Первый слой:

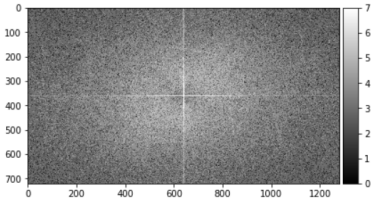
```
res2 = get_laplace_pyramid(img, 2, 5)
imshow(res2[0])
```

```
<matplotlib.image.AxesImage at 0x7f7149d29c10>
```



```
freq(res2[0])
```

```
/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```



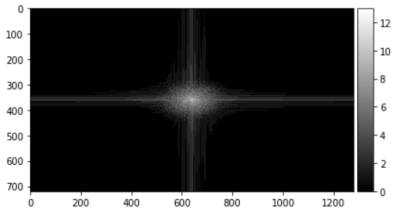
Пятый слой:

```
imshow(res2[4])
```

```
<matplotlib.image.AxesImage at 0x7f714975cd50>
```


```
freq(res2[4])
```

```
/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```

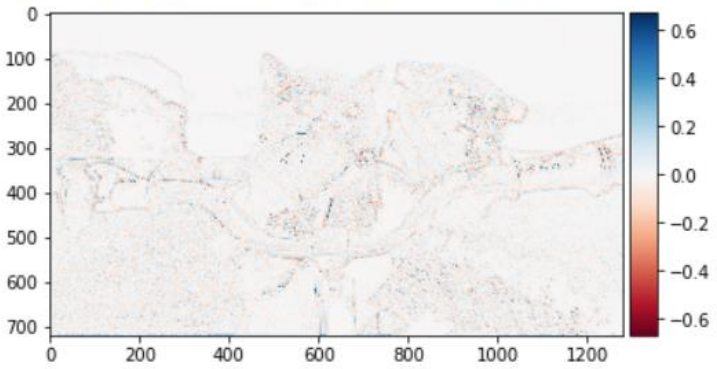


Возьмем σ = 3.

Первый слой:

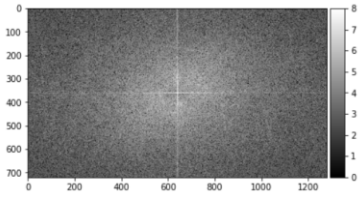
```
res3 = get_laplace_pyramid(img, 3, 5)
imshow(res3[0])
```

```
<matplotlib.image.AxesImage at 0x7f71494374d0>
```




```
freq(res3[0])
```

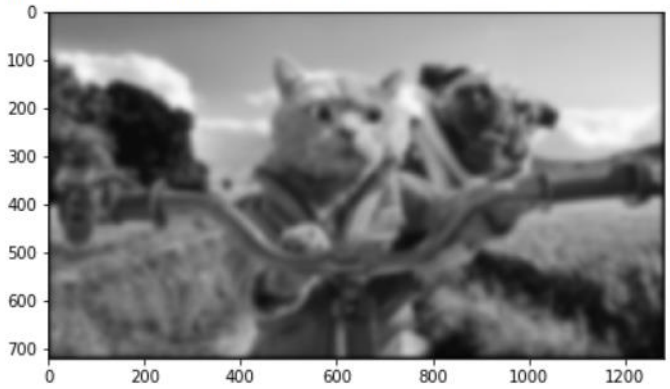
```
/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```



Пятый слой:

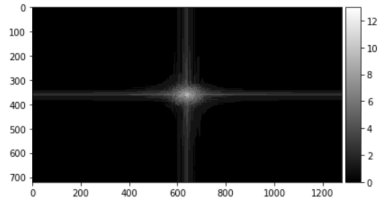
```
imshow(res3[4])
```

```
<matplotlib.image.AxesImage at 0x7f71492c26d0>
```




```
freq(res3[4])
```

```
/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```



4. Склейка изображений

У нас есть два изображения. В этих изображениях присутствуют как низкие, так и высокие частоты. При этом граница, по которой мы хотим склеить изображения, является высокочастотной.

Мы хотели бы, чтобы в склеенном изображении переход между двумя исходными изображениями был более плавным, то есть в нашем случае мы хотим, чтобы в маске слева были единицы, справа - нули, а в середине - какие-то промежуточные значения - от 1 до 0 и 0,5 посередине.

Посмотрим, как мы можем смешивать различные частоты изображений с помощью различных масок. Смотрим на алгоритм склейки.

Мы строим лапласовские пирамиды LA и LB для изображений и гауссовскую пирамиду GM - для маски, а затем комбинируем их по формуле:

$$LS = GM * LA + (1 - GM) * LB$$

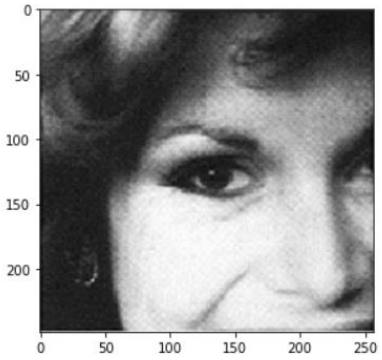
Умножение и сложение в формуле имеется ввиду поэлементное.

Для построения гауссовской пирамиды достаточно применять свёртку с гауссовским фильтром, а для построения лапласовской - из каждого более верхнего слоя гауссовской пирамиды вычитать следующий слой гауссовской пирамиды. Затем мы применяем вышеприведённую формулу для каждого из слоёв полученных пирамид. В результате получим лапласовскую пирамиду склеенного изображения. Само склеенное изображение получаем из этой лапласовской пирамиды, суммируя все её изображения.

Загрузим изображения для тестирования:

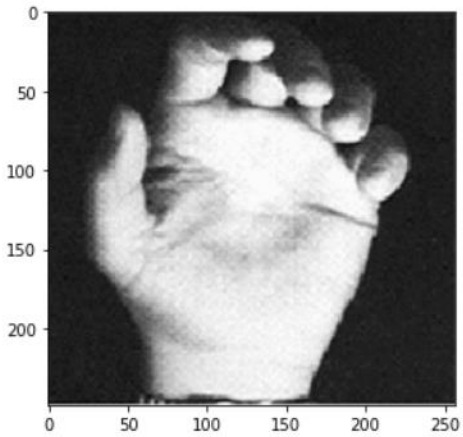
```
img1 = rgb2gray(imread('https://stepik.org/media/attachments/lesson/58410/a.png'))
imshow(img1)
```

```
<matplotlib.image.AxesImage at 0x7f71494a5410>
```



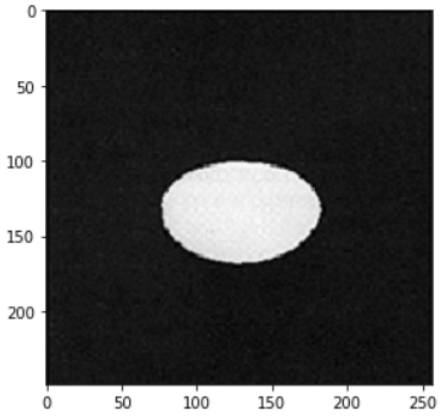

```
img2 = rgb2gray(imread('https://stepik.org/media/attachments/lesson/58410/b.png'))
imshow(img2)
```

```
<matplotlib.image.AxesImage at 0x7f71499a9050>
```




```
mask = rgb2gray(imread('https://stepik.org/media/attachments/lesson/58410/mask.png'))
imshow(mask)
```

```
<matplotlib.image.AxesImage at 0x7f7149d15150>
```



Функция для склейки изображений:

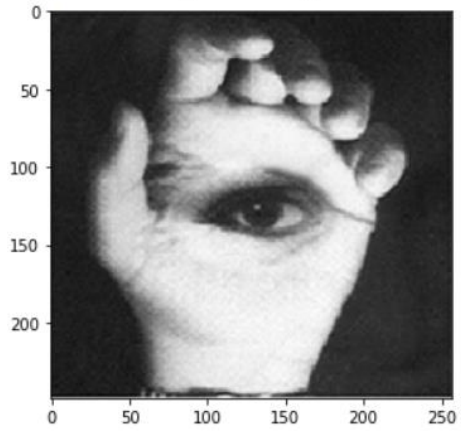
```
def merge(img1, img2, mask, sigma, n_layers):
  img1_lap = get_laplace_pyramid(img1, sigma, n_layers)
  img1_lap.reverse()
  mask_gauss = get_gauss_pyramid(mask, sigma, n_layers)
  mask_gauss.reverse()
  img2_lap = get_laplace_pyramid(img2, sigma, n_layers)
  img2_lap.reverse()
  arr = np.array([0]*mask_gauss[0])
  for i in range(len(mask_gauss)):
    arr += img1_lap[i] * mask_gauss[i] + img2_lap[i] * (1 - mask_gauss[i])
  return arr
```

Посмотрим на результат работы функции при различных σ:

σ = 1:

```
img_out = img_as_ubyte(np.clip(merge(img1, img2, mask, 1, 5), -1, 1))
imshow(img_out)
```

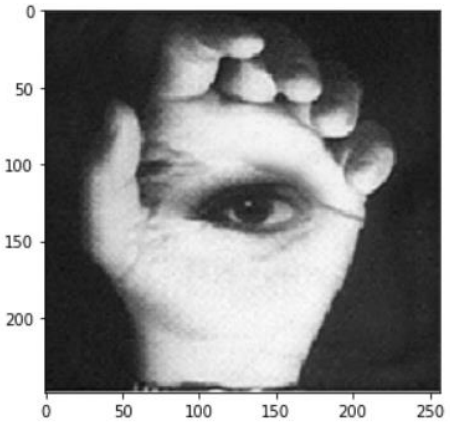
```
<matplotlib.image.AxesImage at 0x7f7149d05410>
```



σ = 2:

```
img_out = img_as_ubyte(np.clip(merge(img1, img2, mask, 2, 5), -1, 1))
imshow(img_out)
```

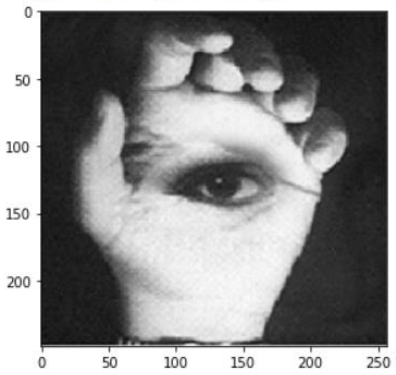
```
<matplotlib.image.AxesImage at 0x7f714986ed10>
```



σ = 3:

```
img_out = img_as_ubyte(np.clip(merge(img1, img2, mask, 3, 5), -1, 1))
imshow(img_out)
```

```
<matplotlib.image.AxesImage at 0x7f71495eb910>
```

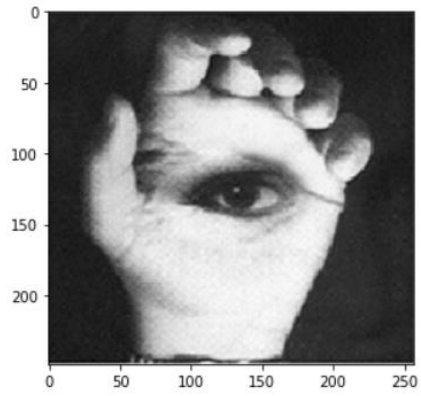


Теперь изменим количество слоёв при σ = 1:

Первый слой:

```
img_out = img_as_ubyte(np.clip(merge(img1, img2, mask, 1, 10), -1, 1))
imshow(img_out)
```

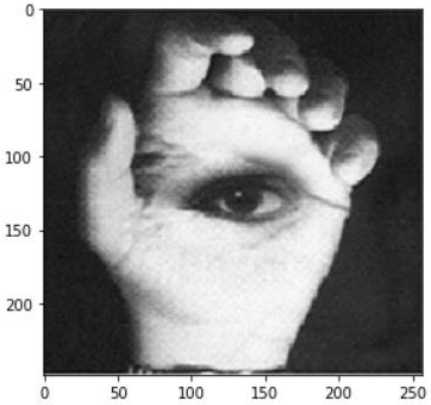
```
<matplotlib.image.AxesImage at 0x7f7149a95190>
```



Второй слой:

```
img_out = img_as_ubyte(np.clip(merge(img1, img2, mask, 1, 15), -1, 1))
imshow(img_out)
```

```
<matplotlib.image.AxesImage at 0x7f71491ed690>
```



Выводы

Преобразование Фурье используется для анализа частотных компонент изображения и может быть использовано для определения, какие компоненты изображения являются низкочастотными или высокочастотными. Это позволяет оптимизировать фильтрацию изображения в гауссовской пирамиде, чтобы сохранить только нужные компоненты изображения.

С помощью операции свертки с гауссовским фильтром, поэлементного вычитания, поэлементного умножения с маской и поэлементного сложения, для того чтобы из лапласовской пирамиды сделать финальное изображение, мы получили качественный результат склейки изображений.