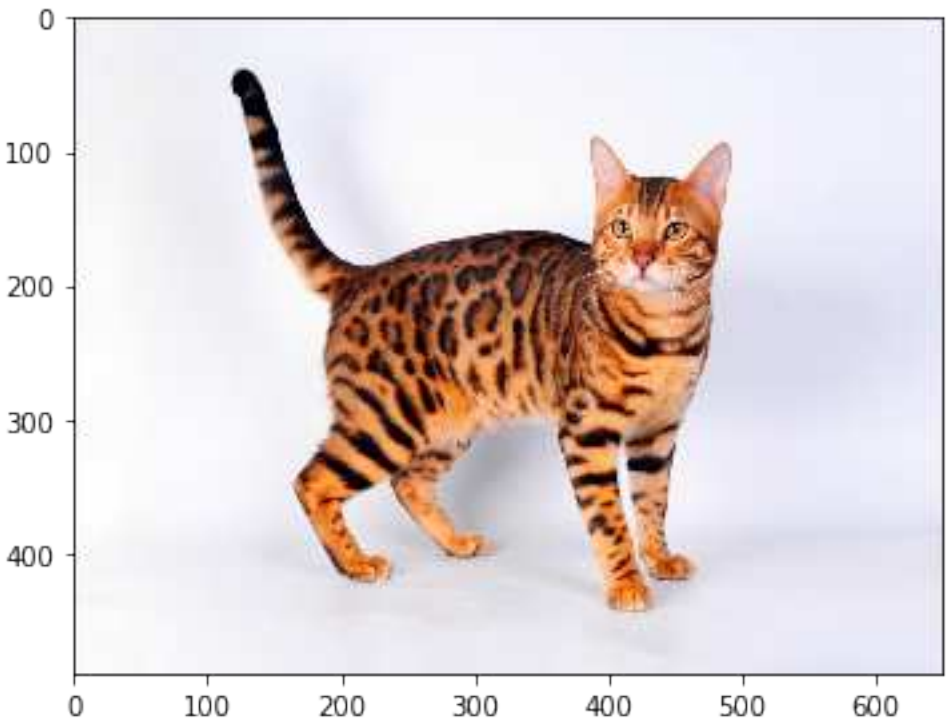
stitching

10 января 2018 г.

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

```
In [1]: from skimage.io import imread, imshow
        from skimage import img_as_float
        from skimage.filters import gaussian
        import matplotlib.pyplot as plt
        %matplotlib inline
```

```
In [2]: img = img_as_float(imread('cat.jpg'))
        imshow(img);
```

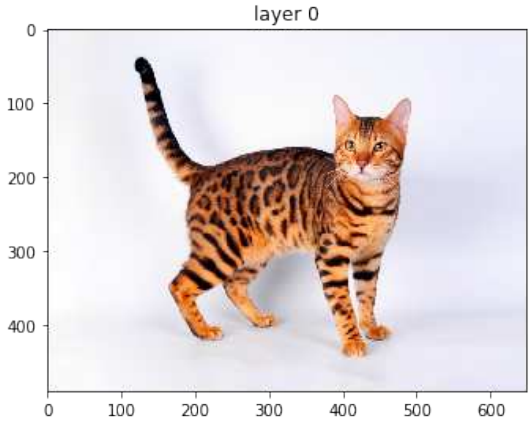


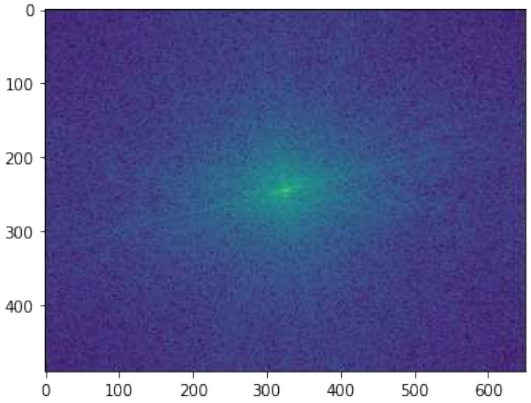
2. (5 баллов) Постройте гауссовскую пирамиду изображения из не менее чем пяти слоев. Визуализируйте полученные изображения и амплитуды частот изображений пирамиды (код

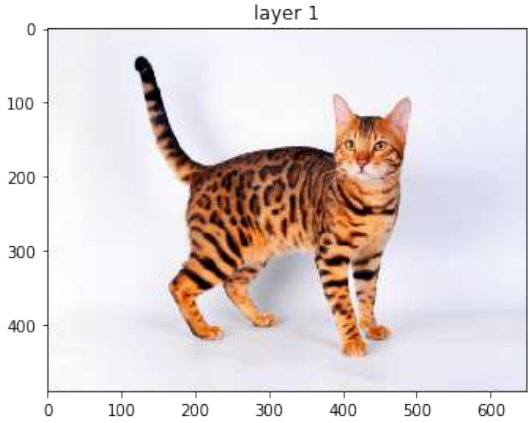
можно найти в видео про преобразование Фурье) и убедитесь, что на каждом слое диапазон частот сужается. Постройте пирамиду для трех различных значения сигмы гауссовского ядра. Для удобства экспериментирования определите отдельную функцию построения гауссовской пирамиды с параметрами `img` (изображение, по которому строится пирамида), `sigma` (параметр гауссовского ядра), `n_layers` (количество слоев пирамиды), возвращающую списки необходимых изображений.

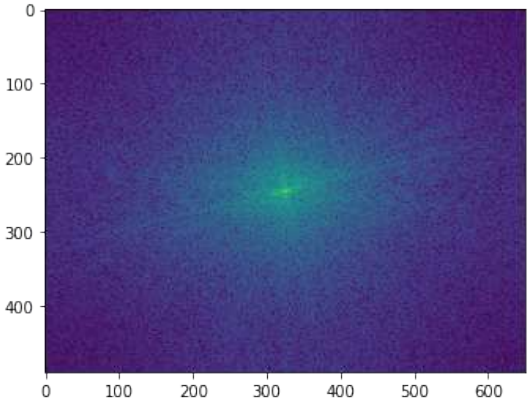
In [3]:
```
import numpy as np
from numpy.fft import fft2, fftshift

def build_gaussian_pyramid(img, sigma, n_layers):
    pyr = [img.copy()]
    for n in range(n_layers):
        img = gaussian(pyr[-1], sigma, multichannel=True)
        pyr.append(img)
    return pyr

def visualize_pyramid(pyr, figsize=(12, 6)):
    for i, layer in enumerate(pyr):
        if layer.ndim == 3:
            gray = layer[..., 1]
        else:
            gray = layer

        plt.figure(figsize=figsize)
        plt.subplot(121)
        if layer.ndim == 3 and (layer.min() < 0 or layer.max() > 1):
            plt.imshow(gray)
        else:
            plt.imshow(layer)
        plt.title(f'layer {i}')

        plt.subplot(122)
        freq = np.log(1 + np.abs(fftshift(fft2(gray))))
        plt.imshow(freq)
        plt.clim([0, 12])
    plt.show()
```

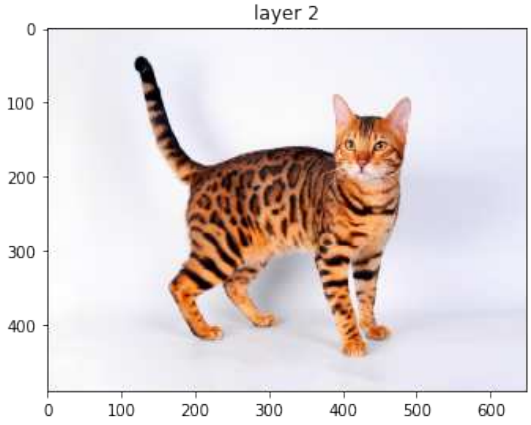
In [4]:
```
pyr = build_gaussian_pyramid(img, 0.5, 5)
visualize_pyramid(pyr)
```

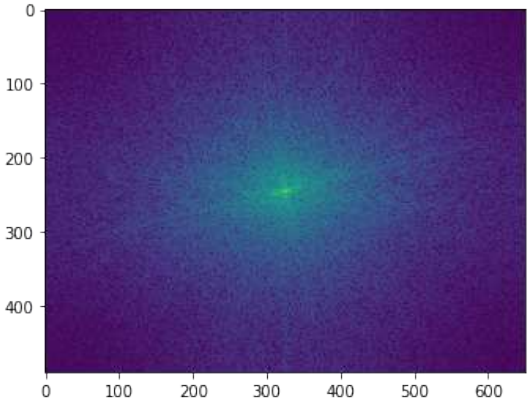

layer 0
0
100
200
300
400
0
100
200
300
400
500
600

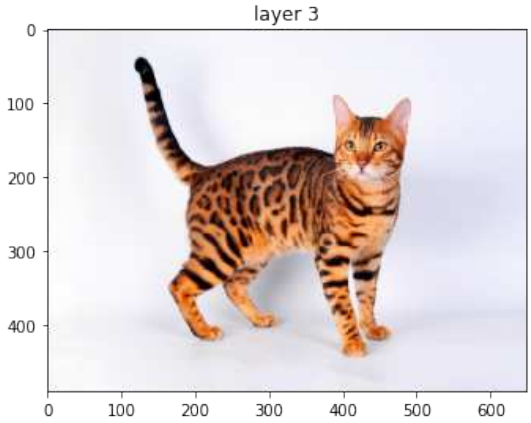

0
100
200
300
400
0
100
200
300
400
500
600

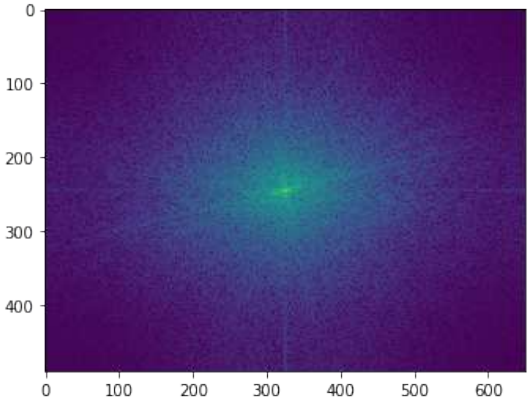

layer 1
0
100
200
300
400
0
100
200
300
400
500
600

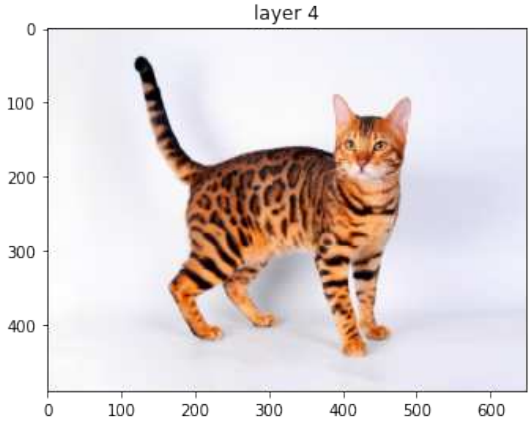

0
100
200
300
400
0
100
200
300
400
500
600

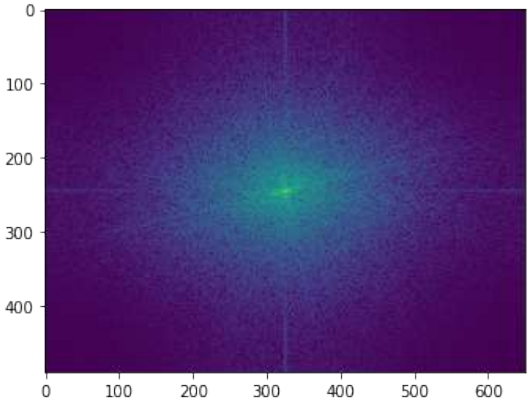

layer 2
0
100
200
300
400
0
100
200
300
400
500
600

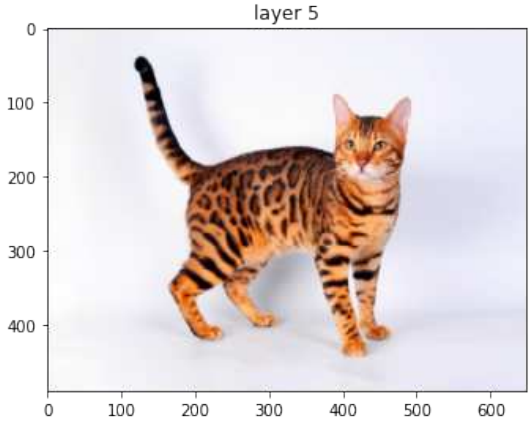

0
100
200
300
400
0
100
200
300
400
500
600

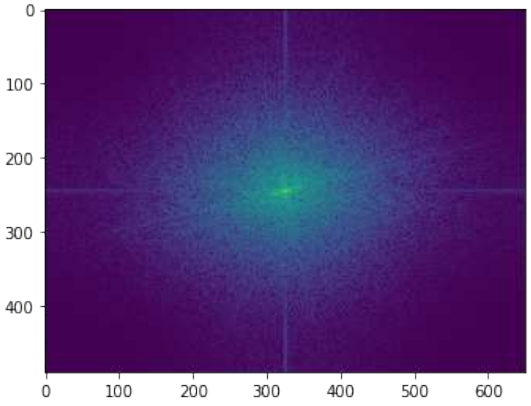

layer 3
0
100
200
300
400
0
100
200
300
400
500
600


0
100
200
300
400
0
100
200
300
400
500
600

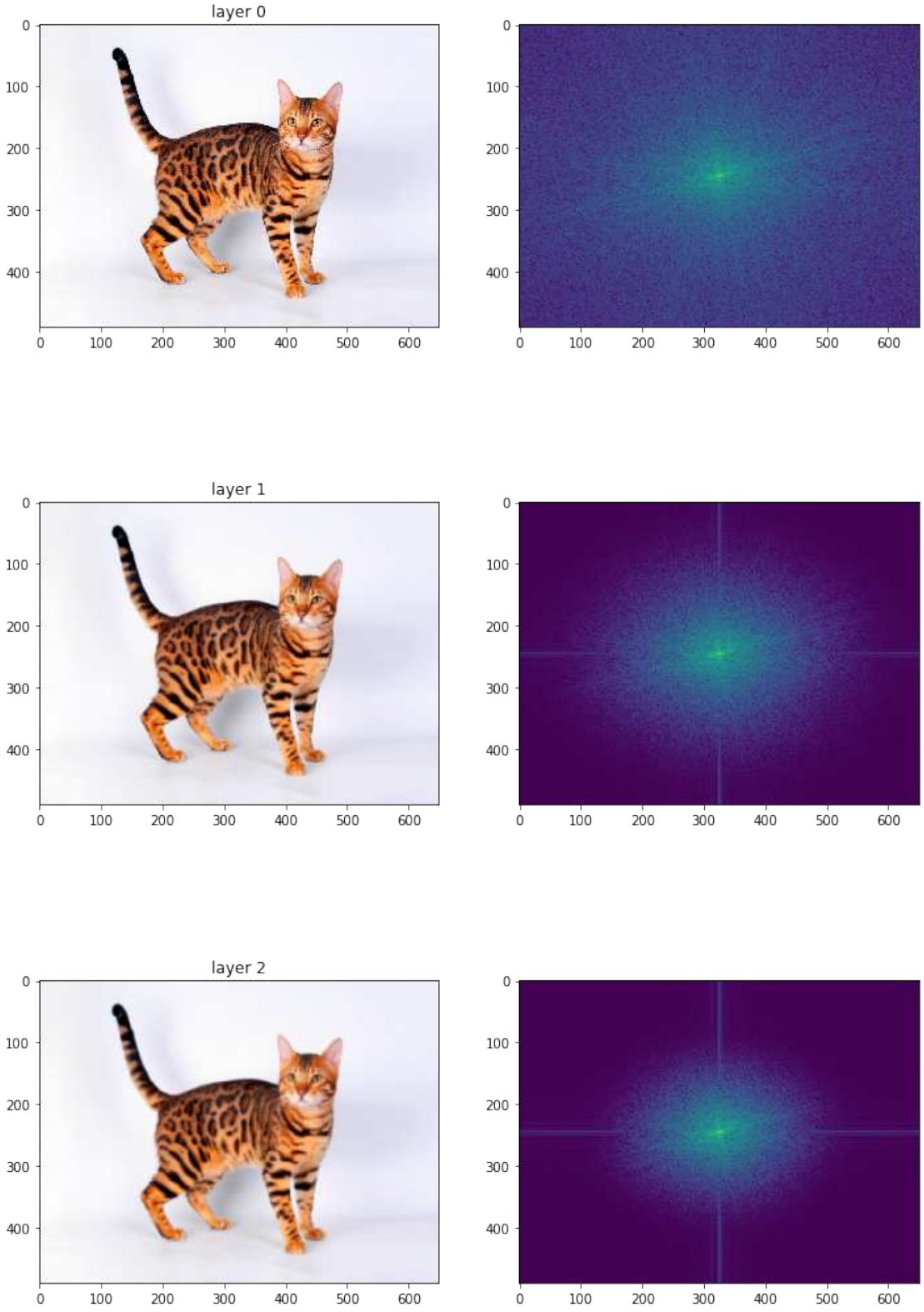

layer 4
0
100
200
300
400
0
100
200
300
400
500
600

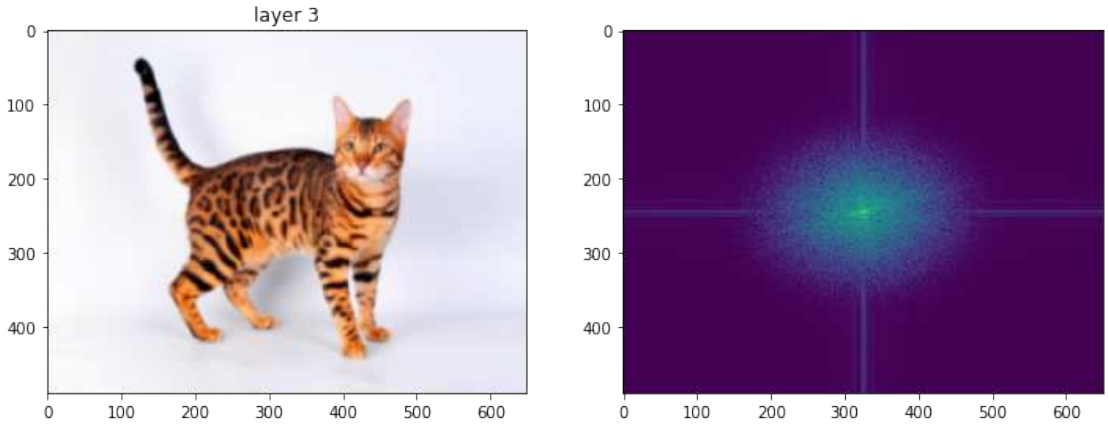

0
100
200
300
400
0
100
200
300
400
500
600

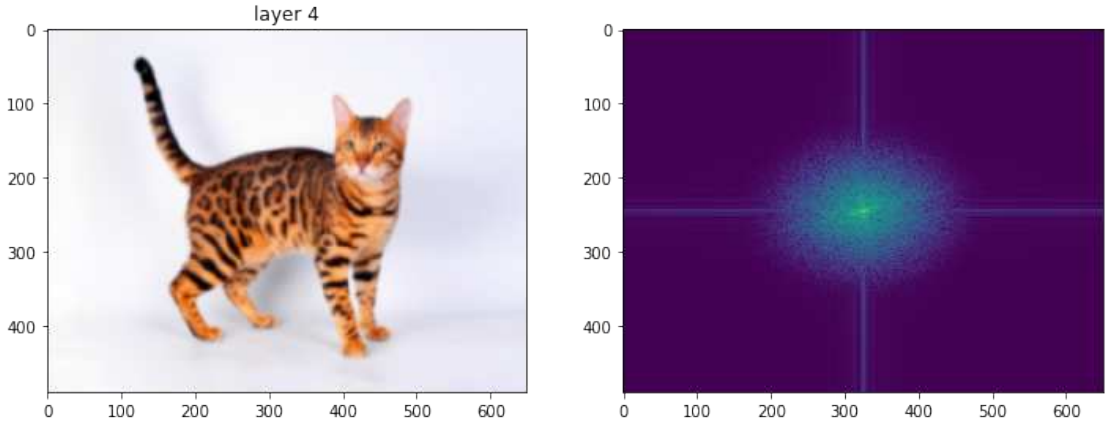

layer 5
0
100
200
300
400
0
100
200
300
400
500
600

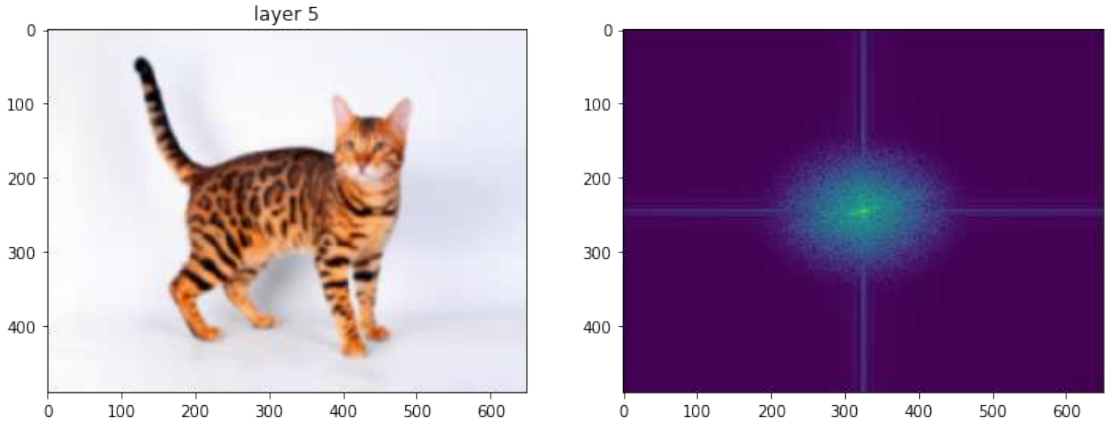

0
100
200
300
400
0
100
200
300
400
500
600

In [5]:
```
pyr = build_gaussian_pyramid(img, 1.0, 5)
visualize_pyramid(pyr)
```

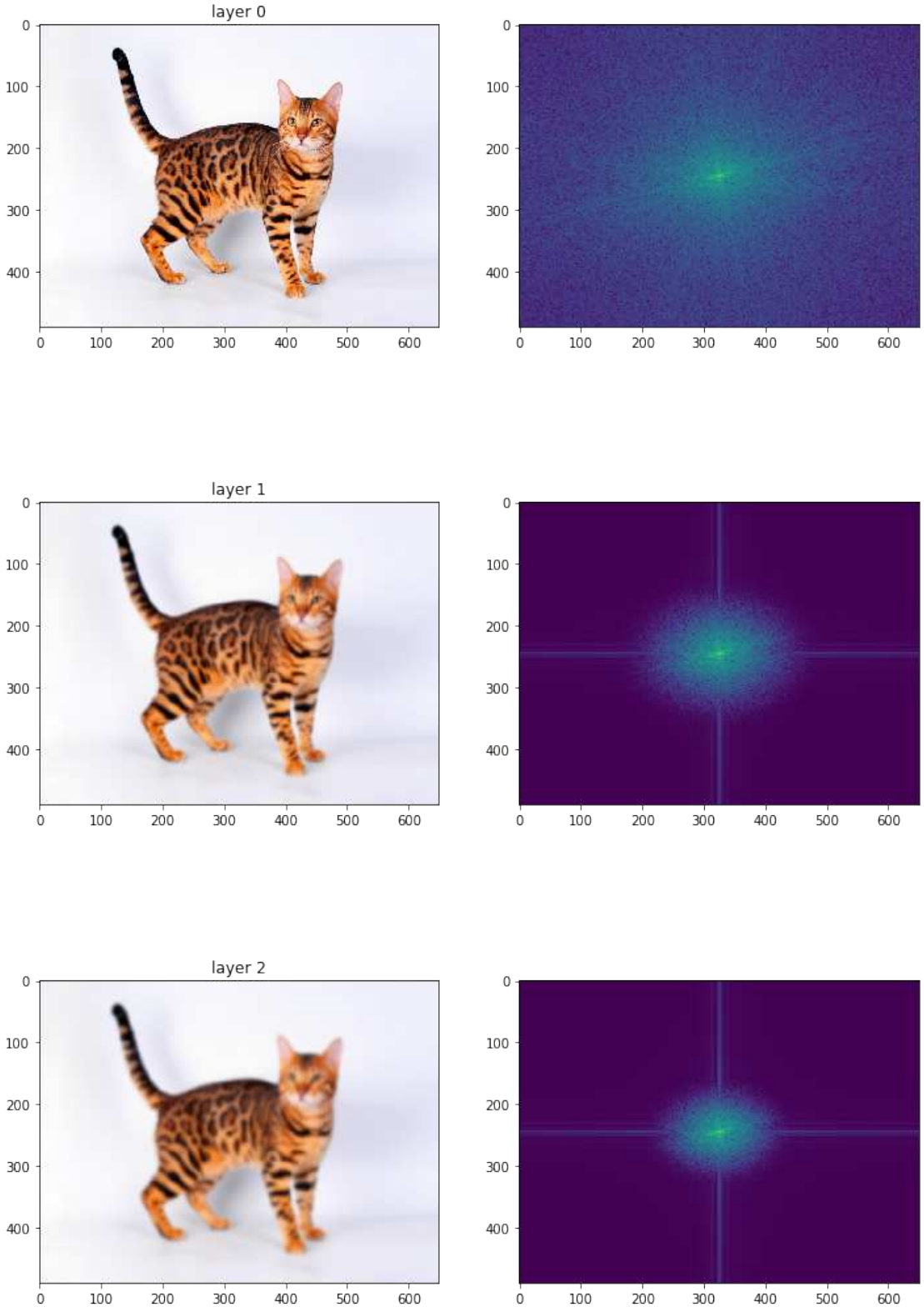


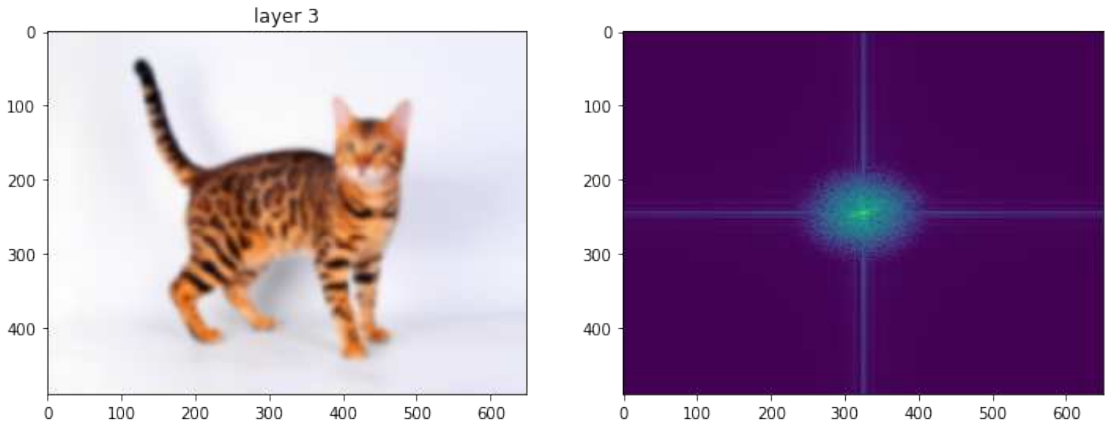

layer 3
0
100
200
300
400
0
100
200
300
400
500
600
0
100
200
300
400
0
100
200
300
400
500
600

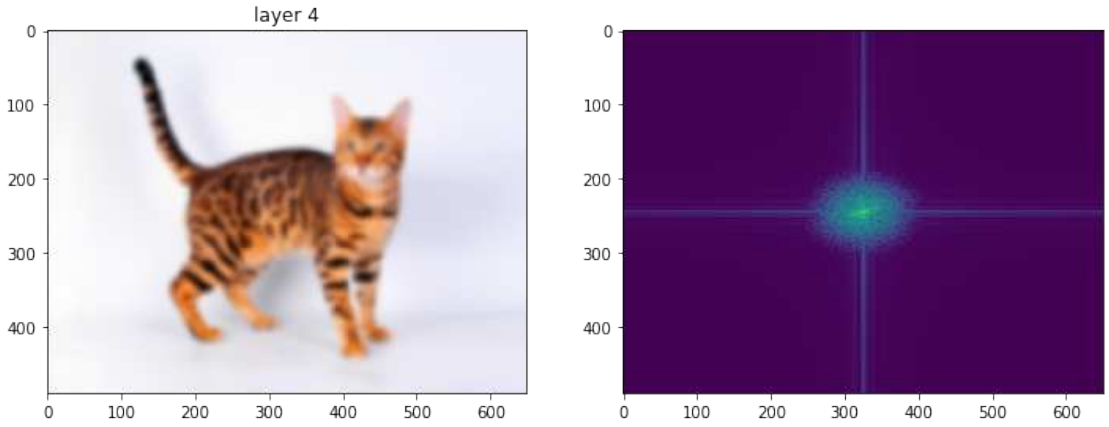

layer 4
0
100
200
300
400
0
100
200
300
400
500
600
0
100
200
300
400
0
100
200
300
400
500
600

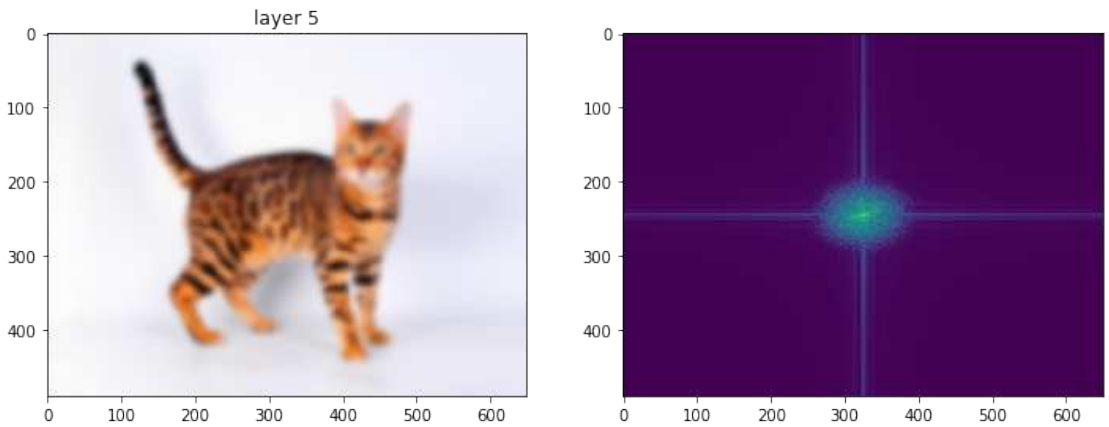

layer 5
0
100
200
300
400
0
100
200
300
400
500
600
0
100
200
300
400
0
100
200
300
400
500
600

In [6]:
```
pyr = build_gaussian_pyramid(img, 2.0, 5)
visualize_pyramid(pyr)
```




layer 3
0
100
200
300
400
0
100
200
300
400
500
600

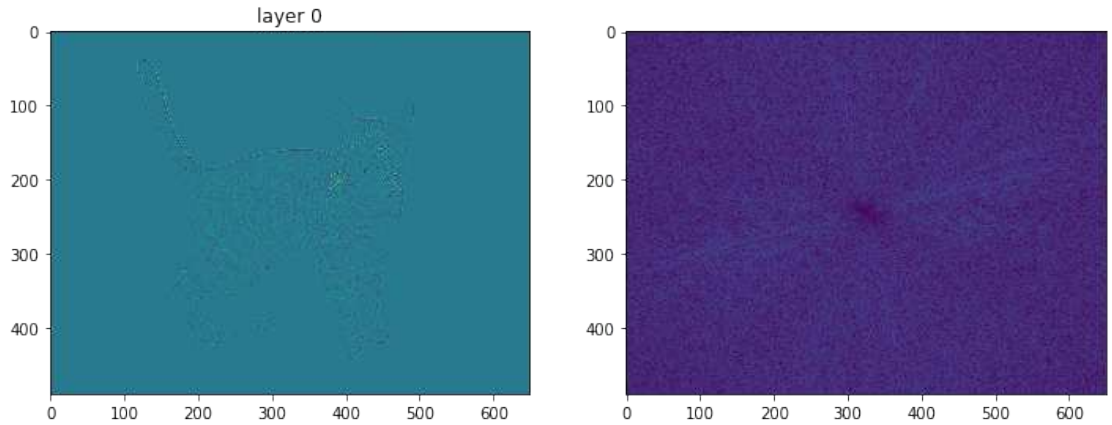

layer 4
0
100
200
300
400
0
100
200
300
400
500
600

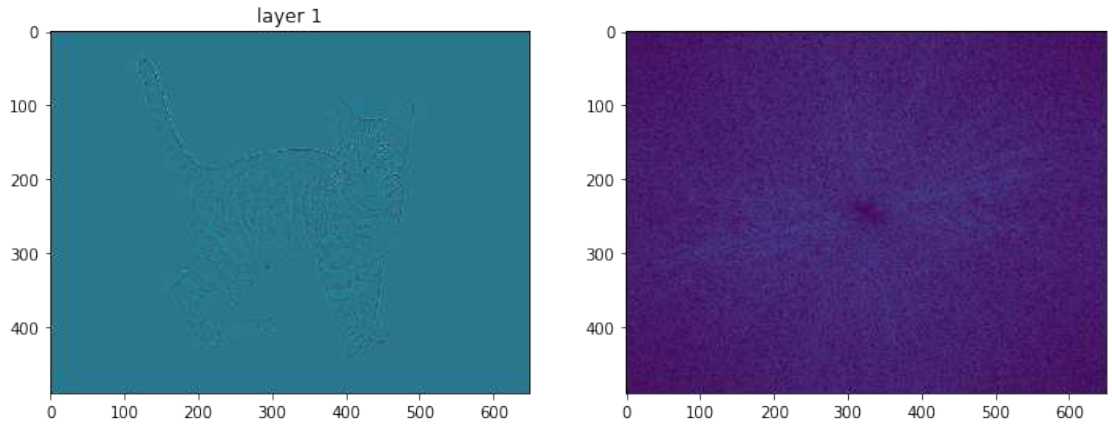

layer 5
0
100
200
300
400
0
100
200
300
400
500
600

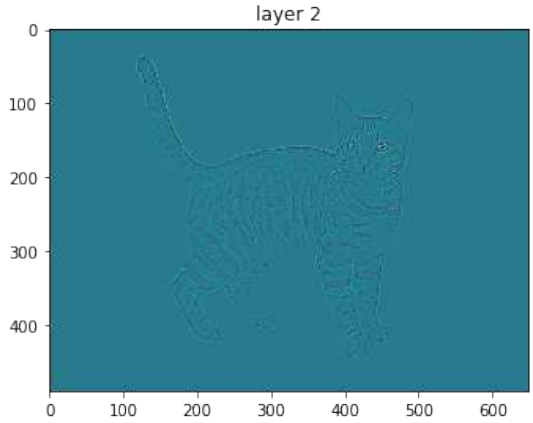
3. (5 баллов) Проведите аналогичные эксперименты с лапласовской пирамидой. Функция для построения лапласовской пирамиды должна использовать функцию построения гауссовской пирамиды и иметь, как и функция гауссовской пирамиды, параметры `img`, `sigma` и `n_layers`.

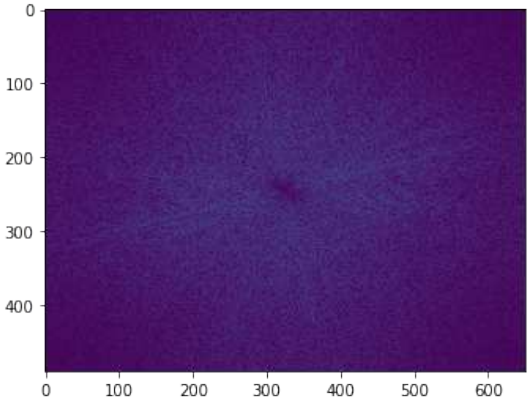
```
In [7]: def build_laplacian_pyramid(img, sigma, n_layers):
            pyr_g = build_gaussian_pyramid(img, sigma, n_layers)
            pyr = []
            for img_1, img_2 in zip(pyr_g, pyr_g[1:]):
                pyr.append(img_1 - img_2)
            pyr.append(pyr_g[-1])
            return pyr
```

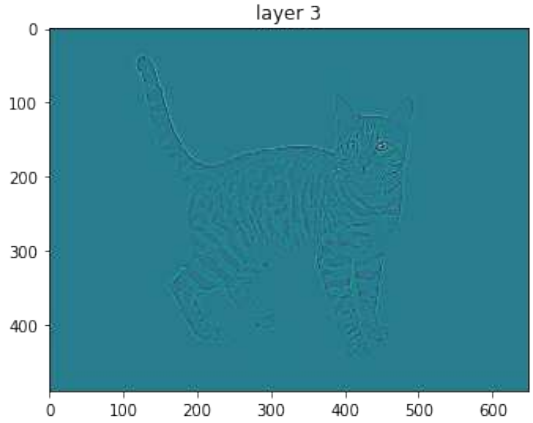
```
In [8]: pyr = build_laplacian_pyramid(img, 0.5, 5)
        visualize_pyramid(pyr)
```

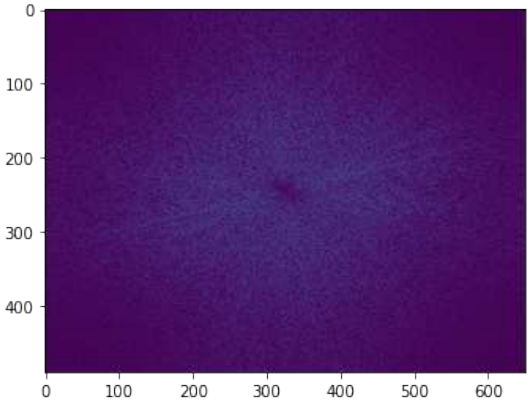


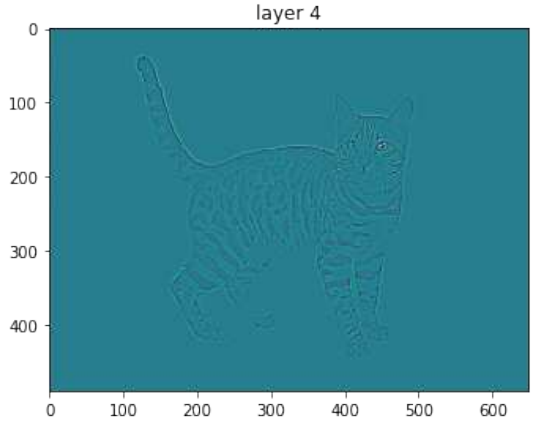


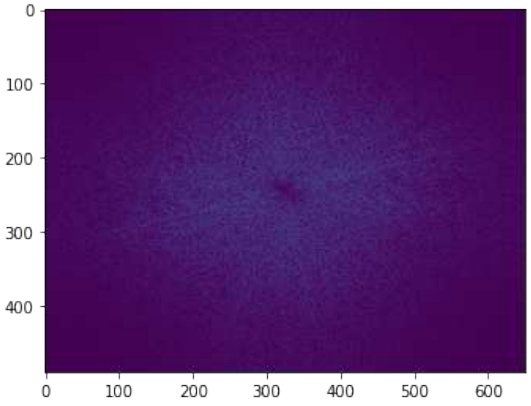

layer 2
0
100
200
300
400
0
100
200
300
400
500
600

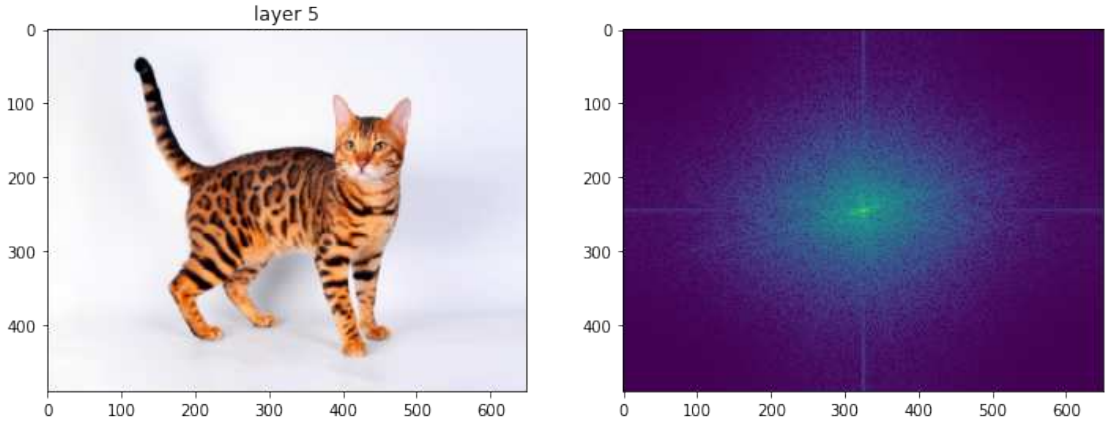

0
100
200
300
400
0
100
200
300
400
500
600


layer 3
0
100
200
300
400
0
100
200
300
400
500
600

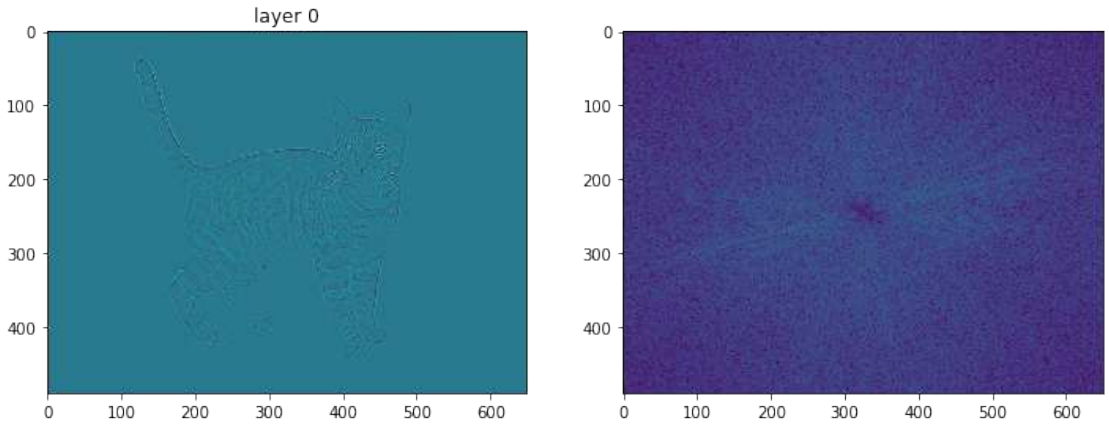

0
100
200
300
400
0
100
200
300
400
500
600

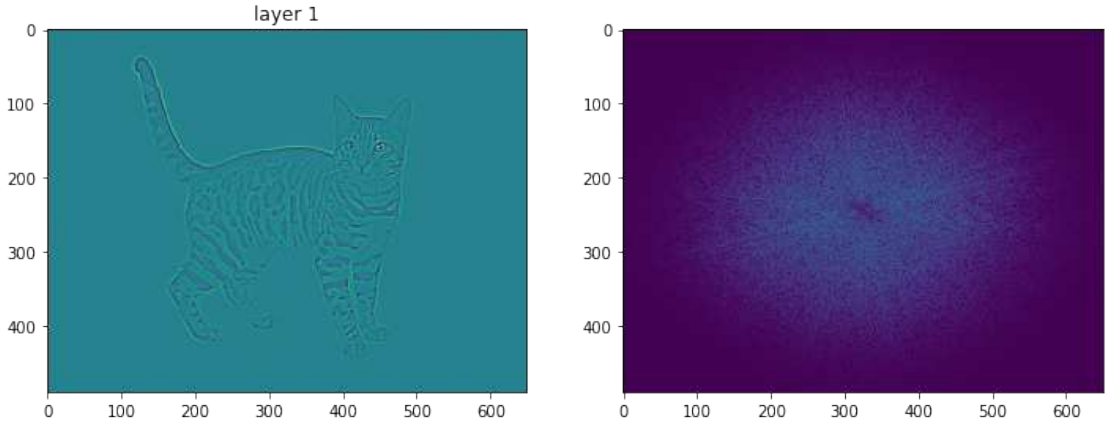

layer 4
0
100
200
300
400
0
100
200
300
400
500
600

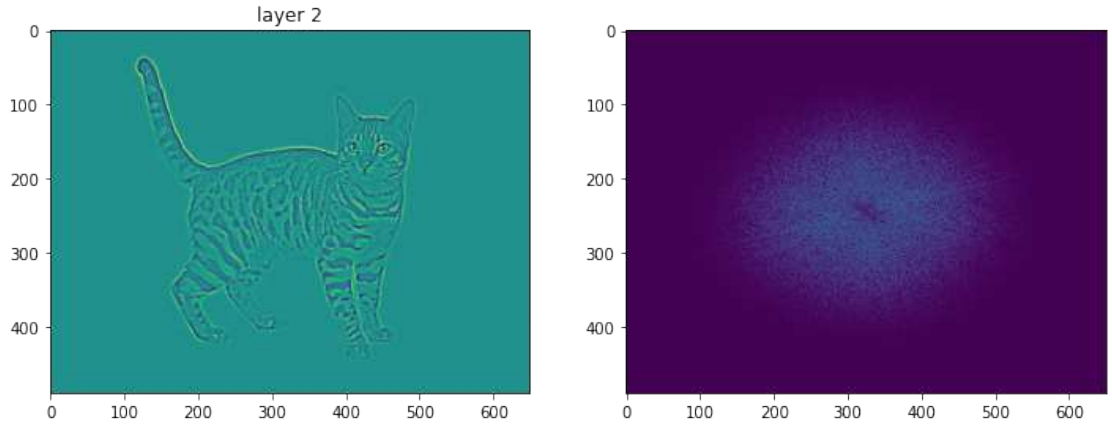

0
100
200
300
400
0
100
200
300
400
500
600

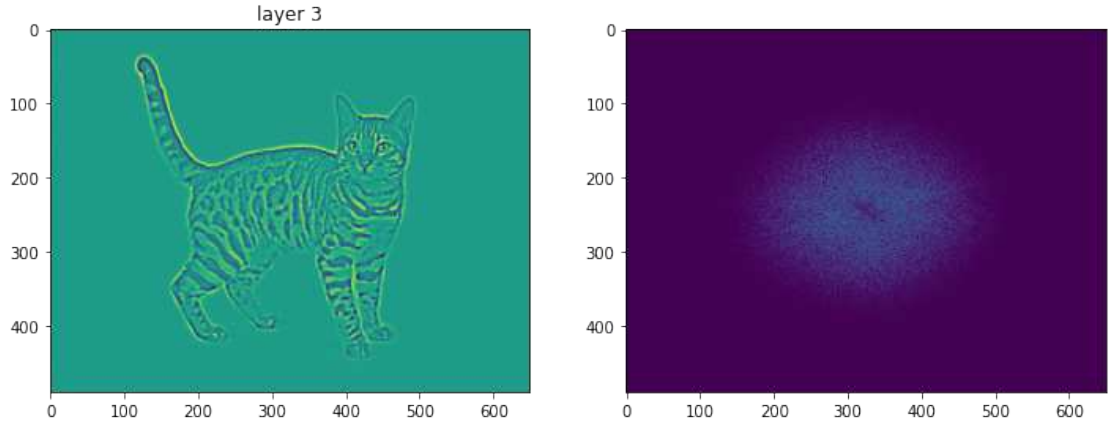


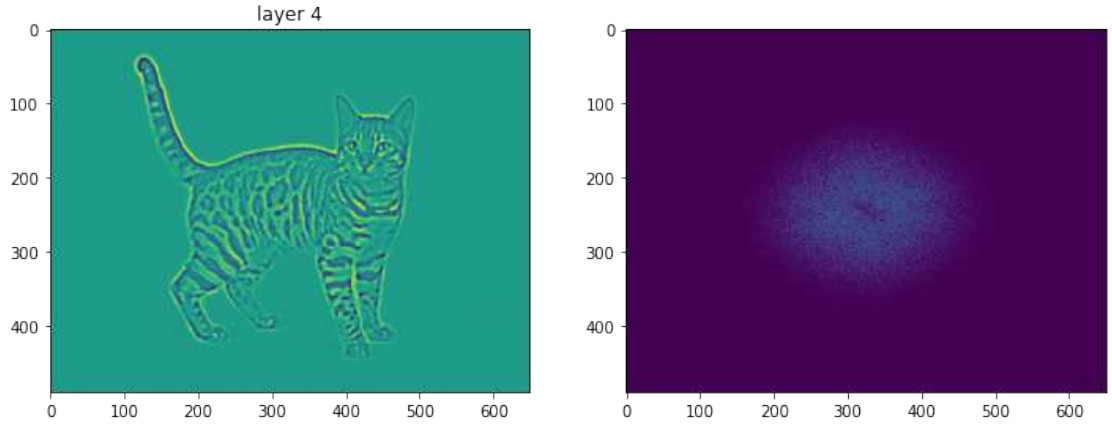
In [9]:
```
pyr = build_laplacian_pyramid(img, 1, 5)
visualize_pyramid(pyr)
```

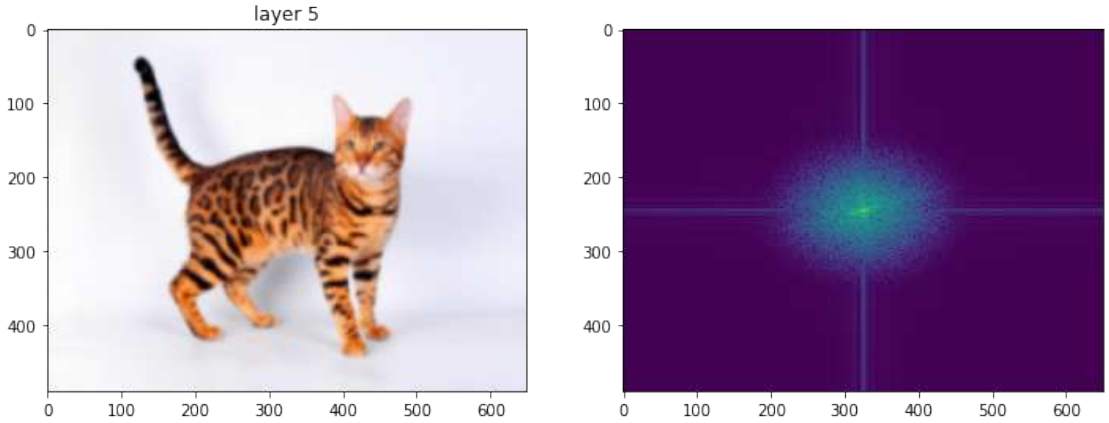




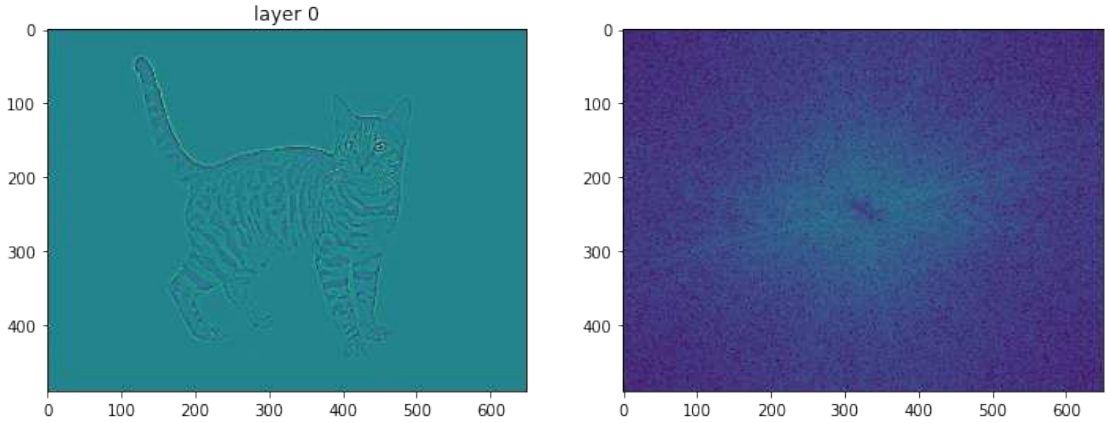

layer 2
0
100
200
300
400
0
100
200
300
400
500
600

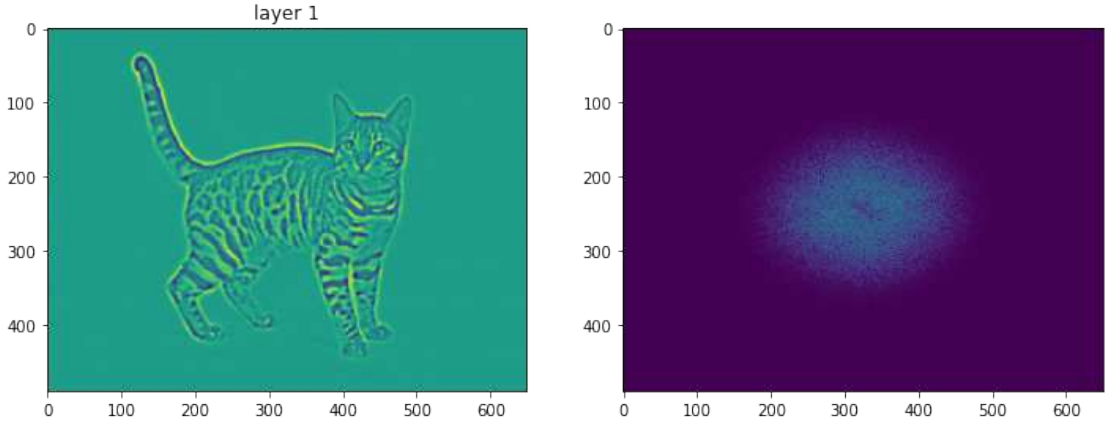

layer 3
0
100
200
300
400
0
100
200
300
400
500
600

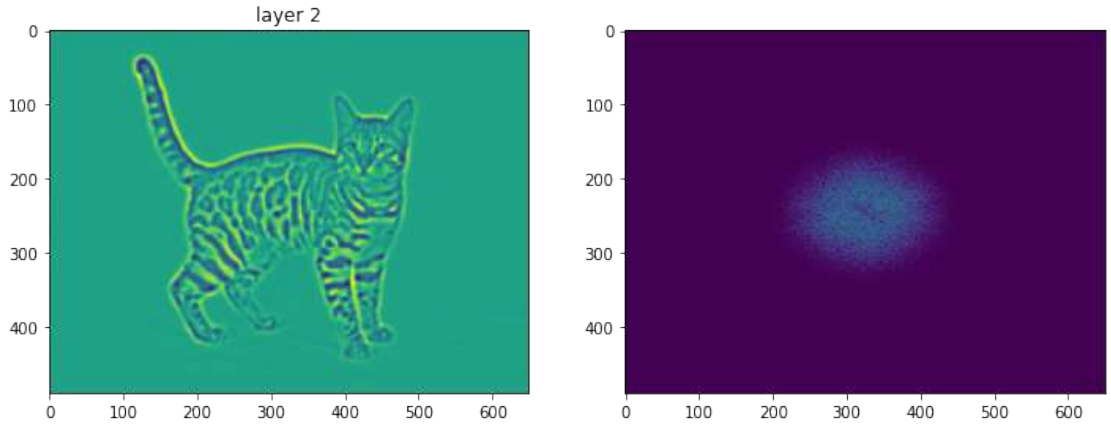

layer 4
0
100
200
300
400
0
100
200
300
400
500
600

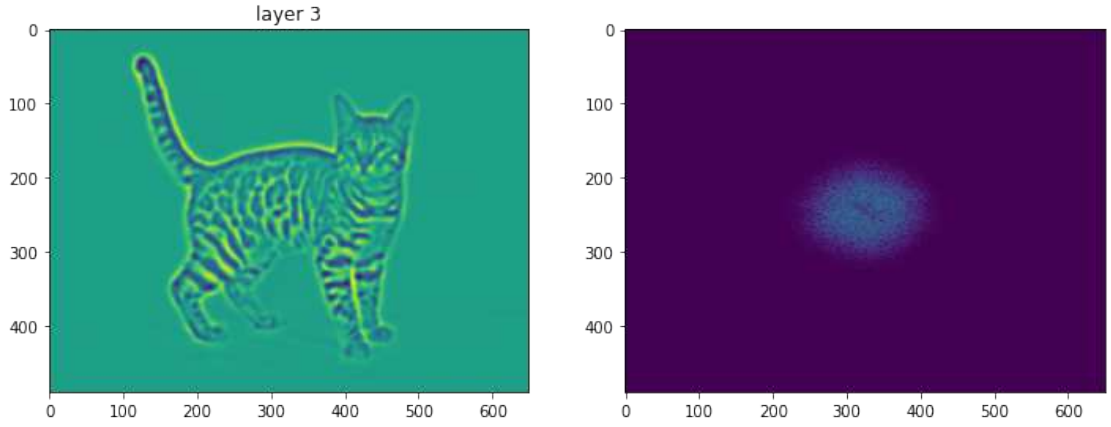


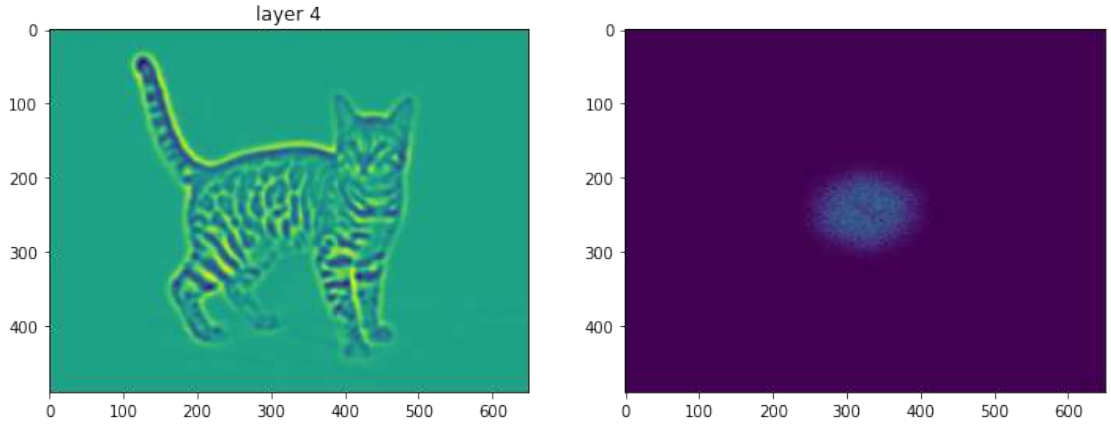

```
In [10]: pyr = build_laplacian_pyramid(img, 2, 5)
         visualize_pyramid(pyr)
```

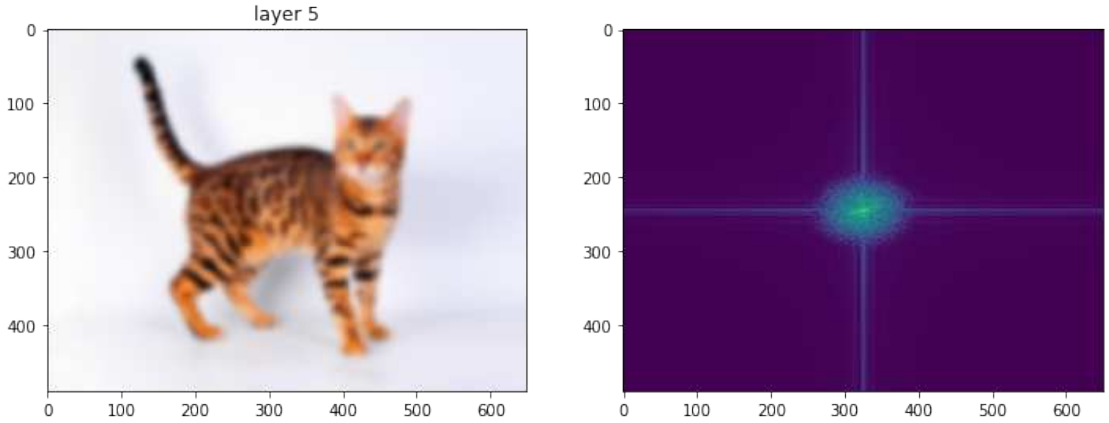





layer 2
0
100
200
300
400
0
100
200
300
400
500
600


layer 3
0
100
200
300
400
0
100
200
300
400
500
600


layer 4
0
100
200
300
400
0
100
200
300
400
500
600

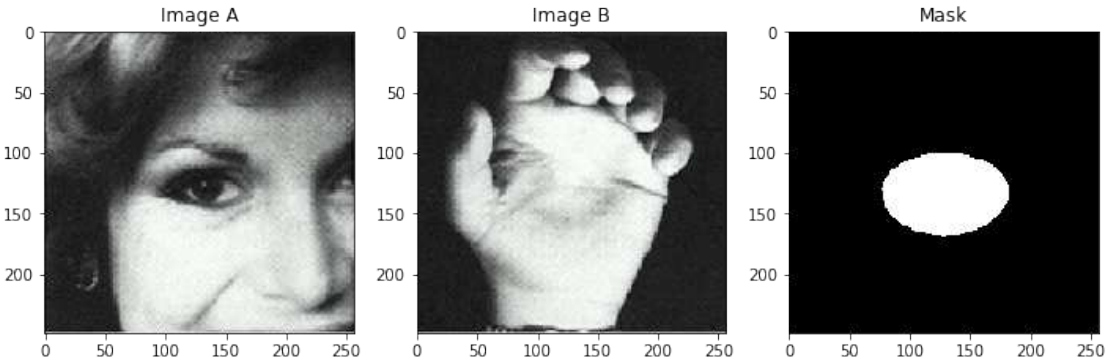


4. (5 баллов) На основе функций построения гауссовской и лапласовской пирамиды напишите функцию склейки двух изображений на основе маски. Функция должна возвращать склееные изображения и промежуточные результаты — склееные изображения разных частот (т.е. лапласовскую пирамиду совмещенного изображения). Изображения для тестирования: `a.png`, `b.png`, `mask.png`. Изображение с маской не является бинарным (т.е. имеет промежуточные градации серого), его можно бинаризовать путём сравнения всех элементов с порогом 128:

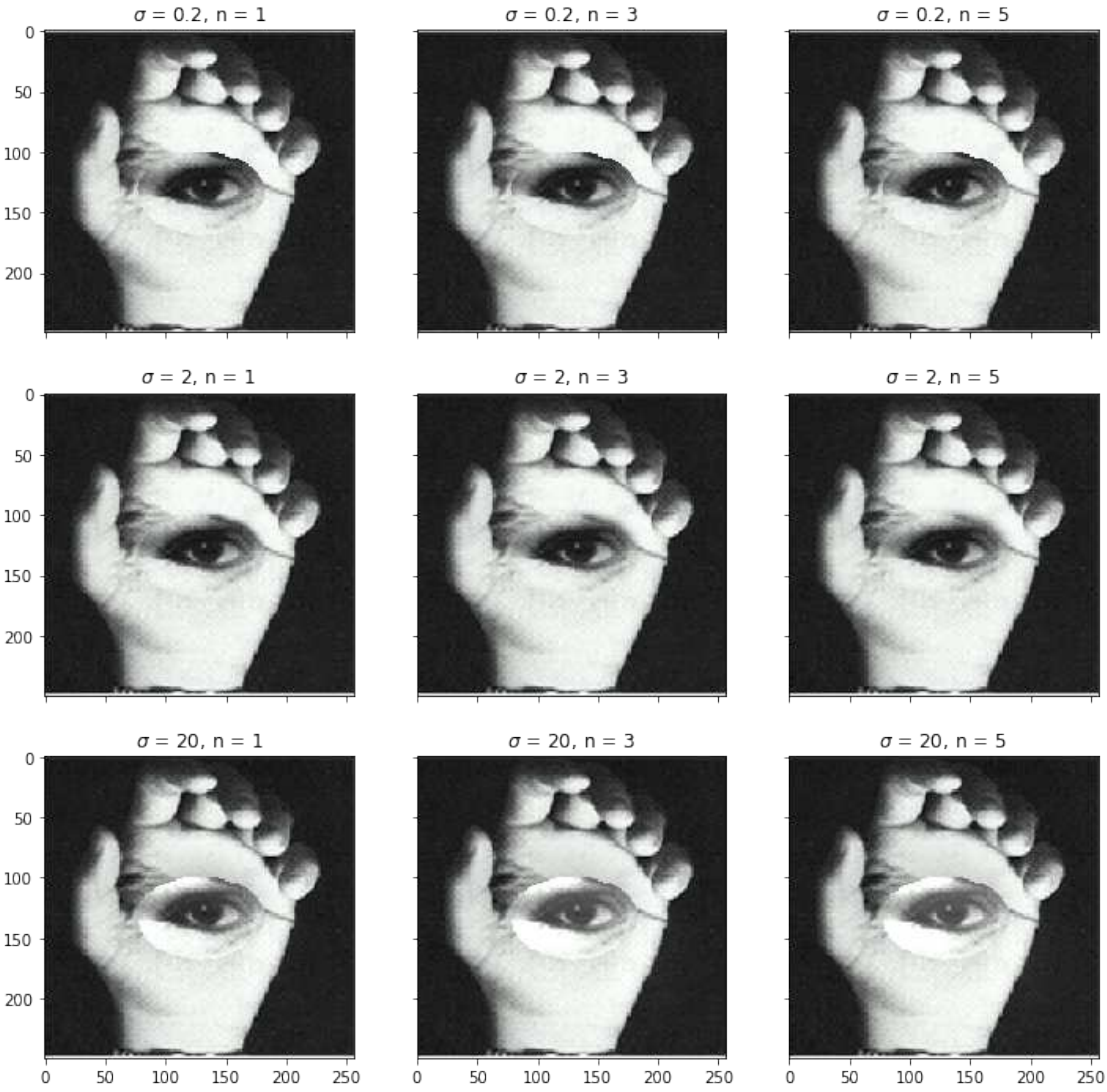
```
mask = imread('mask.png')
mask = (mask > 128).astype('uint')
```

Посмотрите, как ведет себя склейка при изменении `sigma` (попробуйте три варианта значений при фиксированном количестве слоев в пирамидах) и при изменении количества слоев (попробуйте три варианта слоев при фиксированном `sigma`). Склейка должна выдавать качественный результат без видимых артефактов хотя бы с одним набором параметров.

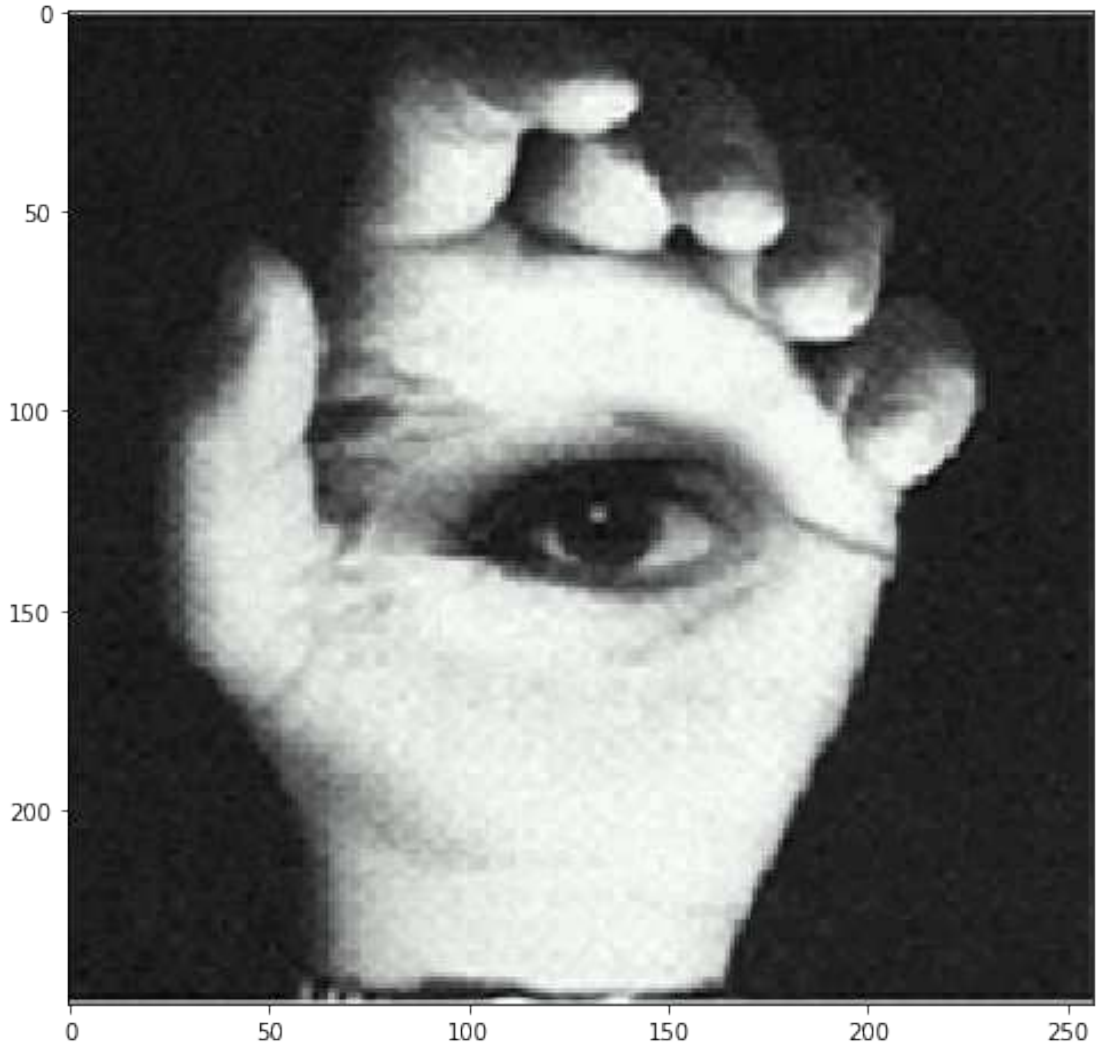
```
In [11]: a = img_as_float(imread('a.png'))
         b = img_as_float(imread('b.png'))
         mask = (imread('mask.png') > 128).astype('float')

         def show_triplet(a, b, mask, figsize=(12, 6)):
             plt.figure(figsize=figsize)
             plt.subplot(131)
             plt.imshow(a)
             plt.title('Image A')
             plt.subplot(132)
             plt.imshow(b)
             plt.title('Image B')
             plt.subplot(133)
             plt.imshow(mask)
             plt.title('Mask')
             plt.show()

         show_triplet(a, b, mask)
```

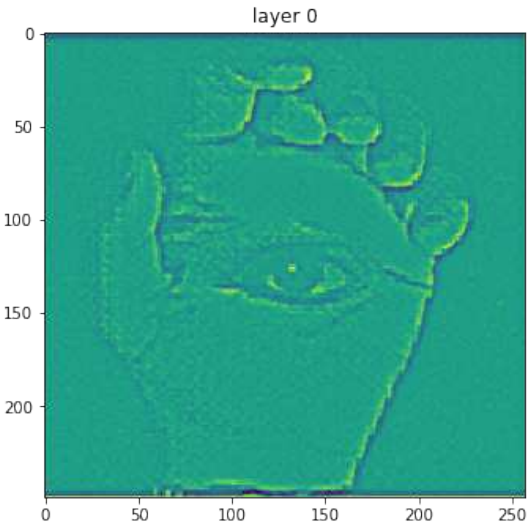


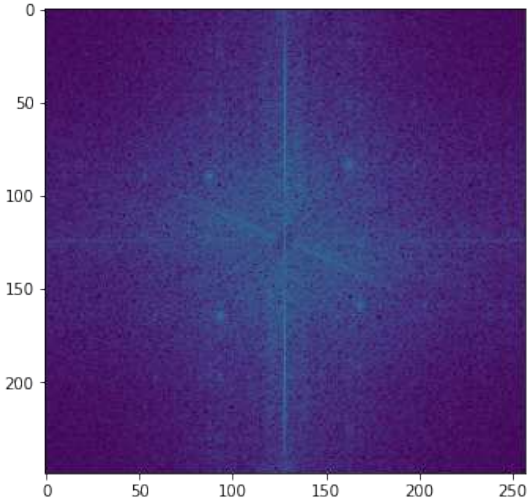

```
In [12]: def fuse_pyramids(pyr_a, pyr_b, pyr_m):
             pyr = []
             for a, b, m in zip(pyr_a, pyr_b, pyr_m):
                 pyr.append(a * m + (1 - m) * b)
             return pyr

         def pyramid_to_image(pyr):
             res = 0
             for layer in pyr[::-1]:
                 res += layer
             return np.clip(res, 0, 1)

         def blend(a, b, mask, sigma=2.0, nlayers=5):
             pyr_a = build_laplacian_pyramid(a, sigma, nlayers)
             pyr_b = build_laplacian_pyramid(b, sigma, nlayers)
             pyr_m = build_gaussian_pyramid(mask, sigma, nlayers)
             pyr = fuse_pyramids(pyr_a, pyr_b, pyr_m)
             res = pyramid_to_image(pyr)
             return res, pyr

In [13]: fig, axes = plt.subplots(3, 3, figsize=(12, 12), sharex='col', sharey='row')
         for ax_row, sigma in zip(axes, [0.2, 2, 20]):
             for ax, nlayers in zip(ax_row, [1, 3, 5]):
                 res, pyr = blend(a, b, mask, sigma, nlayers)
                 ax.imshow(res)
                 ax.set_title(f'$\sigma$ = {sigma}, n = {nlayers}')
```

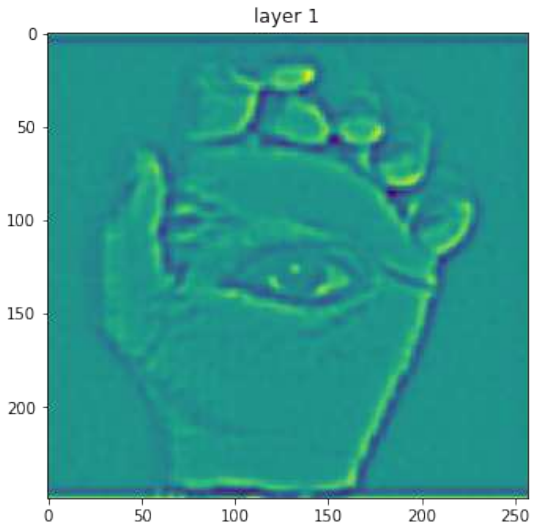


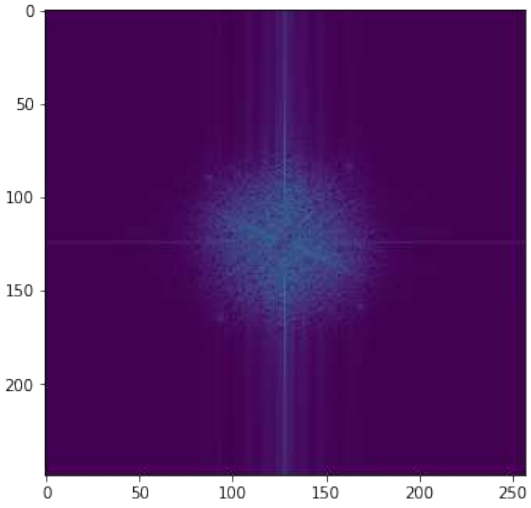
In [14]:
```
res, pyr = blend(a, b, mask, 2.0, 3)
plt.figure(figsize=(12, 8))
plt.imshow(res)
plt.show()
visualize_pyramid(pyr)
```

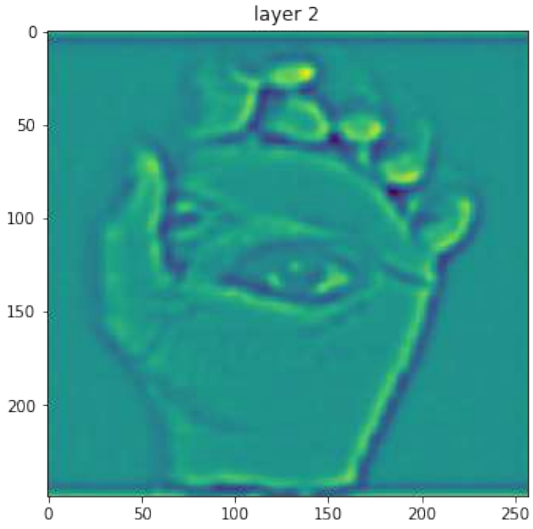

0
50
100
150
200
0
50
100
150
200
250

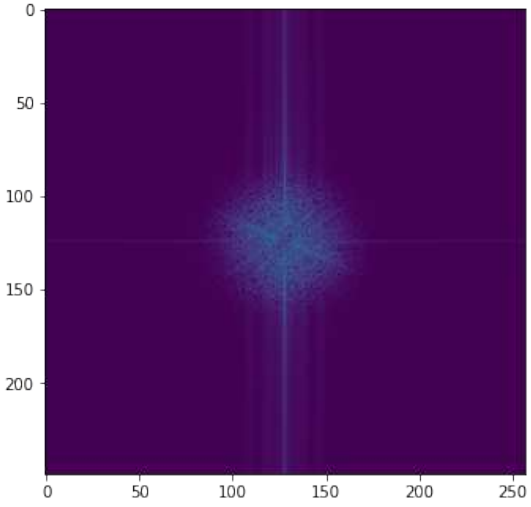

layer 0
0
50
100
150
200
0
50
100
150
200
250

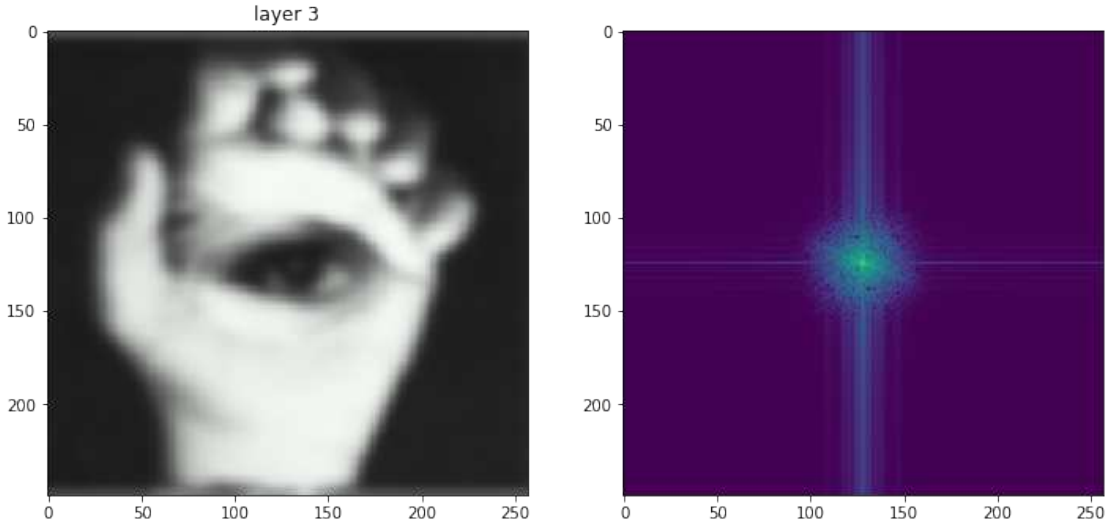

0
50
100
150
200
0
50
100
150
200
250


layer 1
0
50
100
150
200
0
50
100
150
200
250


0
50
100
150
200
0
50
100
150
200
250

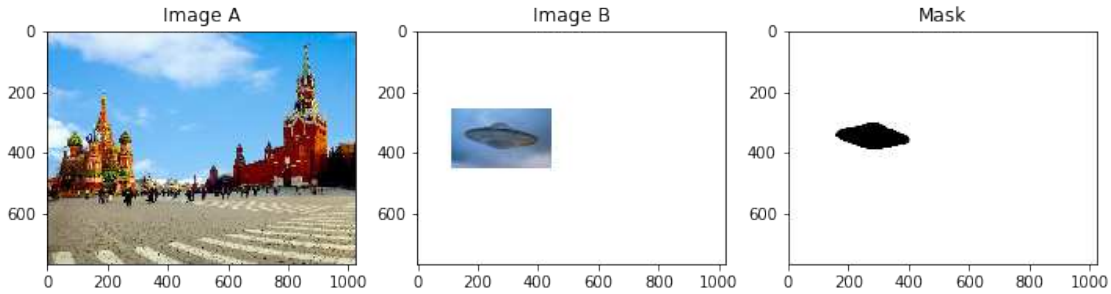

layer 2
0
50
100
150
200
0
50
100
150
200
250


0
50
100
150
200
0
50
100
150
200
250

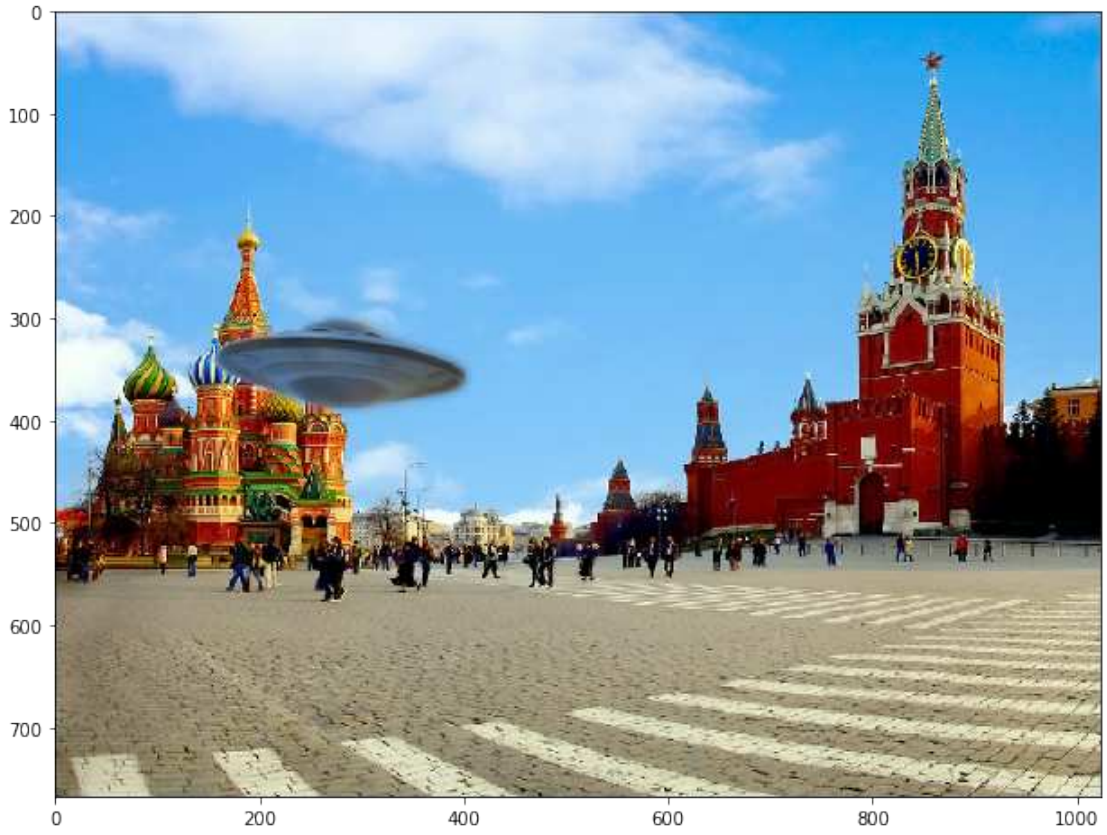


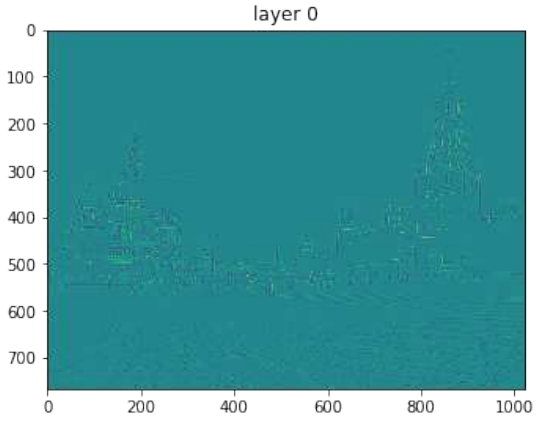
5. (5 баллов) Подготовьте самостоятельно три набора изображений и масок для склейки и визуализируйте результаты функции. У вас должно получиться 3 качественных и интересных коллажа.

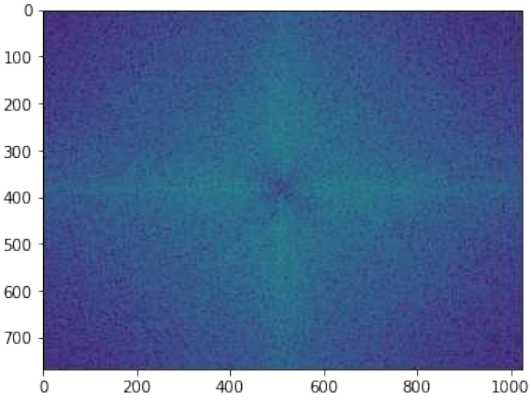
```
In [15]: a = img_as_float(imread('red-square.jpg'))
         b = img_as_float(imread('ufo.jpg'))
         mask = img_as_float(imread('ufo-mask.jpg'))
         show_triplet(a, b, mask)
```

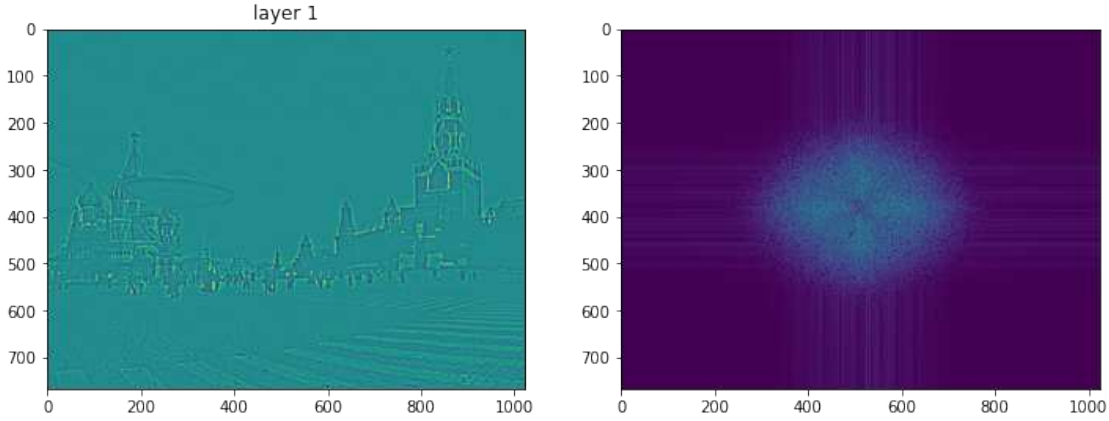


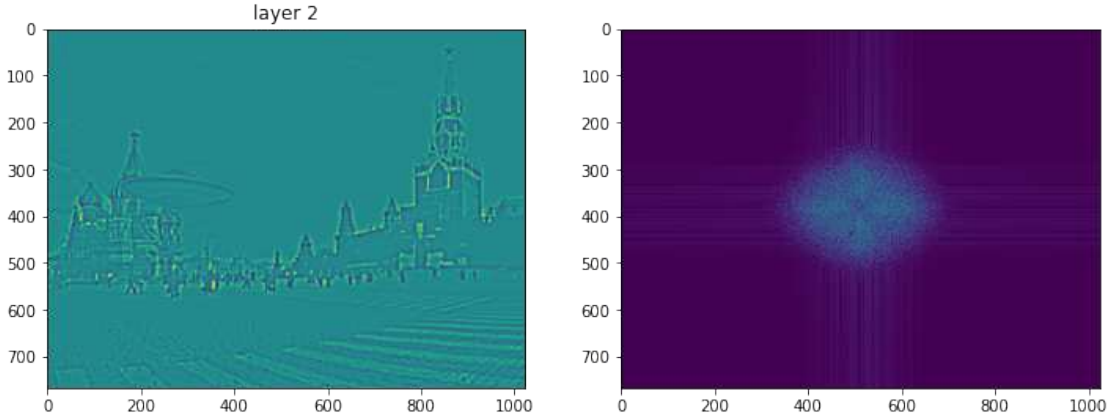

```
In [16]: res, pyr = blend(a, b, mask)
         plt.figure(figsize=(12, 8))
         plt.imshow(res)
         plt.show()
         visualize_pyramid(pyr)
```

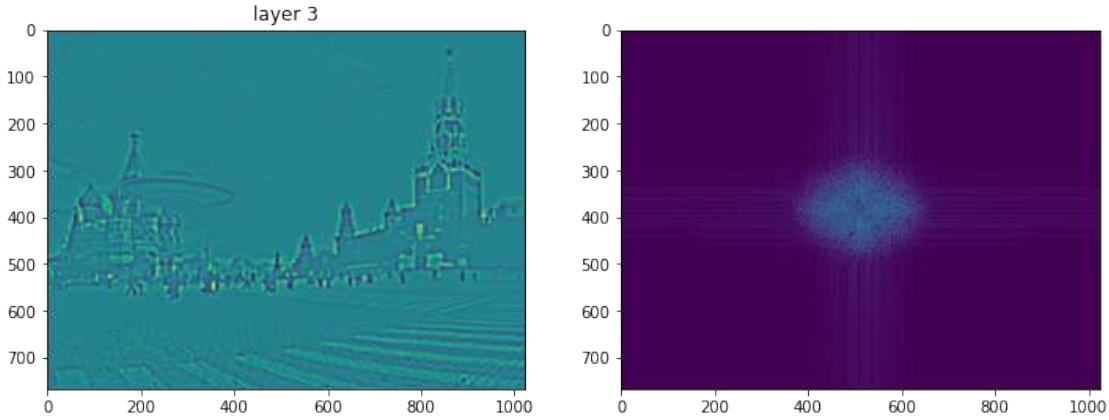

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

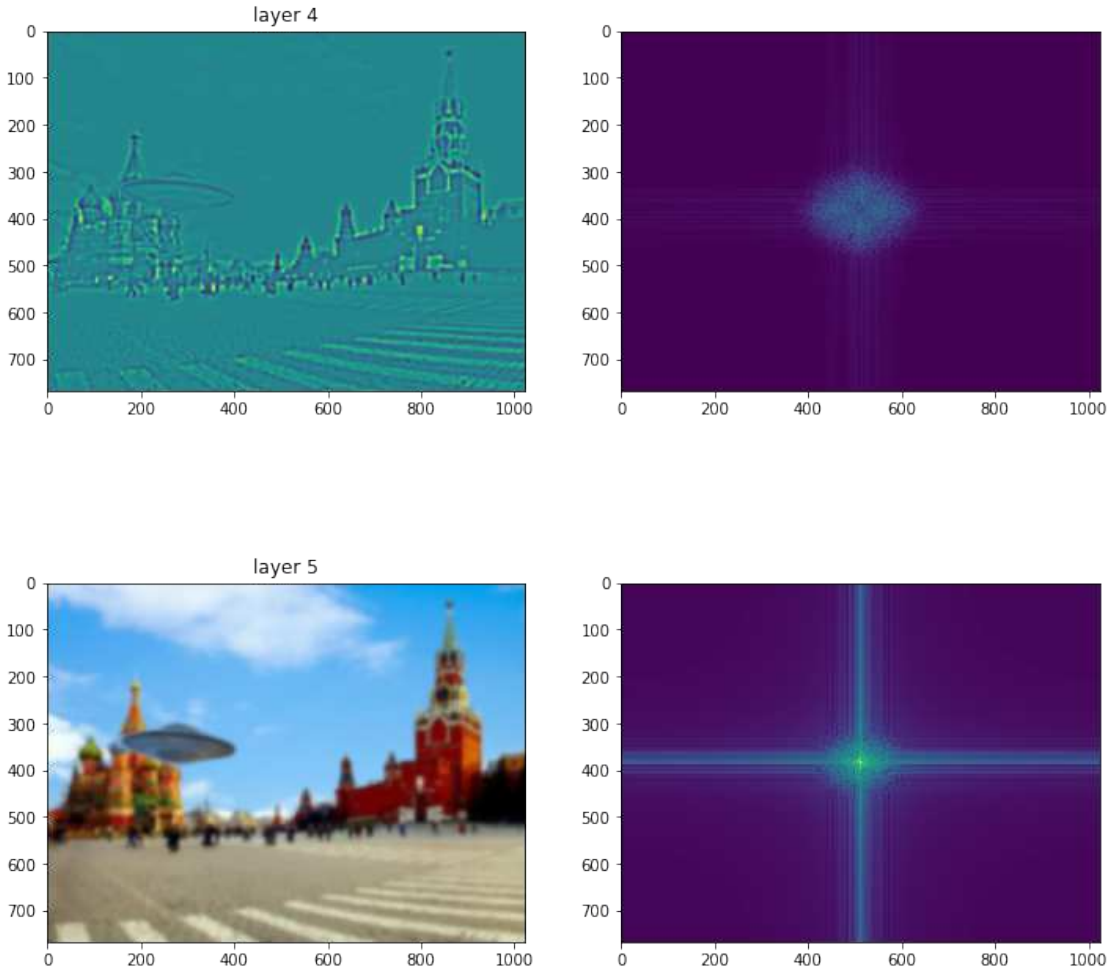

layer 0
0
100
200
300
400
500
600
700
0
200
400
600
800
1000


0
100
200
300
400
500
600
700
0
200
400
600
800
1000

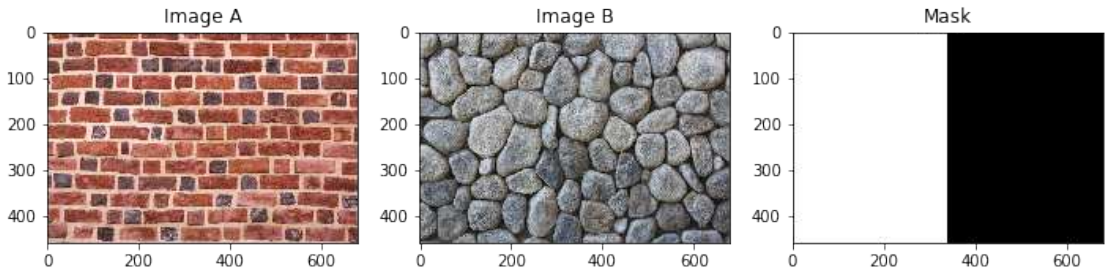

layer 1
0
100
200
300
400
500
600
700
0
200
400
600
800
1000


layer 2
0
100
200
300
400
500
600
700
0
200
400
600
800
1000

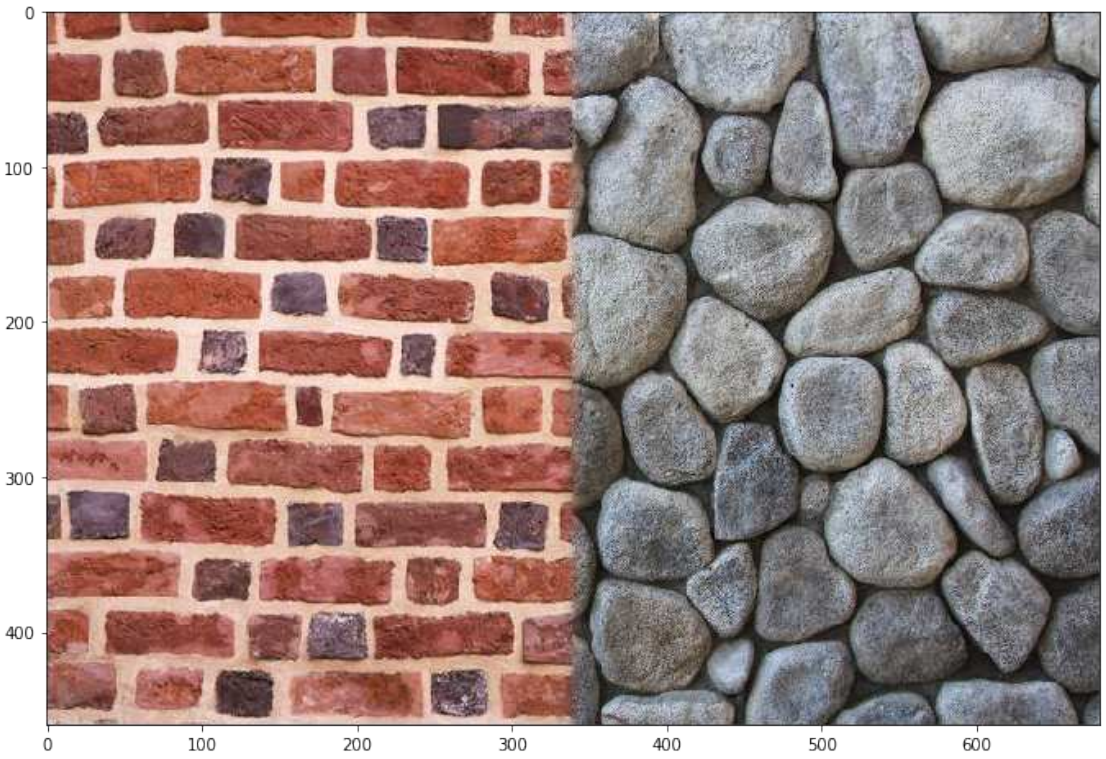

layer 3
0
100
200
300
400
500
600
700
0
200
400
600
800
1000

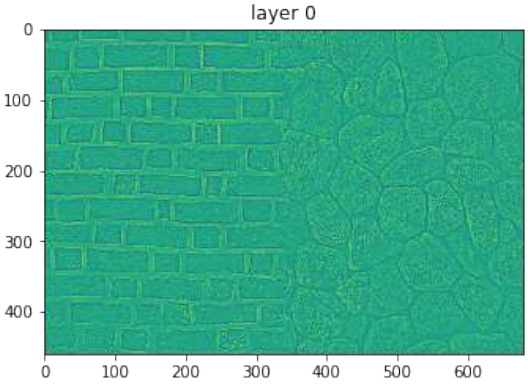


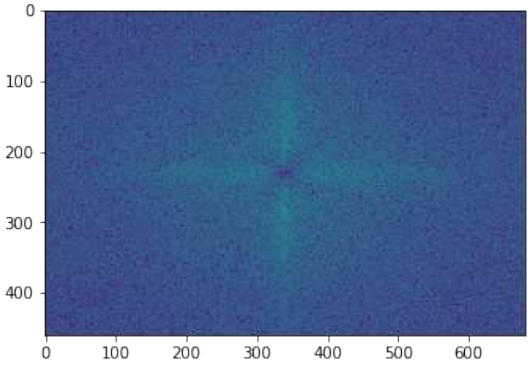

```
In [17]: a = img_as_float(imread('texture-a.jpg'))
         b = img_as_float(imread('texture-b.jpg'))
         h, w, ndims = a.shape
         mask = np.zeros((h, w, ndims), np.float32)
         mask[:, :w//2] = 1
         show_triplet(a, b, mask)
```

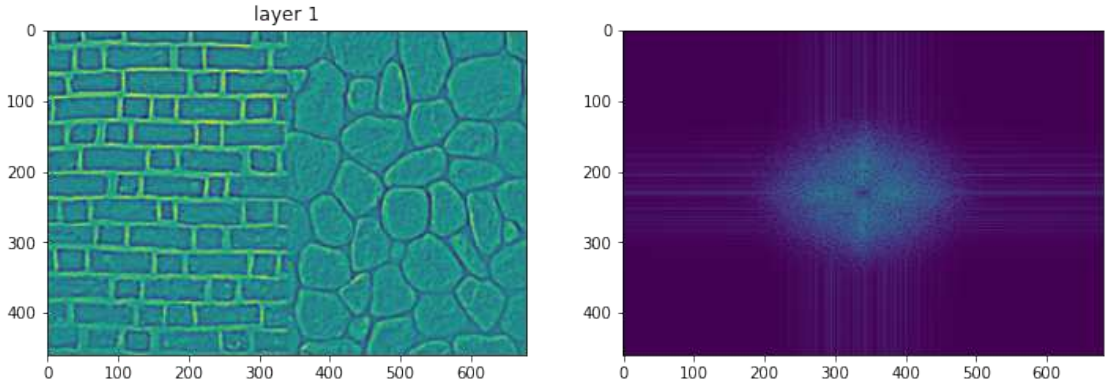


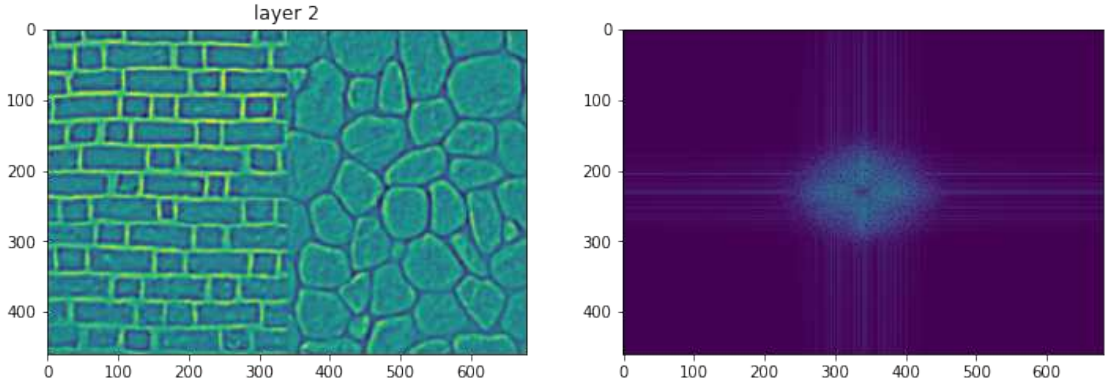
```
In [18]: res, pyr = blend(a, b, mask, sigma=2, nlayers=5)
         plt.figure(figsize=(12, 8))
         plt.imshow(res)
         plt.show()
         visualize_pyramid(pyr)
```

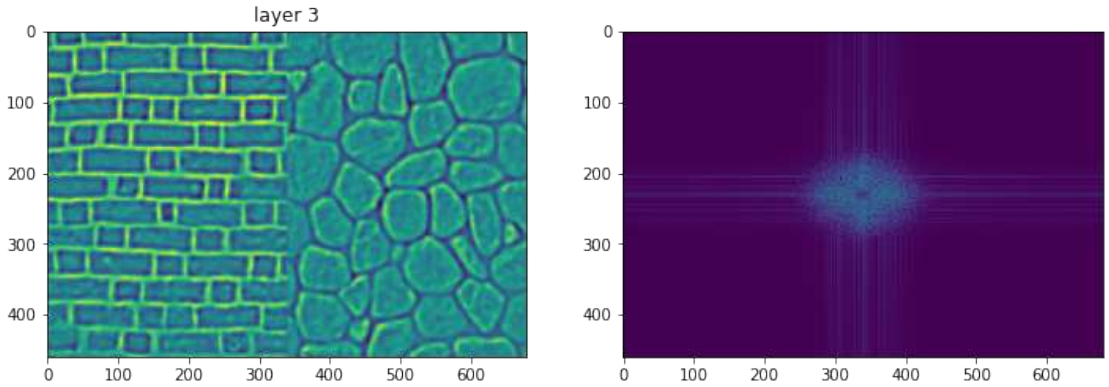


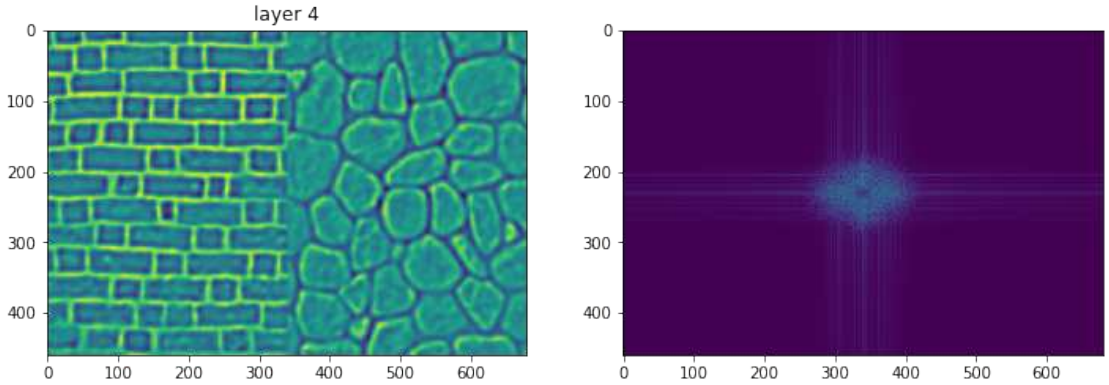


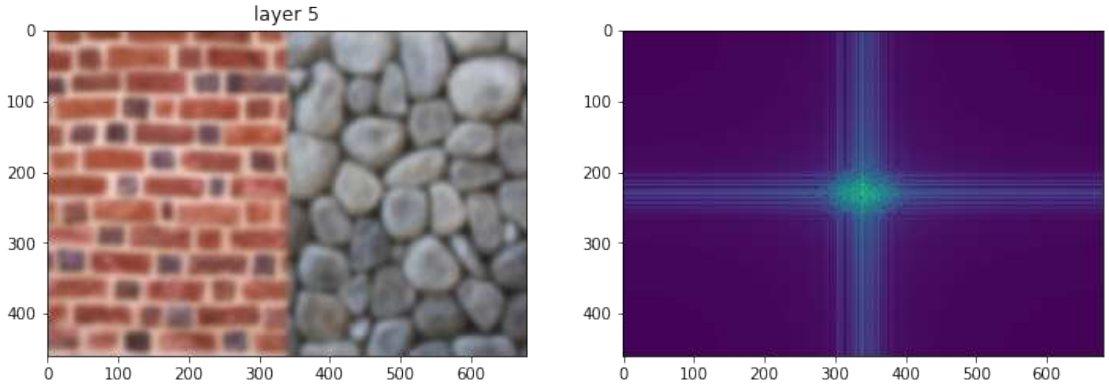



layer 1
0
100
200
300
400
0
100
200
300
400
500
600

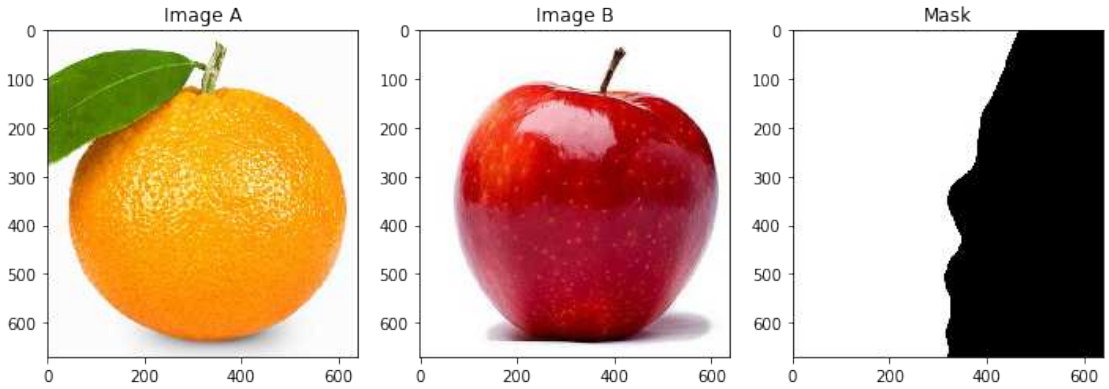

layer 2
0
100
200
300
400
0
100
200
300
400
500
600


layer 3
0
100
200
300
400
0
100
200
300
400
500
600

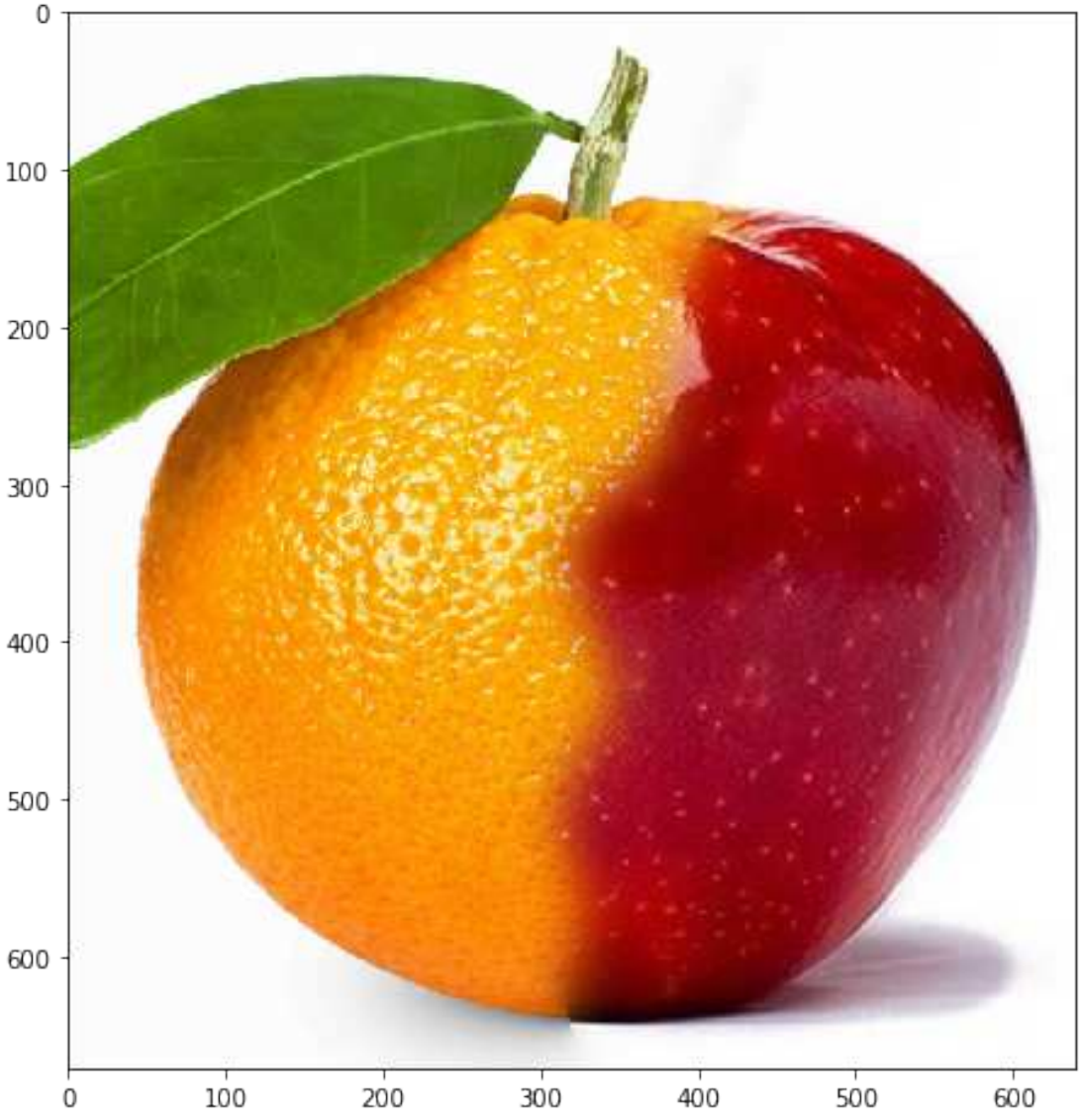


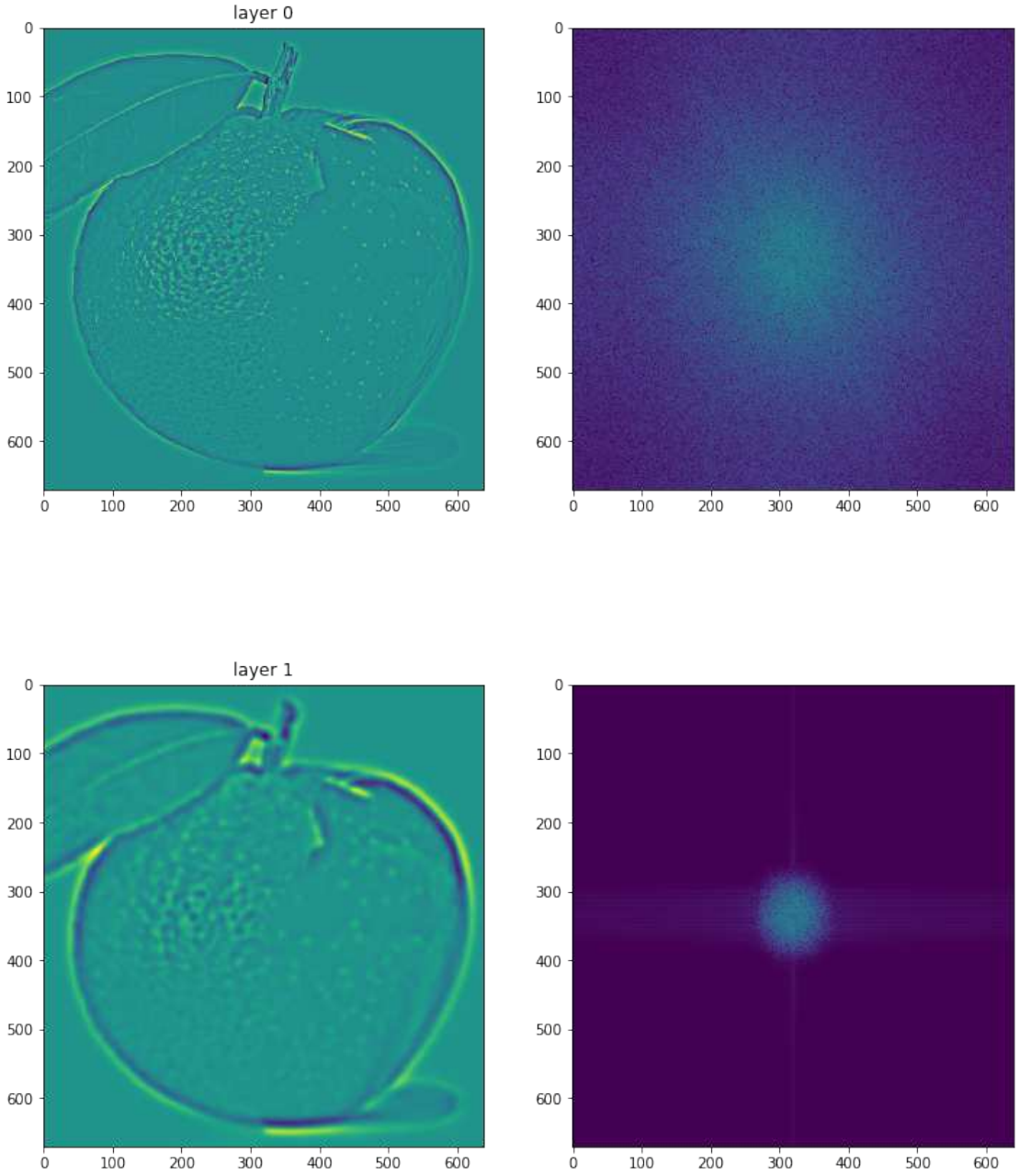


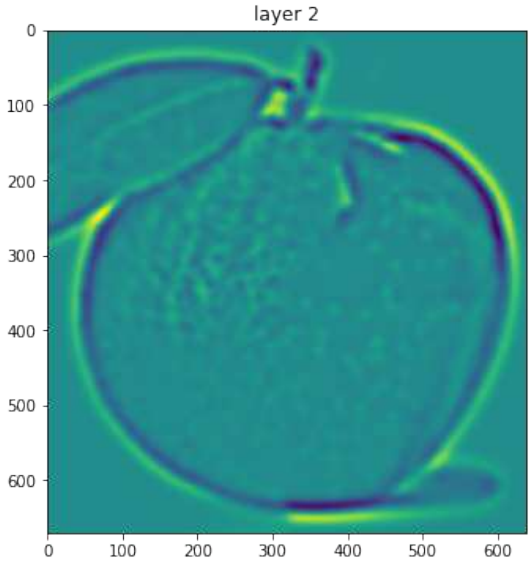

```
In [19]: a = img_as_float(imread('orange.jpg'))
         b = img_as_float(imread('apple.jpg'))
         mask = img_as_float(imread('orange-mask.jpg'))
         show_triplet(a, b, mask)
```

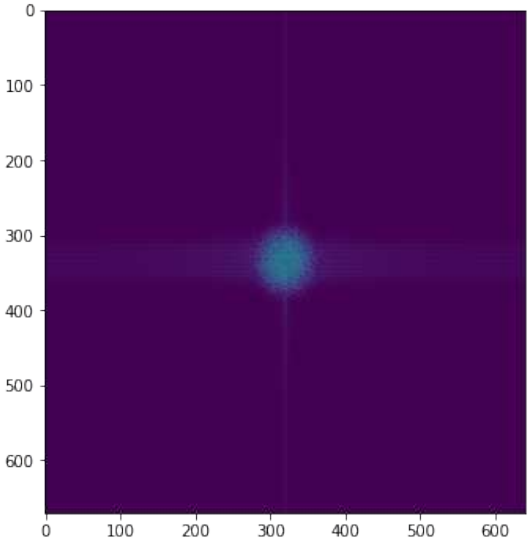


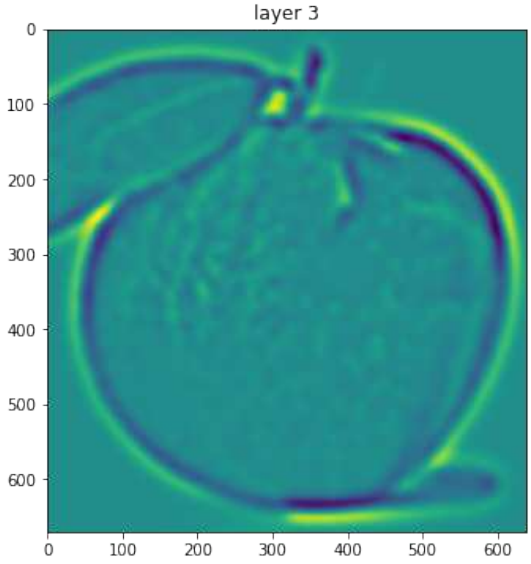
In [20]:
```
res, pyr = blend(a, b, mask, sigma=5, nlayers=10)
plt.figure(figsize=(12, 8))
plt.imshow(res)
plt.show()
visualize_pyramid(pyr)
```

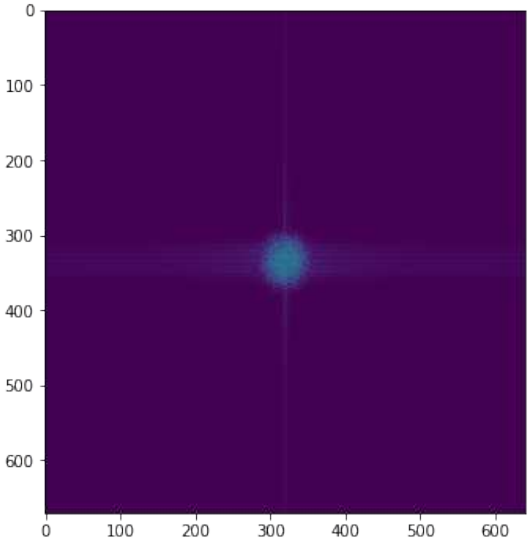


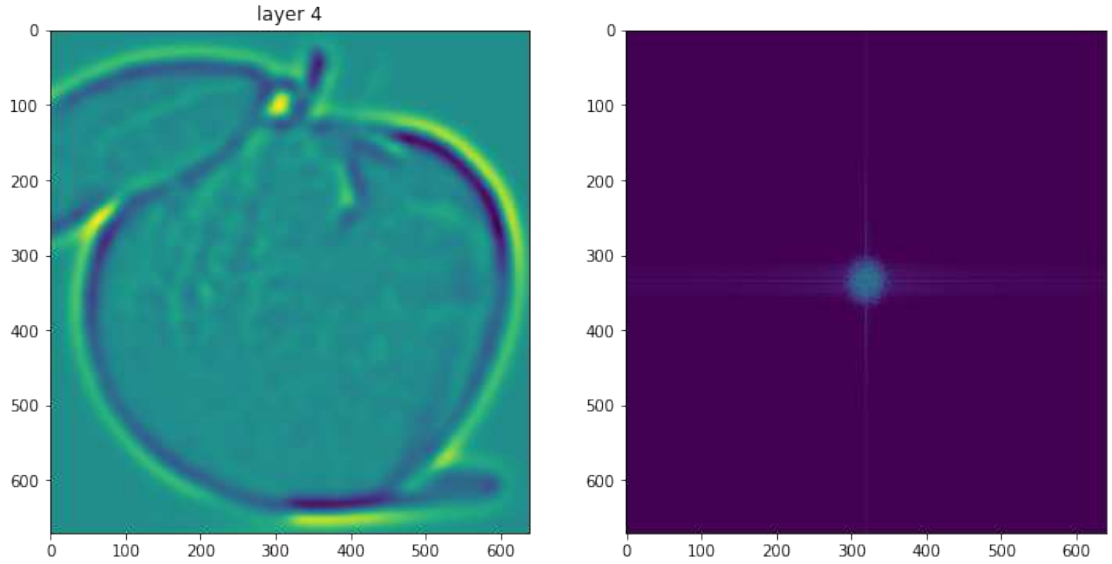

layer 0
layer 1

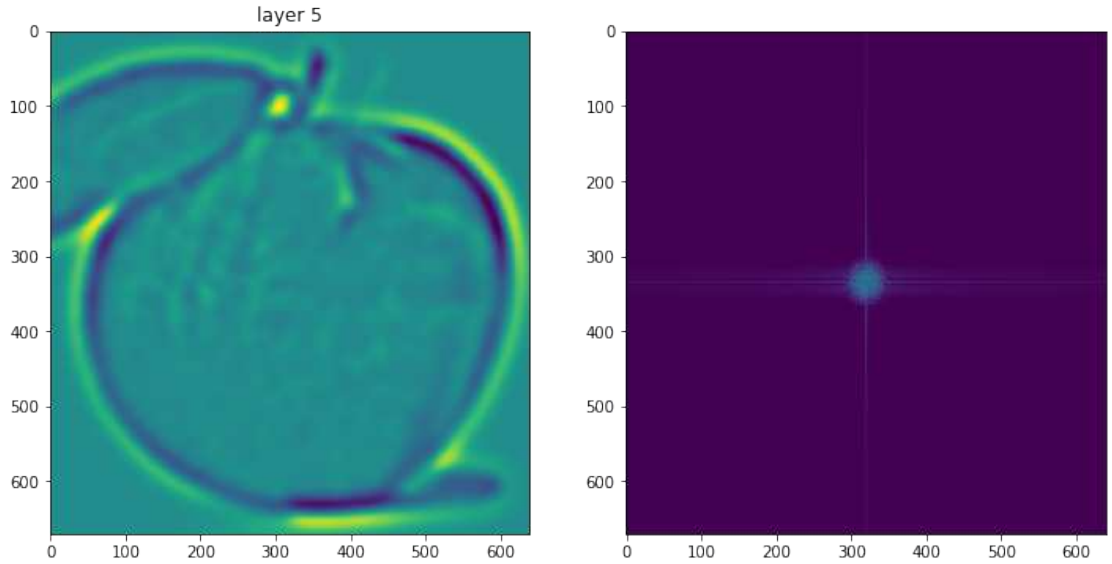

layer 2
0
100
200
300
400
500
600

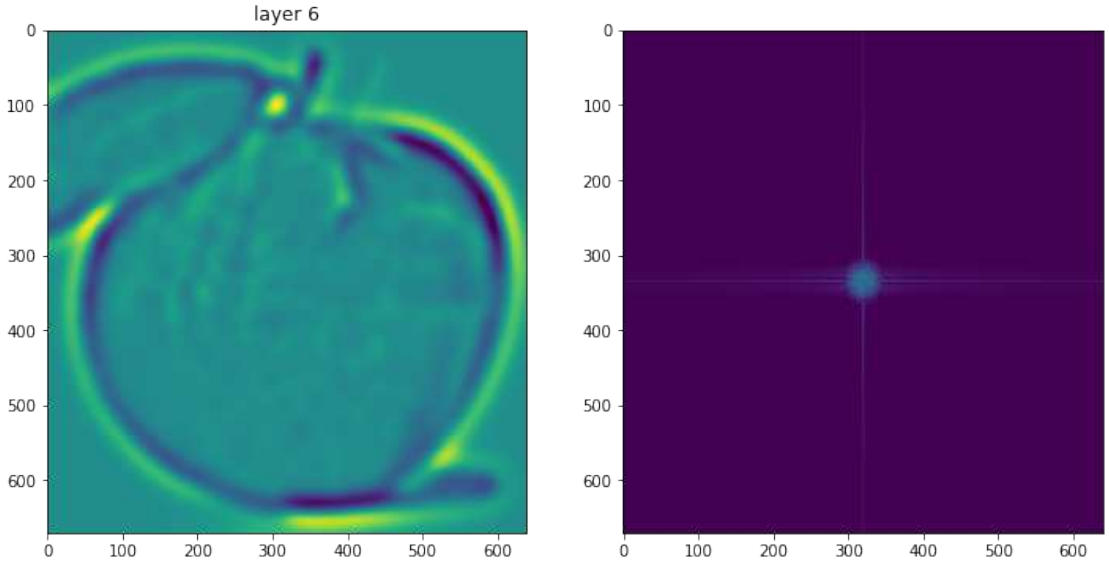

0
100
200
300
400
500
600

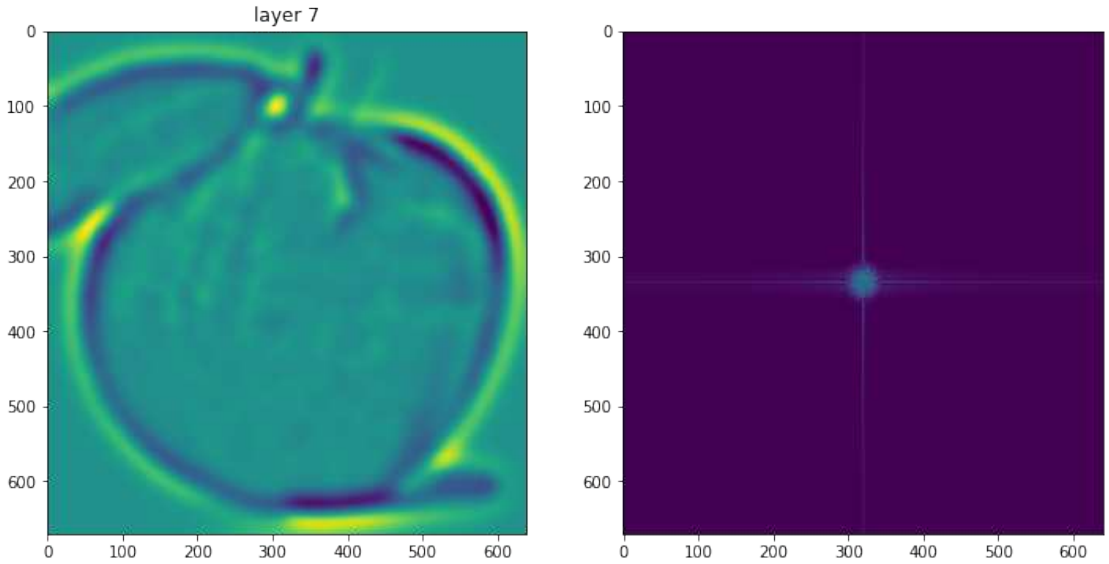

layer 3
0
100
200
300
400
500
600

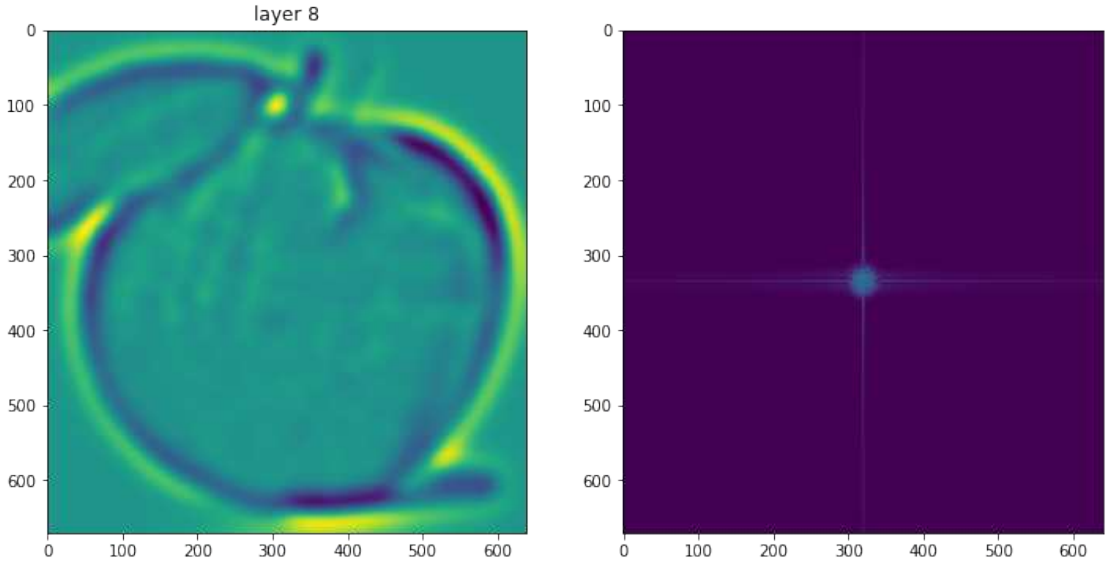

0
100
200
300
400
500
600

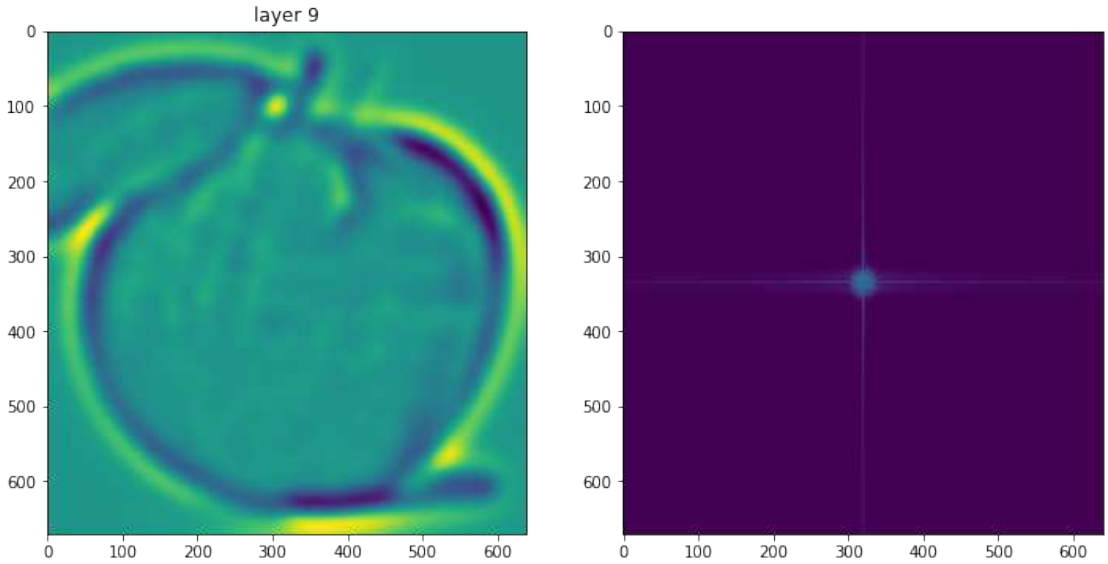

layer 4
0
100
200
300
400
500
600

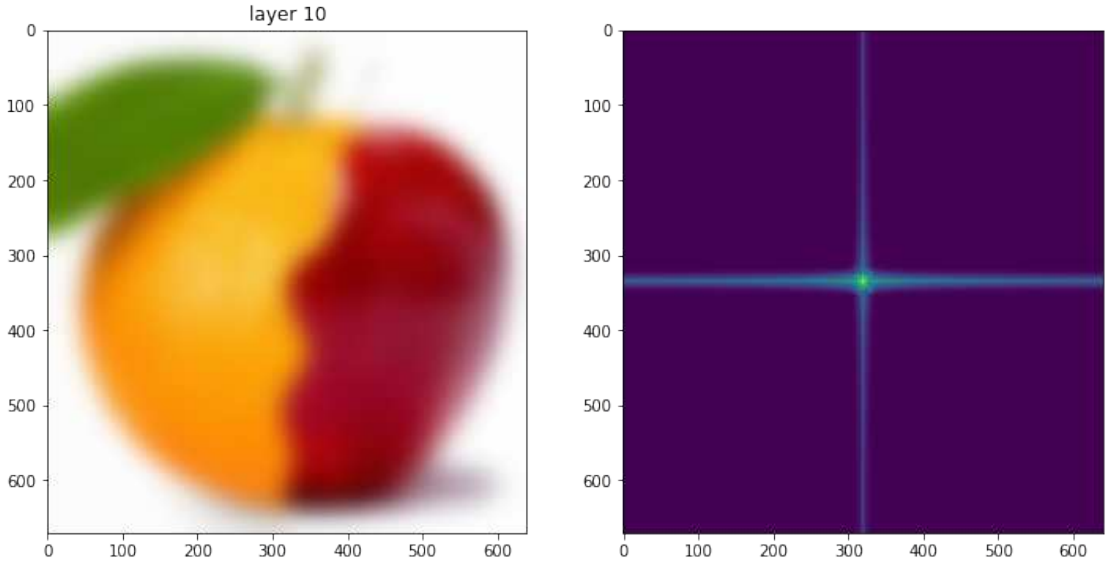

layer 5
0
100
200
300
400
500
600


layer 6
0
100
200
300
400
500
600


layer 7
0
100
200
300
400
500
600


layer 8
0
100
200
300
400
500
600


layer 9
0
100
200
300
400
500
600


layer 10
0
100
200
300
400
500
600