CPSC 425 Assignment 1 （due Monday September 24, 2018）

Quan Zhang
48486154

Part1 Gaussian Filtering
1 boxfilter(n), in which n should be positive odd numbers.
Input: boxfilter(3)
Output: array([[0.11111111, 0.11111111, 0.11111111],
[0.11111111, 0.11111111, 0.11111111],
[0.11111111, 0.11111111, 0.11111111]])

Input: boxfilter(4)
Output: AssertionError(must be odd)

Input: boxfilter(5)
Output: array([[0.04, 0.04, 0.04, 0.04, 0.04],
[0.04, 0.04, 0.04, 0.04, 0.04],
[0.04, 0.04, 0.04, 0.04, 0.04],
[0.04, 0.04, 0.04, 0.04, 0.04],
[0.04, 0.04, 0.04, 0.04, 0.04]])

(implemented in file a1.py as boxfilter(x))

2 gauss1d(sigma), returns a 1D Gaussian filter for a given value of sigma.
Input: gauss1d(0.3)
Output: array([0.00383626, 0.99232748, 0.00383626])

Input: gauss1d(0.5)
Output: array([0.10650698, 0.78698604, 0.10650698])

Input: gauss1d(1)
Output: array([0.00443305, 0.05400558, 0.24203623, 0.39905028, 0.24203623,
0.05400558, 0.00443305])

Input: gauss1d(2)
Output: array([0.0022182 , 0.00877313, 0.02702316, 0.06482519, 0.12110939,
0.17621312, 0.19967563, 0.17621312, 0.12110939, 0.06482519,
0.02702316, 0.00877313, 0.0022182])

(implemented in file a1.py as gauss1d(sigma))

3 gauss2d(sigma), returns a 2D Gaussian filter for a given value of sigma.
Input: gauss2d(0.5)
Output: array([[0.01134374, 0.08381951, 0.01134374],
[0.08381951, 0.61934703, 0.08381951],
[0.01134374, 0.08381951, 0.01134374]])

Input: gauss2d(1)

```
Output: array([[ 1.96519161e-05,  2.39409349e-04,  1.07295826e-03,
         1.76900911e-03,  1.07295826e-03,  2.39409349e-04,
         1.96519161e-05],
       [ 2.39409349e-04,  2.91660295e-03,  1.30713076e-02,
         2.15509428e-02,  1.30713076e-02,  2.91660295e-03,
         2.39409349e-04],
       [ 1.07295826e-03,  1.30713076e-02,  5.85815363e-02,
         9.65846250e-02,  5.85815363e-02,  1.30713076e-02,
         1.07295826e-03],
       [ 1.76900911e-03,  2.15509428e-02,  9.65846250e-02,
         1.59241126e-01,  9.65846250e-02,  2.15509428e-02,
         1.76900911e-03],
       [ 1.07295826e-03,  1.30713076e-02,  5.85815363e-02,
         9.65846250e-02,  5.85815363e-02,  1.30713076e-02,
         1.07295826e-03],
       [ 2.39409349e-04,  2.91660295e-03,  1.30713076e-02,
         2.15509428e-02,  1.30713076e-02,  2.91660295e-03,
         2.39409349e-04],
       [ 1.96519161e-05,  2.39409349e-04,  1.07295826e-03,
         1.76900911e-03,  1.07295826e-03,  2.39409349e-04,
         1.96519161e-05]])
```

(implemented in file a1.py as gauss2d(sigma))

4

(a) (implemented in file a1.py as gaussConvolve2d(image_array, sigma))

Why does Scipy have separate functions for ‘signal.convolve2d’ and ‘signal.correlate2d’?

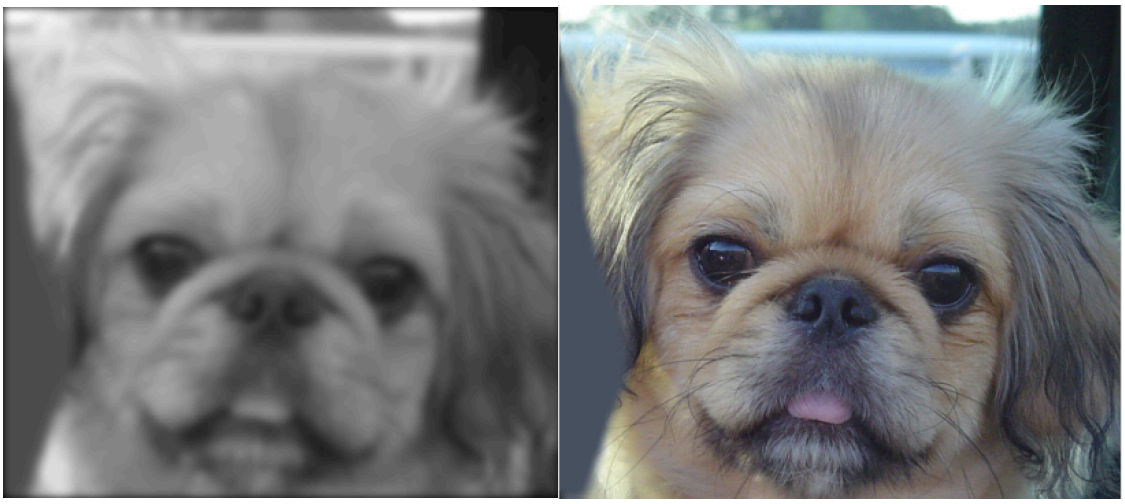
Because convolution is like correlation except filter ‘flipped’ and in situation like F(X, Y) = F(-X, -Y) then convolution = correlation.

(b) (implemented in file a1.py as gaussConvolve2d(image_array, sigma))

imgBW = imgDog1.convert('L') is the greyscale of dog.jpg

so run with gaussConvolve2d(imgBW, 3)

(c) (implemented in file a1.py as gaussConvolve2d(image_array, sigma))



figures(c) filtered image original image

5 We can use the separability of 2D Gaussian equation and express the function as a product of two functions about x and y.

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

$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)$$

function of x function of y

At each pixel originally, we do m*m multiplications. By doing this, at each pixel, we do 2m m multiplications. So the total time consumption is reduced from (m^2)*(n^2) multiplications to (2m)*(n^2) multiplications.

Part2Hybrid Images

1 gaussConvolveColor(imgDog, 9)

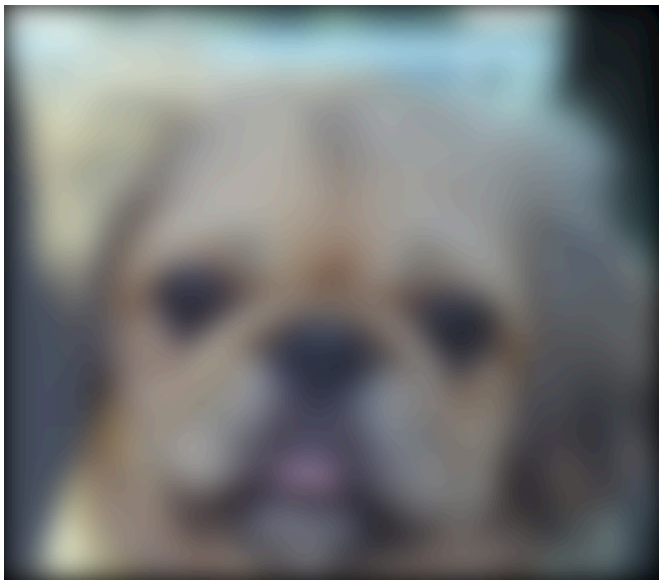


figure1 Gaussian filtered low frequencies image from ‘dog.jpg’

(implemented in file a1.py as gaussConvolveColor(img, sigma))

2 gaussHighFre(imgCat, 5)

figure2 Gaussian filtered high frequencies image from ‘0a_cat.bmp'

Normalized by adding 128 to original arrays of RGB. Same effect as normalize to 0 to 1 scale and add 0.5 to original arrays of RGB.
(implemented in file a1.py as gaussHighFre(img, sigma))

3 As it could be High frequency filtered image of image1 with Low frequency filtered image of image2, or High frequency filtered image of image2 with Low frequency filtered image of image1, both pictures are provided.
gaussHybrid(imgDog, imgCat, 3)

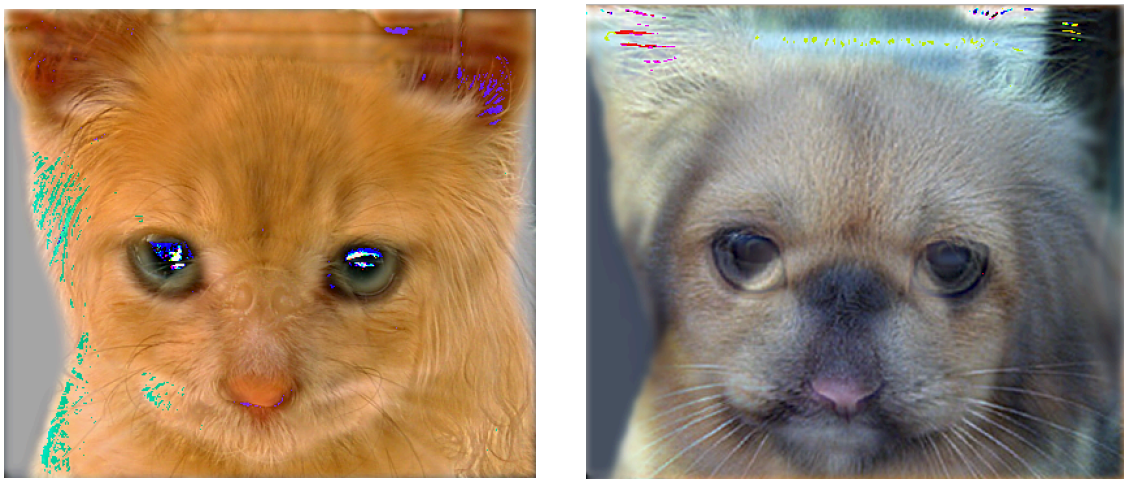


figure3 Hybrid of Cat and Dog with sigma = 3

gaussHybrid(imgDog, imgCat, 5)

figure3 Hybrid of Cat and Dog with sigma = 5

gaussHybrid(imgDog, imgCat, 10)

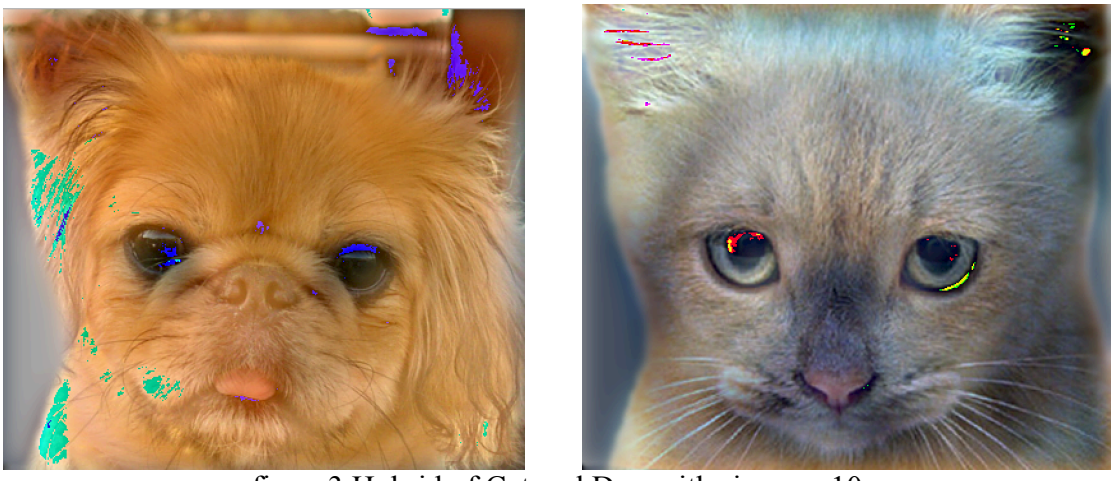


figure3 Hybrid of Cat and Dog with sigma = 10

gaussHybrid(img2a, img2b, 4)

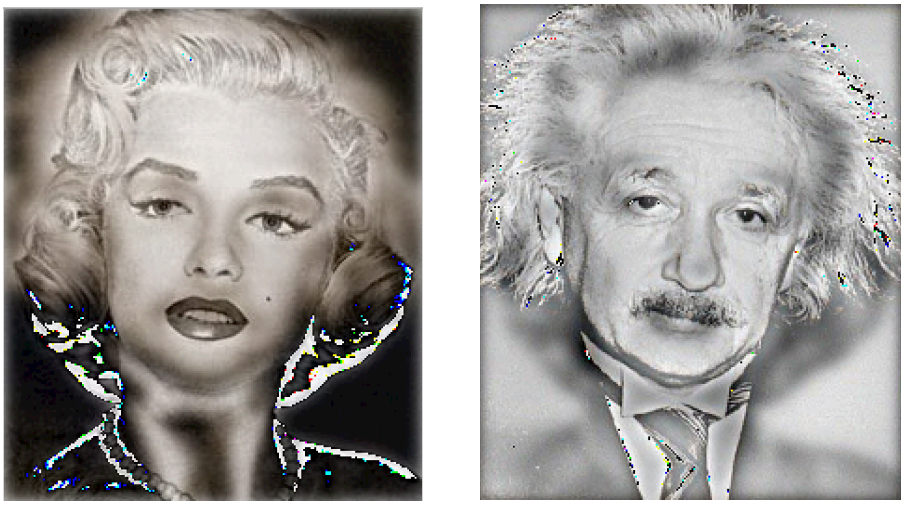


figure3 Hybrid of set 2 with sigma = 4

gaussHybrid(img3a, img3b, 6)

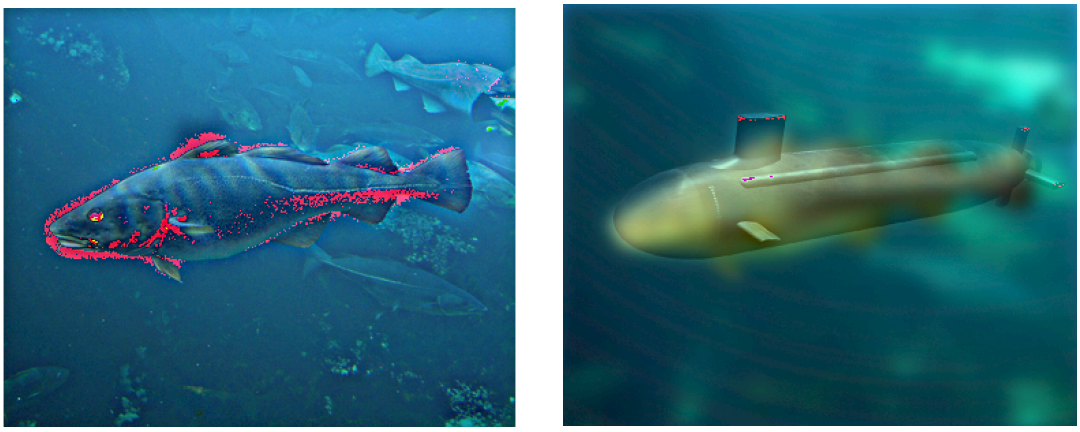


figure3 Hybrid of set 3 with sigma = 6

gaussHybrid(img1a, img1b, 8)

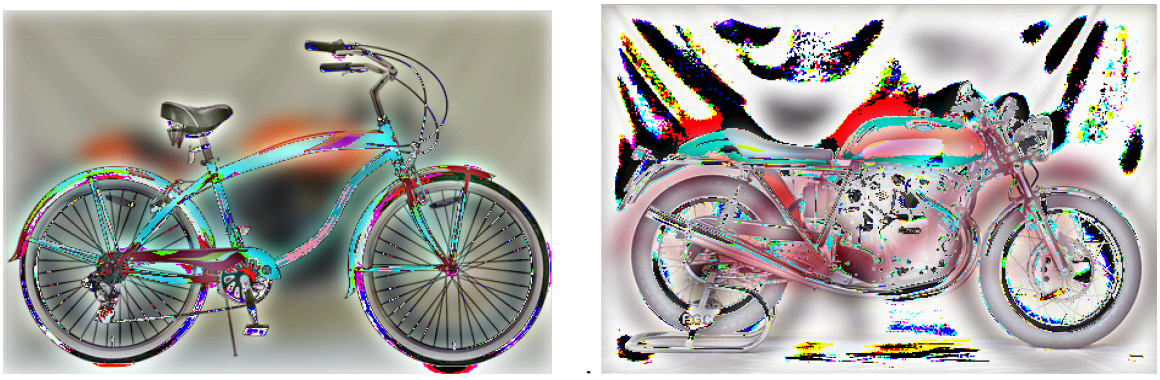


figure3 Hybrid of set 1 with sigma = 8