

## CSC420 Assignment 1

Guanxiong Liu

liuguanx

1002077726

1.

```
import cv2 as cv
import numpy as np
from scipy.ndimage import filters
from matplotlib import pyplot as plt
from scipy import signal

#Question 1
def imfilter(I, f, mode):
    img = cv.imread(I)
    if img is None:
        raise Exception("Error opening image")
    f = np.array(f)
    filter_size = f.shape
    # if img.shape[2] == 3:
    #     img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    if mode == 'valid':
        if img.shape[0] < filter_size[0] or img.shape[1] < filter_size[1]:
            raise Exception("image must be at least as large as filter in any dimension")
        if img.shape[2] == 3:
            result = np.zeros((img.shape[0]+filter_size[0]-1, img.shape[1]+filter_size[1]-1, 3))
        else:
            result = np.zeros((img.shape[0]-filter_size[0]+1, img.shape[1]-filter_size[1]+1))
    elif mode == 'same':
        if img.shape[2] == 3:
            result = np.zeros((img.shape[0], img.shape[1], 3))
            img = np.pad(img, ((filter_size[0]//2, filter_size[0]//2), (filter_size[1]//2, filter_size[1]//2), (0, 0)), 'constant')
        else:
            result = np.zeros((img.shape[0], img.shape[1]))
            img = np.pad(img, ((filter_size[0]//2, filter_size[0]//2), (filter_size[1]//2, filter_size[1]//2)), 'constant')
    elif mode == 'full':
        if img.shape[2] == 3:
            result = np.zeros((img.shape[0]+filter_size[0]-1, img.shape[1]+filter_size[1]-1, 3))
            img = np.pad(img, ((filter_size[0]-1, filter_size[0]-1), (filter_size[1]-1, filter_size[1]-1), (0, 0)), 'constant')
        else:
            result = np.zeros((img.shape[0]+filter_size[0]-1, img.shape[1]+filter_size[1]-1))
            img = np.pad(img, ((filter_size[0]-1, filter_size[0]-1), (filter_size[1]-1, filter_size[1]-1)), 'constant')
    for x in range(result.shape[0]):
        for y in range(result.shape[1]):
            result[x, y] = correlate(img, f, x+filter_size[0]//2, y+filter_size[1]//2)
    return result

def correlate(img, filter, x, y):
    filter_size = filter.shape
    flat_img = img[x-filter_size[0]//2: x+filter_size[0]//2 + 1, y-filter_size[1]//2: y+filter_size[1]//2 + 1].reshape(filter_size[0]*filter_size[1], -1)
    flat_filter = filter.flatten()
    return np.sum(flat_img * flat_filter[:, None], axis=0)
```

2. I will just flip the filter on both axis using filter[::-1,::-1].

```
def gaussian_filter(img, sigma_x, sigma_y):
    a = np.zeros((3*sigma_x, 3*sigma_y))
    a[3*sigma_x//2, 3*sigma_y//2] = 1
    gaussian_f = filters.gaussian_filter(a, (sigma_x, sigma_y))
    return imfilter(img, gaussian_f[::-1,::-1], 'same')
```

```
>>> new = gaussian_filter('iris.jpg', 3, 5)
```

```
>>> cv.imwrite('new_iris.jpg', new)
```



3. Convolution is a commutative operation.

$$\begin{aligned}
 f * g(i, j) &= \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(m, n) g(i - m, j - n) \quad \# \text{let } m = i - x, n = j - y \\
 &= \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} g(x, y) f(i - x, j - y) \\
 &= g * f(i, j)
 \end{aligned}$$

Correlation is not a commutative operation.

$$\begin{aligned}
 f \otimes g(i, j) &= \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(m, n) g(i + m, j + n) \quad \# \text{let } m = x - i, n = y - j \\
 &= \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} g(x, y) f(x - i, y - j) \\
 &\neq g \otimes f(i, j)
 \end{aligned}$$

4. It is a separable filter.

$$\begin{aligned}
 \frac{\partial G(x, y)}{\partial x} &= \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}} * \left(-\frac{2x}{\sigma^2}\right) \\
 &= -\left(\frac{x}{\sqrt{\pi}\sigma^2} e^{-\frac{x^2}{\sigma^2}}\right) \left(\frac{1}{\sqrt{\pi}\sigma^2} e^{-\frac{y^2}{\sigma^2}}\right)
 \end{aligned}$$

- 5.

	Separable	Not Separable
Convolution	$2mn^2$	$m^2 n^2$
Correlation	$m^2 n^2$	$m^2 n^2$

# CSC420 Assignment 1

Guanxiong Liu

liuguanx

1002077726

6.

$$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

All three filters are separable.

$$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} [0 \quad 1 \quad 1]$$

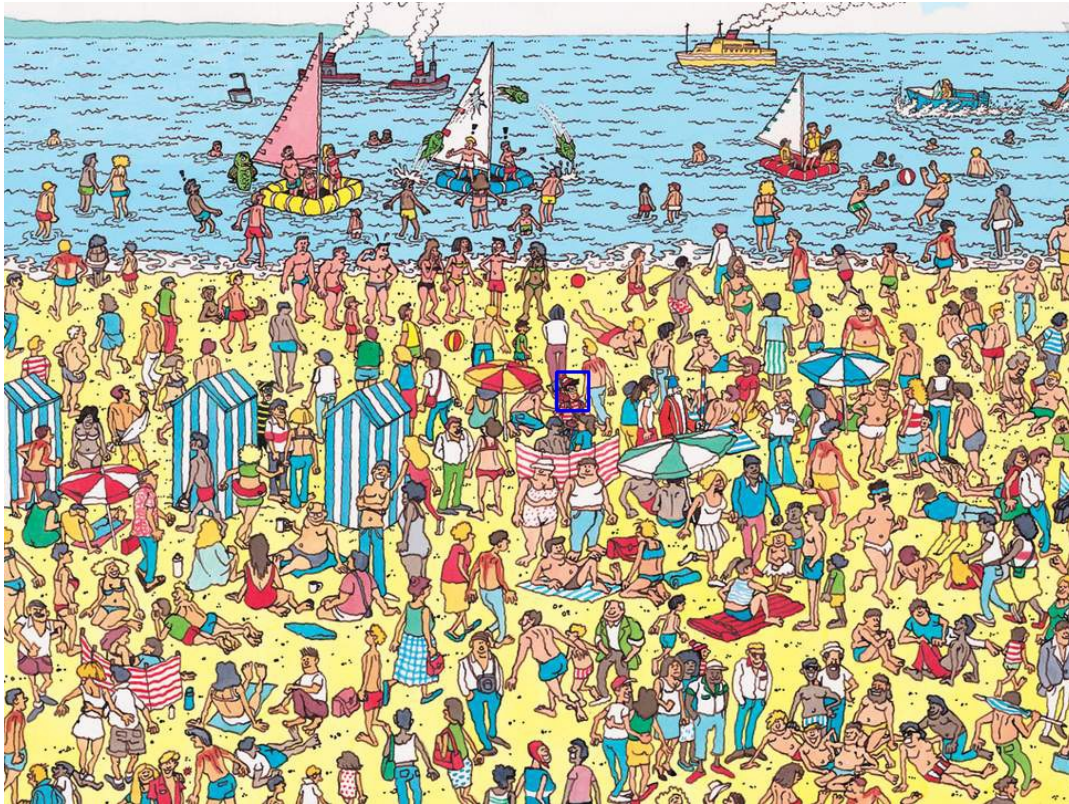
$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} [1 \quad 0 \quad 0]$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} [1 \quad 1 \quad 1]$$

7. Derivative of Gaussian ( $\sigma = 3$ )      Laplacian of Gaussian(cv) ( $\sigma = 3$ )      Laplacian of Gaussian(scipy) ( $\sigma = 3$ )



8. Code is in assignment1.py



9. Canny Edge works as the following:





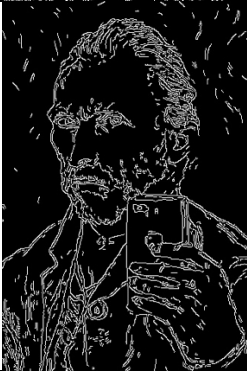




- Noise reduction---since edge detection is susceptible to noise in the image, so we need to smooth the image.
- Filter image with derivative of Gaussian (both horizontal and vertical directions)-- -locating edges
- Find magnitude and orientation of gradient---this step is for later use. It is used for Non-maximum suppression (thinning the edges).
- Non-maximum suppression---check if pixel is local maximum along gradient direction, if yes take it, if not erase it. Thus, thinning the edges.
- Linking and thresholding (hysteresis)---Define two thresholds: low and high. Use the high threshold to start edge curves and the low threshold to continue them. (i.e. find pixels that are higher than high threshold and check adjacent pixels in the direction of the edge. If any of them is higher than the low threshold, take them. Thus, link all the edges).

10. Because edges in images give rise to zero crossings in the Laplacian of Gaussian output. As we know, the second derivative is zero when the graphs are at points of changing concavity, that is, where the steepest slope occur.

11. With threshold 100 and 500. The result is the best. It discards all the background details while preserving details on the portrait. For more testing results, see below.





	200	300	400	500
100				
200				
300				
400				