Computer Vision Homework 10

Name: 黃新予

Student ID: f08922136

Three functions are implemented.

1. def paddingArray(array): padding array for filtering. if kernel is 3x3, padding one time if 11x11, padding 5 times.

```python
def paddingArray(array) -> np.ndarray:
    '''
    :param array: np.ndarray
    '''
    row,column = array.shape
    padding_array = np.zeros((row+2, column+2))
    row, column = padding_array.shape
    padding_array[0,0] = array[0,0]
    padding_array[row-1, 0] = array[row-3, 0]
    padding_array[0, column-1] = array[0, column-3]
    padding_array[row-1, column-1] = array[row-3, column-3]
    for c in range(column):
        for r in range(row):
            if (c==0 or c==column-1) and (r==0 or r==row-1):
                continue
            elif c==0:
                padding_array[r,c] = array[r-1, c]
            elif c==column-1:
                padding_array[r,c] = array[r-1, c-2]
            elif r==0:
                padding_array[r,c] = array[r, c-1]
            elif r == row-1:
                padding_array[r,c] = array[r-2, c-1]
            else:
                padding_array[r,c] = array[r-1, c-1]
    return padding_array
```

2. def Laplacian(image, kernel, threshold): create {1,0,-1} array

```python
def Laplacian(image, kernel, threshold) -> np.ndarray:
    '''
    :param image: PIL Image
    :param kernel: numpy ndarray
    :param threshold: integer
    '''
    padding_array = np.array(image)
    print(padding_array.shape)
    size = kernel.shape[0]//2
    print("size = ", size)
    for i in range(size):
        padding_array = paddingArray(padding_array)

    Laplacian_mask = np.zeros(image.size, dtype = int)
    row, column = padding_array.shape
    for r in range(size, row-size):
        for c in range(size, column-size):
            neighborhood_array = padding_array[r-size:r+size+1, c-size:c+size+1] * kernel
            if np.sum(neighborhood_array) >= threshold:
                Laplacian_mask[r-size,c-size] = 1
            elif np.sum(neighborhood_array) <= -threshold:
                Laplacian_mask[r-size,c-size] = -1
            else:
                Laplacian_mask[r-size,c-size] = 0
    return Laplacian_mask
```

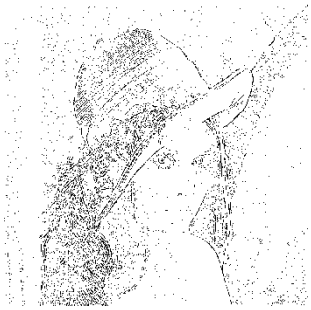
3. def zero_crossing(array, threshold, kernel): zero-crossing edge detection

```python
def zero_crossing(array, threshold, kernel) -> Image:
    size = kernel.shape[0]//2
    padding_array = array
    for i in range(size):
        padding_array = paddingArray(padding_array)
    new_image = Image.new("1", array.shape)
    row, column = padding_array.shape
    for r in range(size, row-size):
        for c in range(size, column-size):
            if (padding_array[r,c] >= threshol size: int m(padding_array[r-size:r+size+1, c-size:c+size+1] <= -threshold) >= 1):
                new_image.putpixel((c-size, r-size), 0)
            else:
                new_image.putpixel((c-size, r-size), 1)

    return new_image
```

Result:

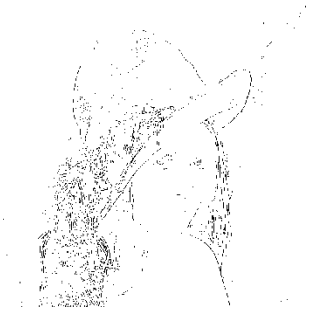
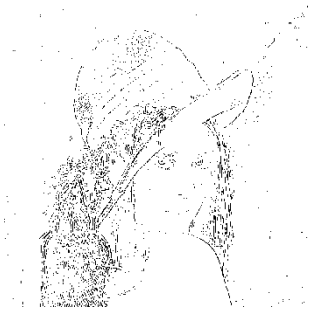
(a) Laplace Mask1 (0, 1, 0, 1, -4, 1, 0, 1, 0): 15

		
原圖	Threshold = 15	Threshold = 20

(b) Laplace Mask2 (1, 1, 1, 1, -8, 1, 1, 1, 1)

		
原圖	Threshold =15	Threshold = 20

(c) Minimum variance Laplacian: 20

		
原圖	Threshold =20	Threshold = 15

(d) Laplace of Gaussian: 3000

		
原圖	Threshold =3000	Threshold = 5000

(e) Difference of Gaussian: 1

		
原圖	Threshold =1	Threshold = 4