HW10 R11944009 何明倩

(a) Laplacian

- Description:
  利用 kernel_1：[[0,1,0], [1,-4,1],[0,1,0]]，kernel_2：1/3*[[1,1,1], [1,-8,1],[1,1,1]]來 Laplacian detect edge。
- Algorithm:
  先建立 convolution functionu 和 zero_crossing_edge function。
  - convolution function with threshold
    針對 kernal 大小進行對應的 padding，上下左右 padding kernel.shape[0]//2，接著從左到右從上到下拜訪 padding_img 的每個 pixel，再針對以此 pixel 為中心擴到對應的 kernel 大小再與 kernel 相乘，若此數值>=閥值，則輸出的該 pixel 值為 1，<=負的閥值則為-1，都不符合則為 0。
  - zero_crossing_edge function
    先針對資料進行上下左右各 1 的 padding，接著從左到右從上到下拜訪每個 pixel，若該數值為 1 時，八鄰域中有-1，則該 pixel 會輸出 0，其餘都皆為 255。

  利用 kernel_1、kernel_2，閥值設 15，先進行 convolution function with threshold，然後再進行 zero_crossing_edge function，即可得到對應的 output。
- Principal code fragment:

```
def convolution(img, kernel, threshold):
    h, w = img.shape
    output = np.array(img).astype(np.float)
    padding = int(kernel.shape[0]//2)
    padding_img = cv2.copyMakeBorder(img, padding, padding, padding, padding, cv2.BORDER_REPLICATE)
    for r in range(h):
        for c in range(w):
            x = r + padding; y = c + padding
            tmp = np.sum(padding_img[x-padding:x+padding+1, y-padding:y+padding+1] * kernel)
            if (tmp >= threshold):
                output[r, c] = 1
            elif (tmp <= -threshold):
                output[r, c] = -1
            else:
                output[r, c] = 0
    return output
```

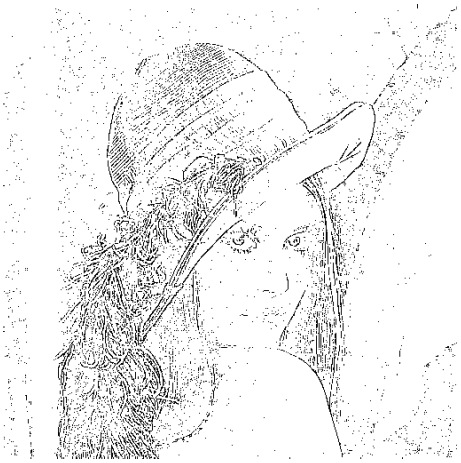
```python
def zero_crossing_edge(img):
    h, w = img.shape
    output = np.zeros(img.shape).astype(np.uint8)
    padding_img = cv2.copyMakeBorder(img, 1, 1, 1, 1, cv2.BORDER_REPLICATE)
    for r in range(h):
        for c in range(w):
            x = r + 1; y = c + 1
            output[r, c] = 255
            if (img[r, c] == 1):
                n = np.sum(padding_img[x-1:x+1+1, y-1:y+1+1] == -1)
                if (n > 0):
                    output[r, c] = 0
    return output
```

```python
def Laplacian(img, kernel, threshold=15):
    output = convolution(img, kernel, threshold=threshold)
    output = zero_crossing_edge(output)
    return output
```

- Result:

kernel_1：[[0,1,0], [1,-4,1],[0,1,0]]

kernel_2：1/3*[[1,1,1], [1,-8,1],[1,1,1]]

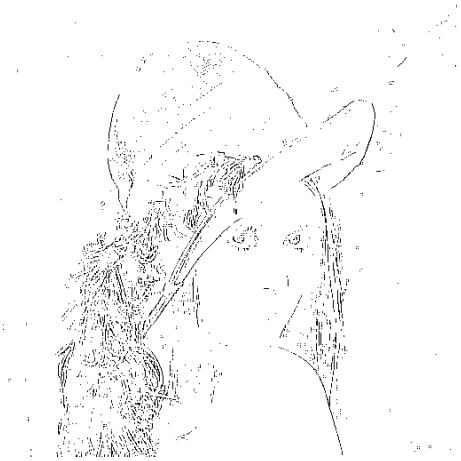


(b) minimum_variance_Laplacian

- Description:
  利用 minimum_variance_Laplacian 做 edge detect
- Algorithm:
  先建立 kernel: 1/3*[[2, -1, 2],[-1, -4, -1], [2, -1, 2]]，閥值設 20，先進行 convolution function with threshold，然後再進行 zero_crossing_edge function，即可得到對應的 output。
- Principal code fragment:

```
def minimum_variance_Laplacian(img, threshold=20):
    kernel = 1/3*np.array([[2, -1, 2],[-1, -4, -1], [2, -1, 2]])
    output = convolution(img, kernel, threshold=threshold)
    output = zero_crossing_edge(output)
    return output
```

- Result:



(c) Laplacian_of_Gaussian

- Description:
  利用 Laplacian of Gaussian 找 edge
- Algorithm:

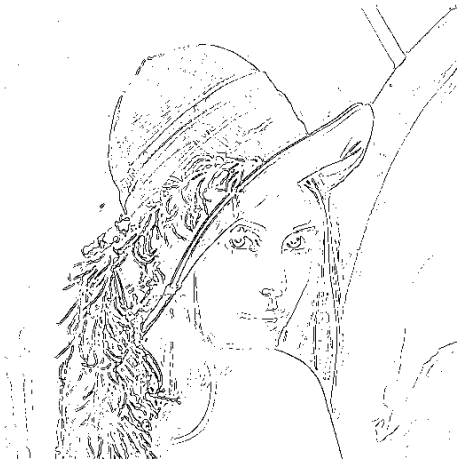
先建立 kernel = np.array([[0,0,0,-1,-1,-2,-1,-1,0,0,0],
[0,0,-2,-4,-8,-9,-8,-4,-2,0,0],
[0,-2,-7,-15,-22,-23,-22,-15,-7,-2,0],
[-1,-4,-15,-24,-14,-1,-14,-24,-15,-4,-1],
[-1,-8,-22,-14,52,103,52,-14,-22,-8,-1],
[-2,-9,-23,-1,103,178,103,-1,-23,-9,-2],
[-1,-8,-22,-14,52,103,52,-14,-22,-8,-1],
[-1,-4,-15,-24,-14,-1,-14,-24,-15,-4,-1],
[0,-2,-7,-15,-22,-23,-22,-15,-7,-2,0],

[0,0,-2,-4,-8,-9,-8,-4,-2,0,0],
[0,0,0,-1,-1,-2,-1,-1,0,0,0]])，閥值設 3000，先進行 convolution function with threshold，然後再進行 zero_crossing_edge function，即可得到對應的 output。

- Principal code fragment:

```
def Laplacian_of_Gaussian(img, threshold=3000):
    kernel = np.array([[0,0,0,-1,-1,-2,-1,-1,0,0,0],
                       [0,0,-2,-4,-8,-9,-8,-4,-2,0,0],
                       [0,-2,-7,-15,-22,-23,-22,-15,-7,-2,0],
                       [-1,-4,-15,-24,-14,-1,-14,-24,-15,-4,-1],
                       [-1,-8,-22,-14,52,103,52,-14,-22,-8,-1],
                       [-2,-9,-23,-1,103,178,103,-1,-23,-9,-2],
                       [-1,-8,-22,-14,52,103,52,-14,-22,-8,-1],
                       [-1,-4,-15,-24,-14,-1,-14,-24,-15,-4,-1],
                       [0,-2,-7,-15,-22,-23,-22,-15,-7,-2,0],
                       [0,0,-2,-4,-8,-9,-8,-4,-2,0,0],
                       [0,0,0,-1,-1,-2,-1,-1,0,0,0]])
    output = convolution(img, kernel, threshold=threshold)
    output = zero_crossing_edge(output)
    return output
```

- Result:



(d) Difference_of_Gaussian

- Description:

  利用 Difference of Gaussian 做 edge detect

  Algorithm:

  先建立 kernel = np.array([[-1,-3,-4,-6,-7,-8,-7,-6,-4,-3,-1],
  [-3,-5,-8,-11,-13,-13,-13,-11,-8,-5,-3],
  [-4,-8,-12,-16,-17,-17,-17,-16,-12,-8,-4],
  [-6,-11,-16,-16,0,15,0,-16,-16,-11,-6],
  [-7,-13,-17,0,85,160,85,0,-17,-13,-7],
  [-8,-13,-17,15,160,283,160,15,-17,-13,-8],
  [-7,-13,-17,0,85,160,85,0,-17,-13,-7],
  [-6,-11,-16,-16,0,15,0,-16,-16,-11,-6],
  [-4,-8,-12,-16,-17,-17,-17,-16,-12,-8,-4],
  [-3,-5,-8,-11,-13,-13,-13,-11,-8,-5,-3],
  [-1,-3,-4,-6,-7,-8,-7,-6,-4,-3,-1]])，閥值設 1，先進行 convolution function with threshold，然後再進行 zero_crossing_edge function，即可得到對應的 output。

- Principal code fragment:

```
def Difference_of_Gaussian(img, threshold=1):
    kernel = np.array([[-1,-3,-4,-6,-7,-8,-7,-6,-4,-3,-1],
                       [-3,-5,-8,-11,-13,-13,-13,-11,-8,-5,-3],
                       [-4,-8,-12,-16,-17,-17,-17,-16,-12,-8,-4],
                       [-6,-11,-16,-16,0,15,0,-16,-16,-11,-6],
                       [-7,-13,-17,0,85,160,85,0,-17,-13,-7],
                       [-8,-13,-17,15,160,283,160,15,-17,-13,-8],
                       [-7,-13,-17,0,85,160,85,0,-17,-13,-7],
                       [-6,-11,-16,-16,0,15,0,-16,-16,-11,-6],
                       [-4,-8,-12,-16,-17,-17,-17,-16,-12,-8,-4],
                       [-3,-5,-8,-11,-13,-13,-13,-11,-8,-5,-3],
                       [-1,-3,-4,-6,-7,-8,-7,-6,-4,-3,-1]])
    output = convolution(img, kernel, threshold=threshold)
    output = zero_crossing_edge(output)
    return output
```

- Result: