Computer Vision - Assignment 10

R09922A04 資工所人工智慧組 黃品硯

[Code]

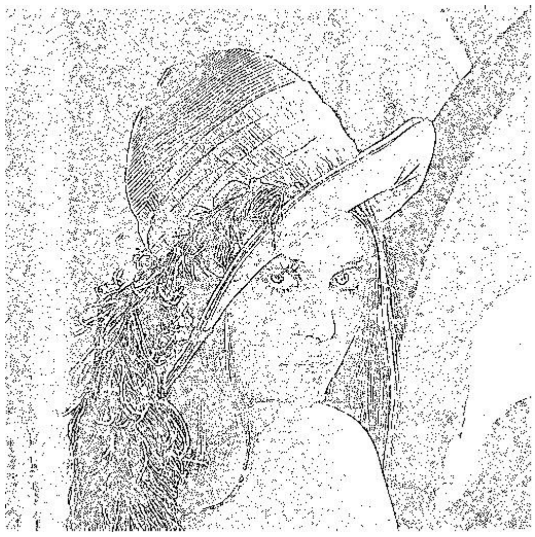
```
def apply_mask(img, y, x, mask):
    r = 0
    for m_y in range(mask_height):
        for m_x in range(mask_width):
            if y + m_y < img_height and x + m_x < img_width:
                r += img[y + m_y, x + m_x] * mask[m_y, m_x]
    return r


def zero_crossing_edge_detection(img, mask, threshold):
    pad_img = cv2.copyMakeBorder(img, offset, offset, offset, offset, cv2.BORDER_REFLECT)

    for y in range(height):
        for x in range(width):
            t = apply_mask(pad_img, y + offset, x + offset, mask)
            if t >= threshold:
                mask_img[y, x] = 1
            elif t <= -threshold:
                mask_img[y, x] = -1
            else:
                mask_img[y, x] = 0

    for y in range(height):
        for x in range(width):
            is_zero_crossing = False

            if mask_img[y, x] >= 1:
                for n in get_neighbors_pixel(mask_img, y, x):
                    if n <= -1:
                        out_img[y, x] = 0
                        is_zero_crossing = True
                        break

            if not is_zero_crossing:
                out_img[y, x] = 255

    return out_img
```

(a) Laplace Mask1 (0, 1, 0, 1, -4, 1, 0, 1, 0), Threshold=15

Apply the mask on each of the pixels and get the value t. If t is larger or equal to the threshold then set the pixel to 1, if it is smaller or equal to the -1*threshold then set the pixel to -1, otherwise set it to 0. Check every pixel's value on the new image, if the value is larger or equal to 1 and also there exists a neighbor pixel that is smaller or equal to -1 then set the pixel value to black (value 0), otherwise set it to white (value 255).

[Code]

```
mask = np.array([[0, 1, 0], [1, -4, 1], [0, 1, 0]])
zero_crossing_edge_detection(img, mask, 15)
```

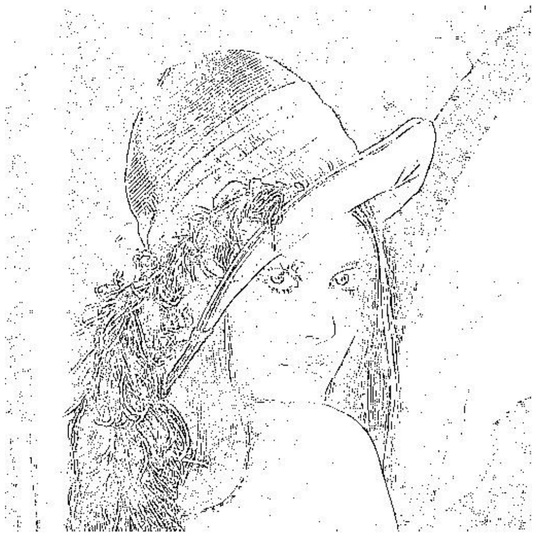


(b) Laplace Mask2 (1, 1, 1, 1, -8, 1, 1, 1, 1), Threshold=15

Same procedure with (a) but with a different threshold and masks.

[Code]

```
mask = np.array([[1, 1, 1], [1, -8, 1], [1, 1, 1]]) / 3
zero_crossing_edge_detection(img, mask, 15)
```

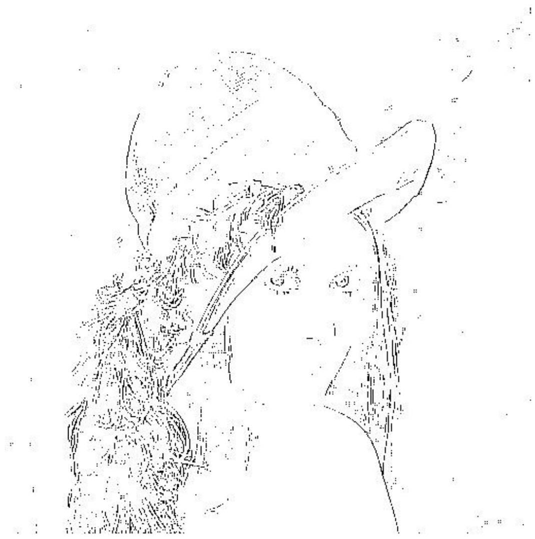


(c) Minimum variance Laplacian, Threshold=20

Same procedure with (a) but with a different threshold and masks.

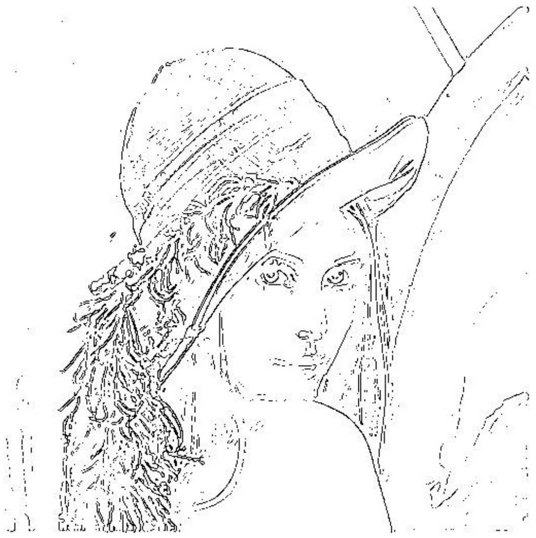
[Code]

```
mask = np.array([[2, -1, 2],[-1, -4, -1],[2, -1, 2]])/3
zero_crossing_edge_detection(img, mask, 20)
```



(d) Laplace of Gaussian, Threshold=3000

Same procedure with (a) but with a different threshold and masks.

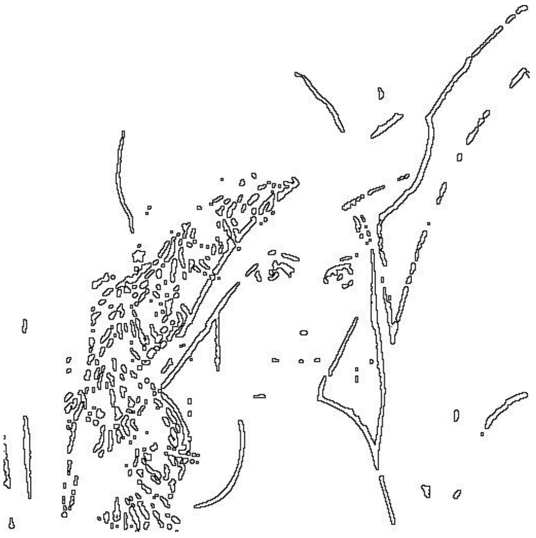


[Code]

```
mask = 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],
])
zero_crossing_edge_detection(img, mask, 3000)
```

(e) Difference of Gaussian, Threshold=1

Same procedure with (a) but with a different threshold and masks.



[Code]

```
mask = 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],
])
zero_crossing_edge_detection(img, mask, 1)
```