CV HW10

R11922196 林佑鑫

本次作業 algorithm 方面都相同，只有 mask 更換

Brief description, algorithm:

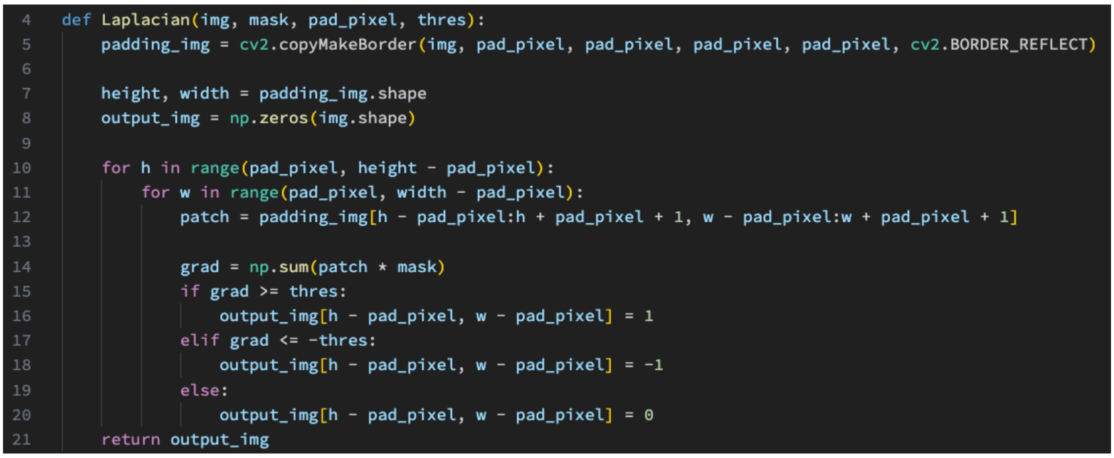
Step 1: 建立 mask

Step 2: 用 mask 做 Laplacian algorithm

Step 3: 做 Zero-Crossing algorithm

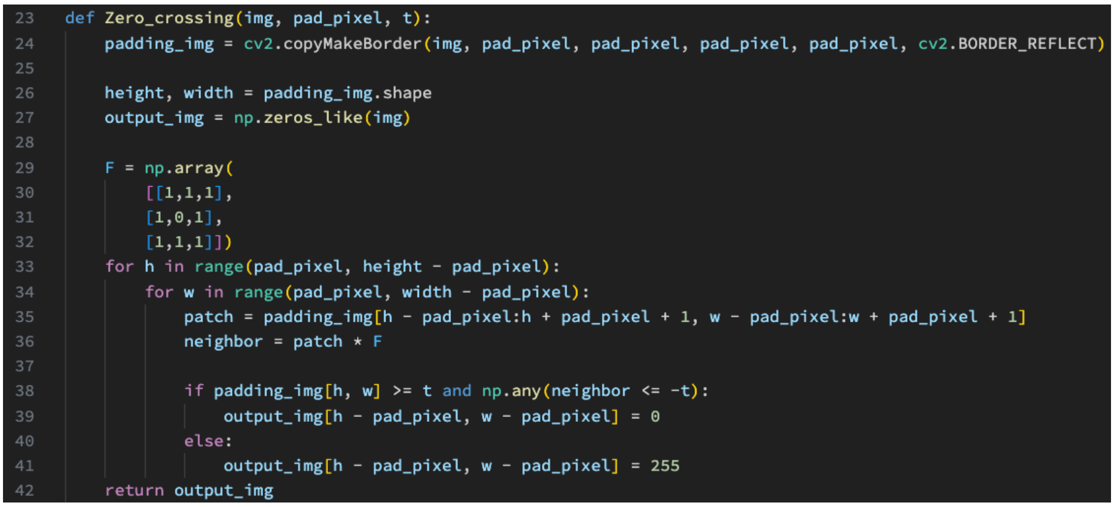
Principal code fragment:

Laplacian:



```python
def Laplacian(img, mask, pad_pixel, thres):
    padding_img = cv2.copyMakeBorder(img, pad_pixel, pad_pixel, pad_pixel, pad_pixel, cv2.BORDER_REFLECT)

    height, width = padding_img.shape
    output_img = np.zeros(img.shape)

    for h in range(pad_pixel, height - pad_pixel):
        for w in range(pad_pixel, width - pad_pixel):
            patch = padding_img[h - pad_pixel:h + pad_pixel + 1, w - pad_pixel:w + pad_pixel + 1]

            grad = np.sum(patch * mask)
            if grad >= thres:
                output_img[h - pad_pixel, w - pad_pixel] = 1
            elif grad <= -thres:
                output_img[h - pad_pixel, w - pad_pixel] = -1
            else:
                output_img[h - pad_pixel, w - pad_pixel] = 0
    return output_img
```

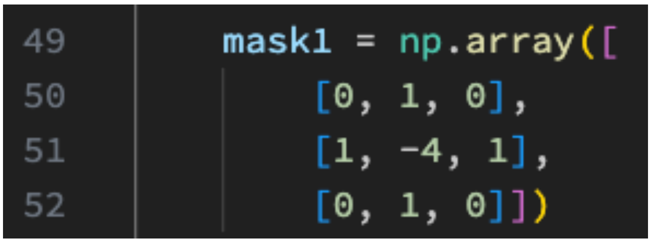
Zero-Crossing:



```python
def Zero_crossing(img, pad_pixel, t):
    padding_img = cv2.copyMakeBorder(img, pad_pixel, pad_pixel, pad_pixel, pad_pixel, cv2.BORDER_REFLECT)

    height, width = padding_img.shape
    output_img = np.zeros_like(img)

    F = np.array(
        [[1,1,1],
        [1,0,1],
        [1,1,1]])
    for h in range(pad_pixel, height - pad_pixel):
        for w in range(pad_pixel, width - pad_pixel):
            patch = padding_img[h - pad_pixel:h + pad_pixel + 1, w - pad_pixel:w + pad_pixel + 1]
            neighbor = patch * F

            if padding_img[h, w] >= t and np.any(neighbor <= -t):
                output_img[h - pad_pixel, w - pad_pixel] = 0
            else:
                output_img[h - pad_pixel, w - pad_pixel] = 255
    return output_img
```

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

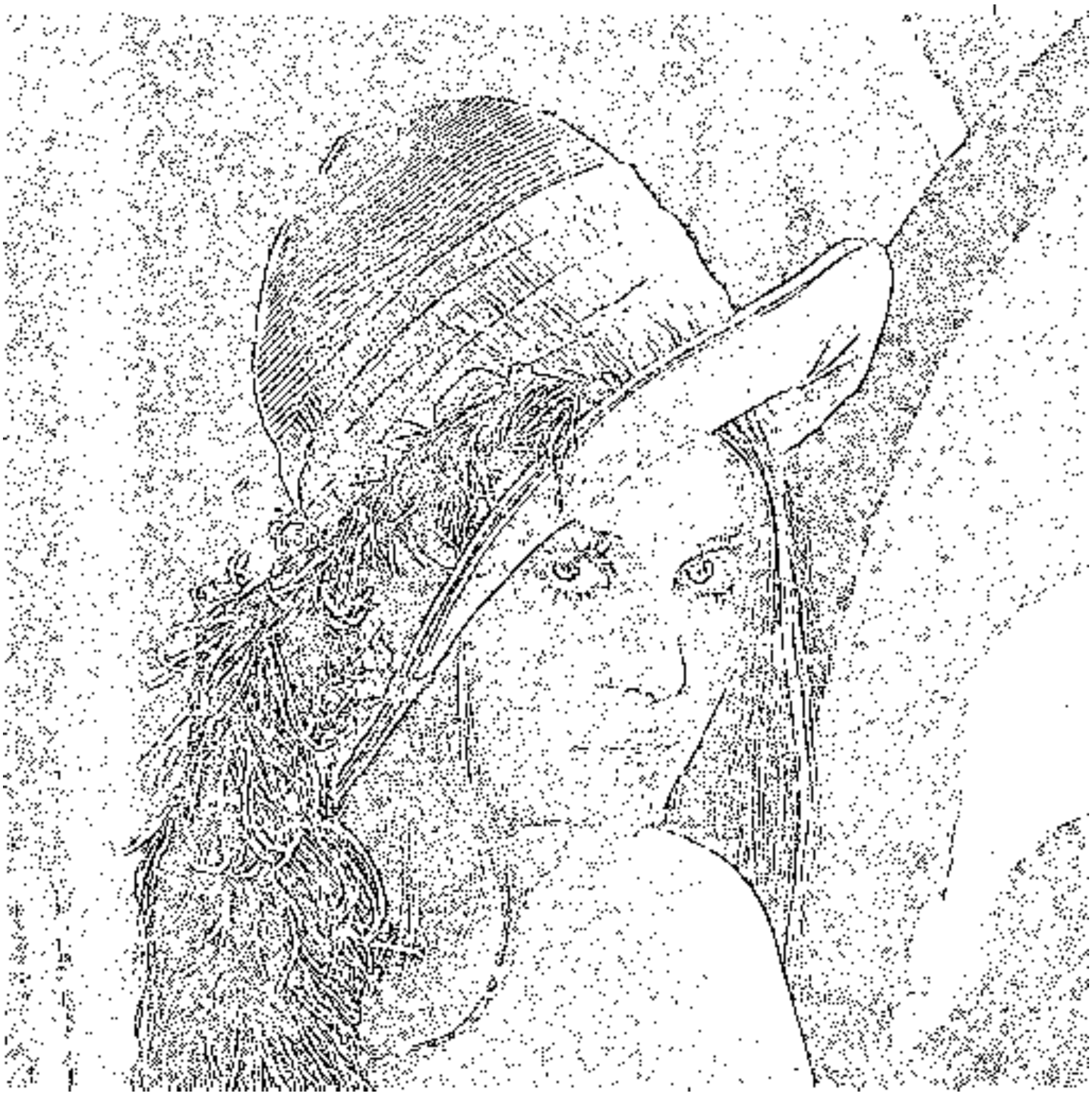
Principal code fragment:



```
    mask1 = np.array([
        [0, 1, 0],
        [1, -4, 1],
        [0, 1, 0]])
```

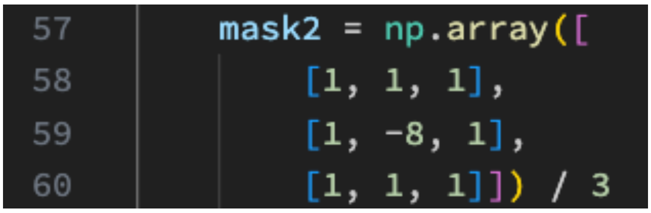
Parameters: None

Resulting image:



(b) Laplace Mask2 (1, 1, 1, 1, -8, 1, 1, 1, 1): 15

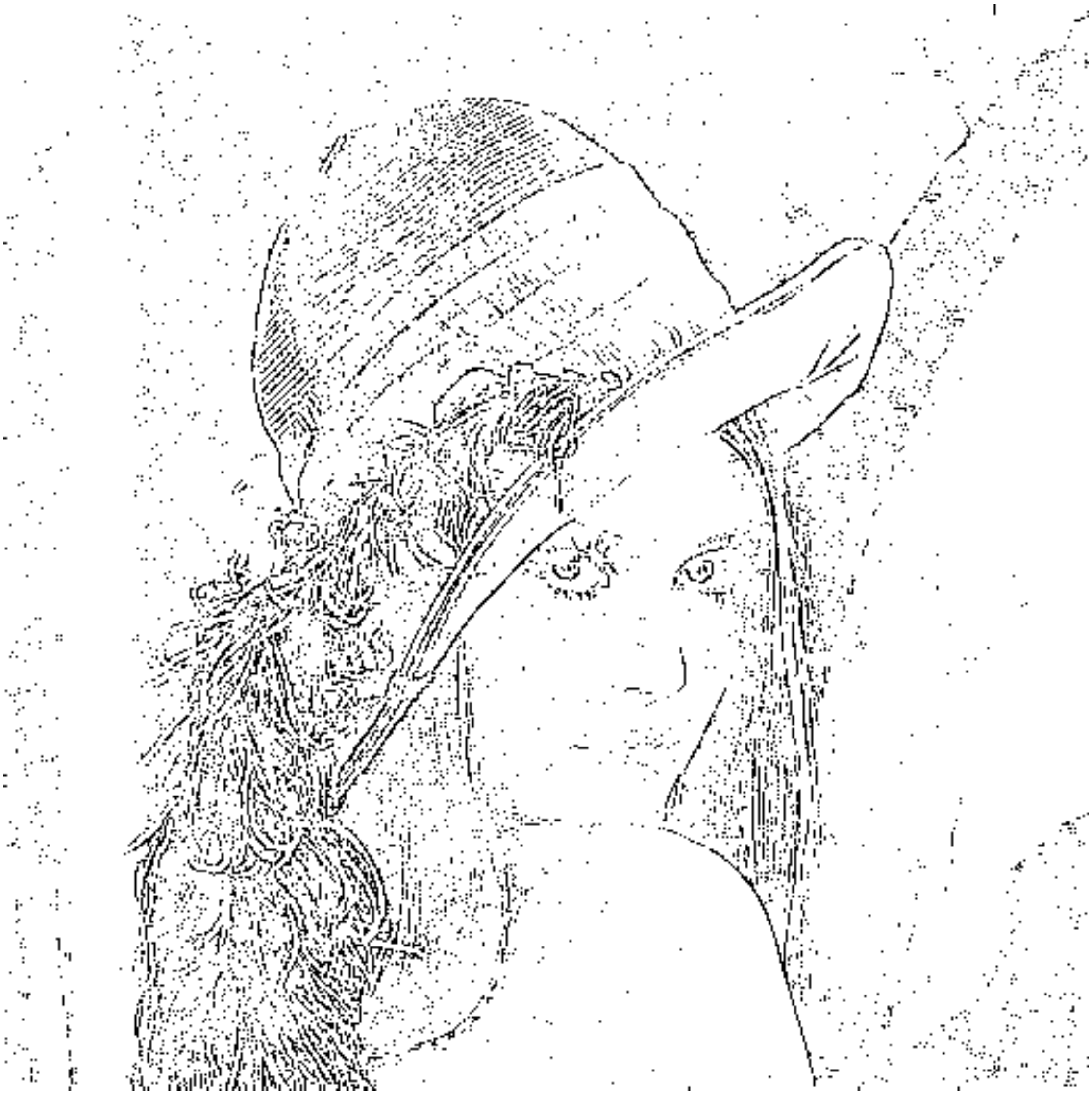
Principal code fragment:



```
    mask2 = np.array([
        [1, 1, 1],
        [1, -8, 1],
        [1, 1, 1]]) / 3
```

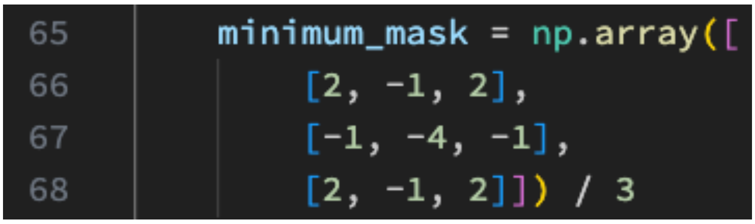
Parameters: None

Resulting image:



(c) Minimum variance Laplacian: 20

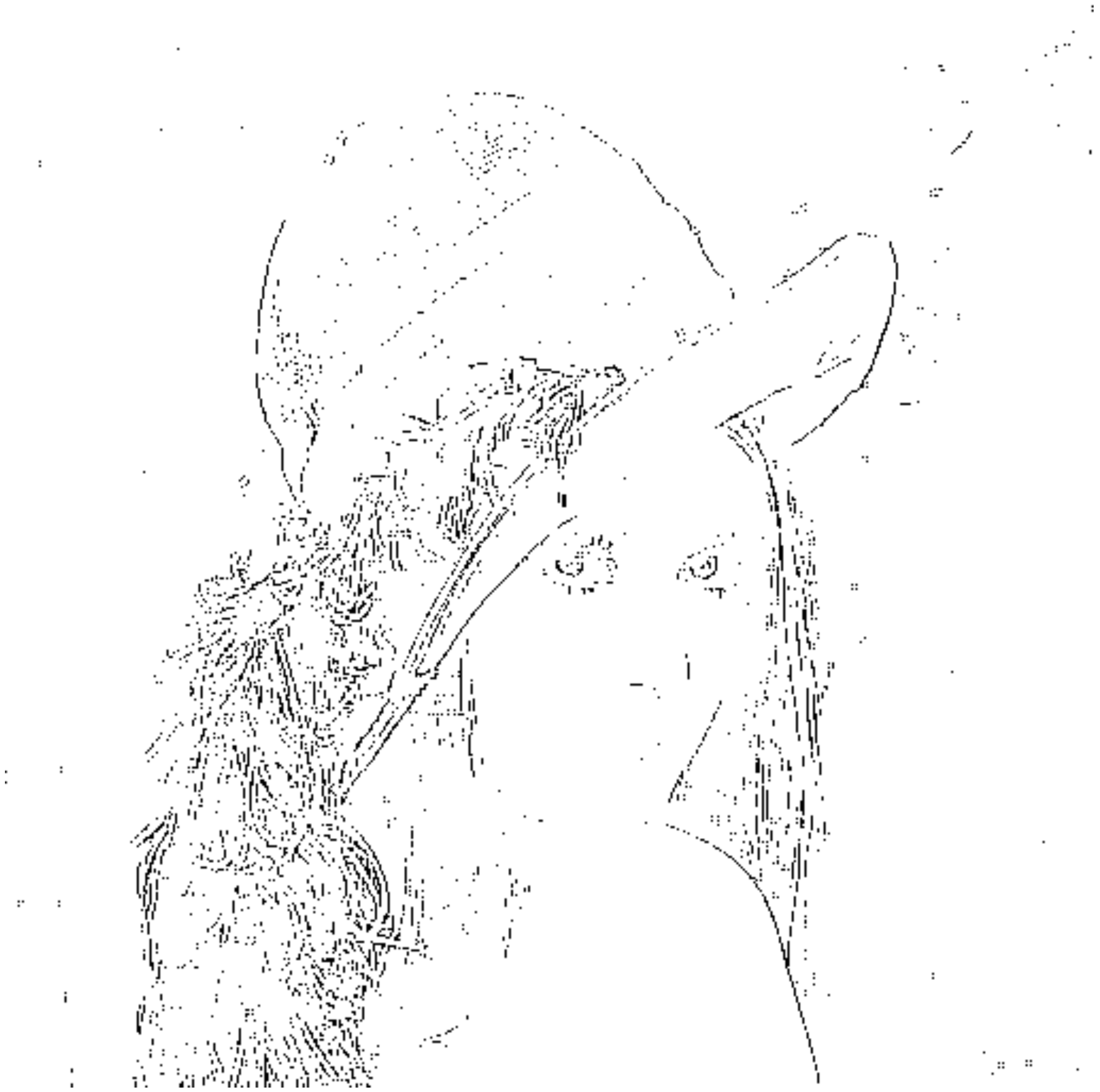
Principal code fragment:



```
    minimum_mask = np.array([
        [2, -1, 2],
        [-1, -4, -1],
        [2, -1, 2]]) / 3
```

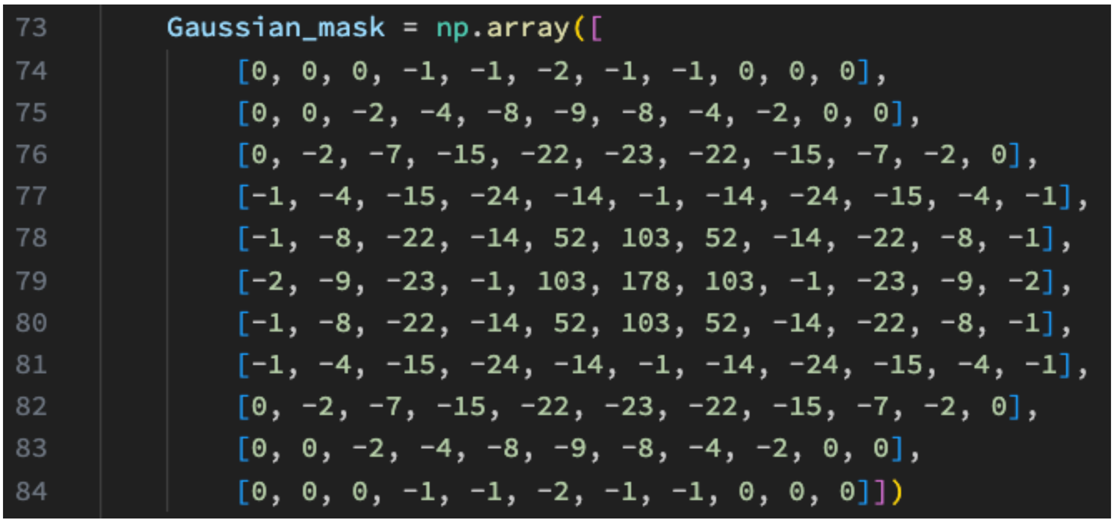
Parameters: None

Resulting image:



(d) Laplace of Gaussian: 3000

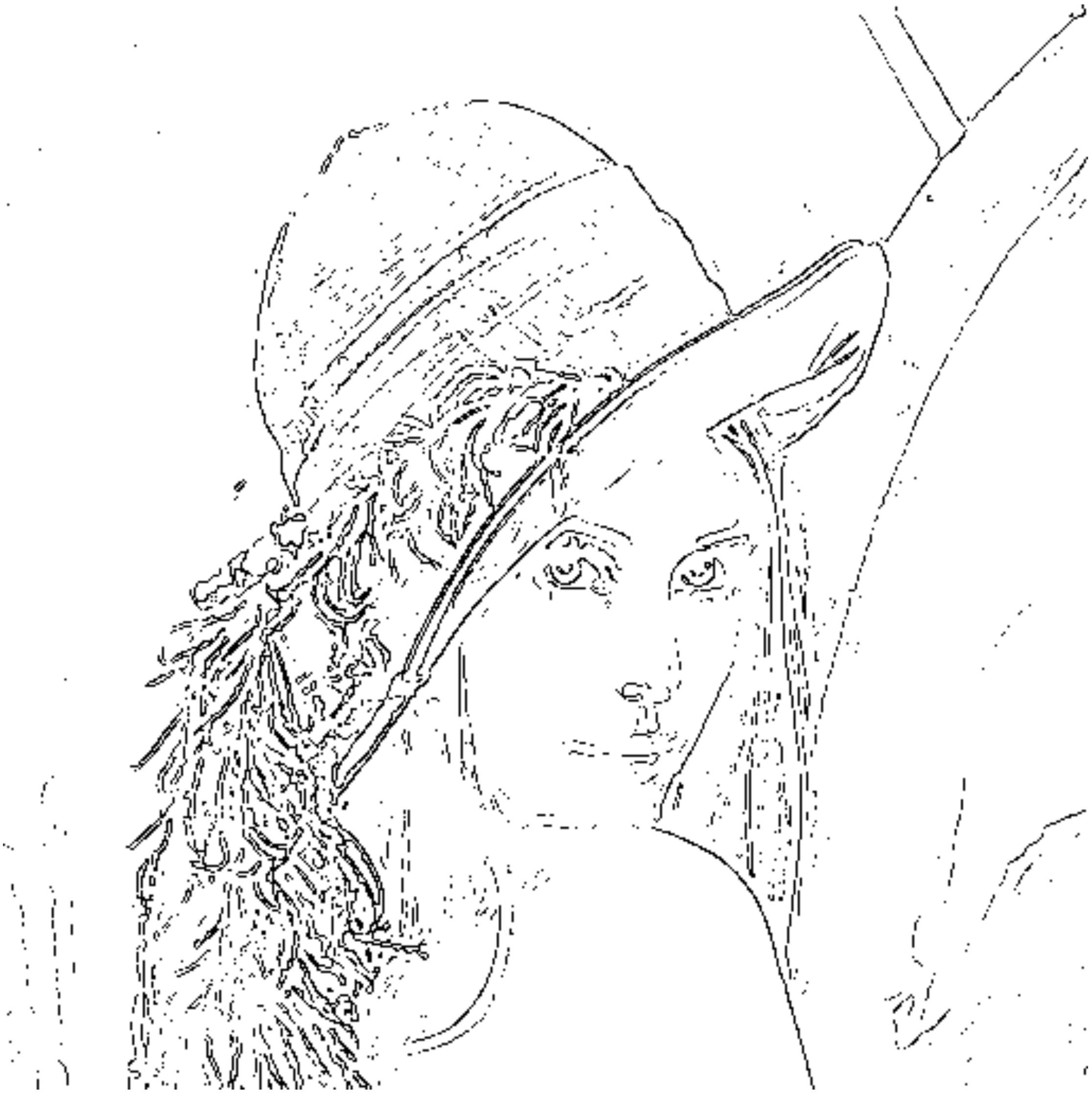
Principal code fragment:



```python
    Gaussian_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]])
```

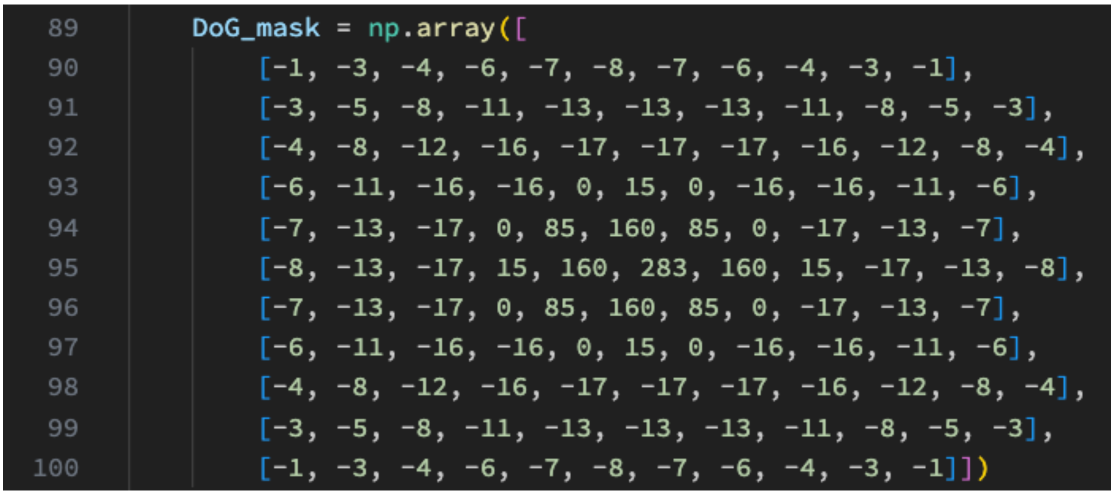
Parameters: None

Resulting image:



(e) **Difference of Gaussian: 1**

Principal code fragment:



```
    DoG_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]])
```

Parameters: None

Resulting image: