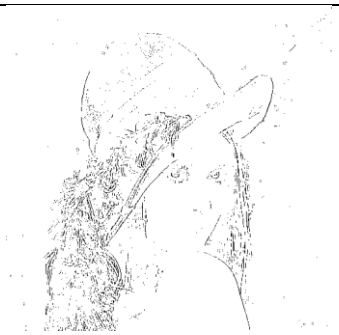
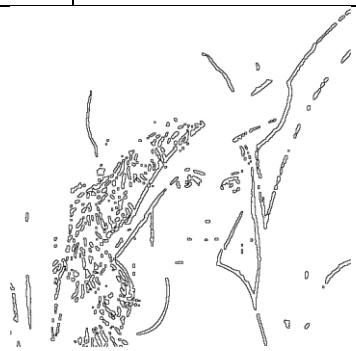
2021 CV HW10 Report

Results

<table>
<tr>
<td> 
Laplace Mask 1
Threshold = 15</td>
<td> 
Laplace Mask 2
Threshold = 15</td>
<td> 
Minimum variance Laplacian
Threshold = 20</td>
</tr>
<tr>
<td> 
Laplace of Gaussian
Threshold = 3000</td>
<td> 
Difference of Gaussian
Threshold = 1</td>
</tr>
</table>

Zero Crossing Edge Detection

以下說明 4 個 function 的作法，其餘 Detector 的實作只需呼叫這些 function 即可。

convolution : 計算該 pixel 與 mask 運算後的結果。

```
def convolution(img, i, j, mask):
    rows, cols = mask.shape
    value = 0
    for a in range(rows):
        for b in range(cols):
            value += img[i+a][j+b] * mask[a][b]
    return value
```

edgeDetection : 先對圖片做 padding，接著去計算每個 pixel 的 gradient，若 gradient 大於等於 threshold 則 value 設為 1；若 gradient 小於等於 threshold 的相反數則設為 -1；其餘設為 0。最後做 zeroCrossing（後面會提到的 function）。

```
def edgeDetection(img, threshold, kernel):
    output = np.zeros((h, w), np.int8)
```

```
    padding = kernel.shape[0] // 2
    img_padding = cv2.copyMakeBorder(img, padding, padding, padding, padding,
cv2.BORDER_REFLECT)
    for i in range(h):
        for j in range(w):
            gradient = convolution(img_padding, i, j, kernel)
            if gradient >= threshold:
                output[i][j] = 1
            elif gradient <= threshold * (-1):
                output[i][j] = -1
            else:
                output[i][j] = 0
    output = zeroCrossing(output, padding)
    return output
```

checkNeighbors：檢查該 pixel 周圍 3x3 鄰居是否有 value 值為 -1 的。

```
def checkNeighbors(img, i, j):
    for a in range(-1, 2):
        for b in range(-1, 2):
            if img[i+a][j+b] == -1:
                return 1
    return 0
```

zeroCrossing：對圖片先做 padding，接著去查看 value 值為 1 的 pixel 的周圍 3x3 的鄰居是否有 value 值為 -1 的，若有則 pixel value 設為 0；其餘都設為 255。

```
def zeroCrossing(img, padding):
    output = np.zeros((h, w), np.uint8)
    img_padding = cv2.copyMakeBorder(img, padding, padding, padding, padding,
cv2.BORDER_REFLECT)
    hh, ww = img_padding.shape
    for i in range(padding, hh-padding):
        for j in range(padding, ww-padding):
            if img_padding[i][j] == 1:
                if checkNeighbors(img_padding, i, j) == 1:
                    output[i-padding][j-padding] = 0
                    continue
            output[i-padding][j-padding] = 255
    return output
```

以下為各個 Detector 所使用的 mask 和 threshold：

(a) Laplace Mask 1

```
Laplace_1 = np.array([[0, 1, 0],
                      [1,-4, 1],
                      [0, 1, 0]])
laplace1_img = edgeDetection(image, 15, Laplace_1) # threshold = 15
```

(b) Laplace Mask 2

```
Laplace_2 = np.array([[1, 1, 1],
                      [1,-8, 1],
                      [1, 1, 1]])
Laplace_2 = np.divide(Laplace_2, 3) # 1/3 * mask
laplace2_img = edgeDetection(image, 15, Laplace_2) # threshold = 15
```

(c) Minimum-variance Laplacian

```
Laplace_min = np.array([[ 2, -1, 2],
                        [-1, -4,-1],
                        [ 2, -1, 2]])
Laplace_min = np.divide(Laplace_min, 3) # 1/3 * mask
laplaceMin_img = edgeDetection(image, 20, Laplace_min) # threshold = 20
```

(d) Laplacian of Gaussian

```
LoG = 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]])
LoG_img = edgeDetection(image, 3000, LoG) # threshold = 3000
```

(e) Difference of Gaussian

```
DoG = 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]])
DoG_img = edgeDetection(image, 1, DoG) # threshold = 1
```