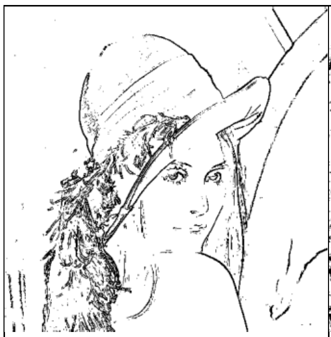
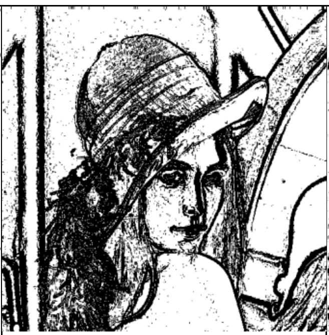
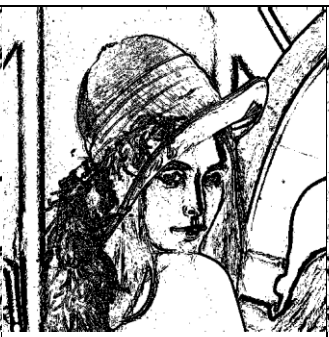
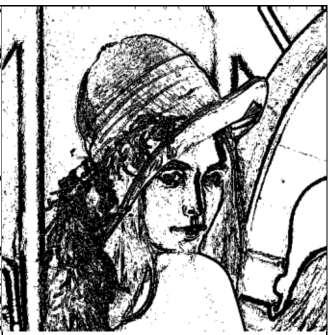
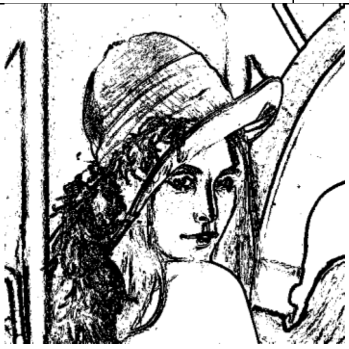
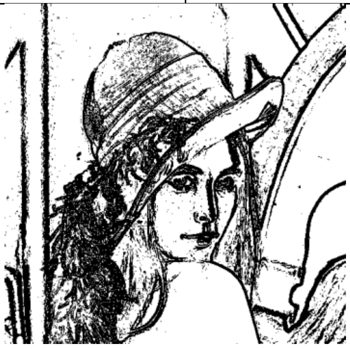
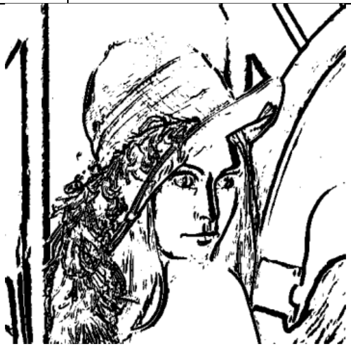
2021 CV HW9 Report

Results

			
Robert's Operator Threshold = 30	Prewitt's Edge Detector Threshold = 24	Sobel's Edge Detector Threshold = 38	Frei and Chen's Gradient Operator Threshold = 30

		
Kirsch's Compass Operator Threshold = 135	Robinson's Compass Operator Threshold = 43	Nevatia-Babu 5x5 Operator Threshold = 12500

Edge Detection

以下說明 3 個 function 的作法，其餘 Operator 的實作只需呼叫這些 function 即可。

計算該 pixel 與 mask 運算後的結果。

```
def mask(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
```

先對圖片做 padding，接著去計算每個 pixel 的 gradient（ex：gradient magnitude: $\sqrt{r_1^2 + r_2^2}$ ），若 gradient 大於等於 threshold 則 pixel value 設為 0；反之，設為 255。（通用於底下的 **(a)~(d)** 的 Operator）

```
def edgeDetector(img, threshold, mask1, mask2, start = 0):
    output = np.zeros((h, w), np.uint8)
    img_padding = cv2.copyMakeBorder(img, 1, 1, 1, 1, cv2.BORDER_REFLECT)
    for i in range(h):
        for j in range(w):
            gradient = math.sqrt(mask(img_padding, i+start, j+start, mask1)**2 +
mask(img_padding, i+start, j+start, mask2)**2)
            if gradient >= threshold:
                output[i][j] = 0
            else:
                output[i][j] = 255
    return output
```

先對圖片做 padding，接著去計算每個 pixel 和各個 mask 的結果，取所有結果中的最大值作為 gradient（ex：gradient magnitude: $\max_{n,n=0,\ldots,7} k_n$ ），若最大值大於等於 threshold 則 pixel value 設為 0；反之，設為 255。（此 function 可通用於底下的 **(e)~(g)** 的 Operator）

```
def compassOperator(img, threshold, masks, padding = 1):
    output = np.zeros((h, w), np.uint8)
    img_padding = cv2.copyMakeBorder(img, padding, padding, padding, padding,
cv2.BORDER_REFLECT)
    for i in range(h):
        for j in range(w):
            maskValue = []
            for k in range(len(masks)):
                maskValue.append(mask(img_padding, i, j, masks[k]))
            if max(maskValue) >= threshold:
                output[i][j] = 0
            else:
                output[i][j] = 255
    return output
```

以下為各 Operator 所使用的 mask 和 threshold：

(a) Robert's Operator

```
robert_1 = np.array([[-1, 0],
                     [0, 1]])
robert_2 = np.array([[0, -1],
                     [1, 0]])
robert_img = edgeDetector(image, 30, robert_1, robert_2, 1) # threshold = 30
```

(b) Prewitt's Edge Detector

```
prewitt_1 = np.array([[-1, -1, -1],
                      [ 0,  0,  0],
                      [ 1,  1,  1]])
```

```
prewitt_2 = np.array([[-1,  0,  1],
                      [-1,  0,  1],
                      [-1,  0,  1]])
prewitt_img = edgeDetector(image, 24, prewitt_1, prewitt_2) # threshold = 24
```

(c) Sobel's Edge Detector

```
sobel_1 = np.array([[-1, -2, -1],
                    [ 0,  0,  0],
                    [ 1,  2,  1]])
sobel_2 = np.array([[-1,  0,  1],
                    [-2,  0,  2],
                    [-1,  0,  1]])
sobel_img = edgeDetector(image, 38, sobel_1, sobel_2) # threshold = 38
```

(d) Frei and Chen's Gradient Operator

```
frei_1 = np.array([[-1, -1*math.sqrt(2), -1],
                   [ 0,  0,  0],
                   [ 1,  math.sqrt(2),  1]])
frei_2 = np.array([[-1,  0,  1],
                   [-1*math.sqrt(2),  0,  math.sqrt(2)],
                   [-1,  0,  1]])
frei_img = edgeDetector(image, 30, frei_1, frei_2) # threshold = 30
```

(e) Kirsch's Compass Operator

```
k_0 = np.array([[-3, -3,  5], [-3,  0,  5], [-3, -3,  5]])
k_1 = np.array([[-3,  5,  5], [-3,  0,  5], [-3, -3, -3]])
k_2 = np.array([[ 5,  5,  5], [-3,  0, -3], [-3, -3, -3]])
k_3 = np.array([[ 5,  5, -3], [ 5,  0, -3], [-3, -3, -3]])
k_4 = np.array([[ 5, -3, -3], [ 5,  0, -3], [ 5, -3, -3]])
k_5 = np.array([[-3, -3, -3], [ 5,  0, -3], [ 5,  5, -3]])
k_6 = np.array([[-3, -3, -3], [-3,  0, -3], [ 5,  5,  5]])
k_7 = np.array([[-3, -3, -3], [-3,  0,  5], [-3,  5,  5]])
kirsches = [k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7]
kirsch_img = compassOperator(image, 135, kirsches) # threshold = 135
```

(f) Robinson's Compass Operator

```
r_0 = np.array([[-1,  0,  1], [-2,  0,  2], [-1,  0,  1]])
r_1 = np.array([[ 0,  1,  2], [-1,  0,  1], [-2, -1,  0]])
r_2 = np.array([[ 1,  2,  1], [ 0,  0,  0], [-1, -2, -1]])
r_3 = np.array([[ 2,  1,  0], [ 1,  0, -1], [ 0, -1, -2]])
r_4 = np.array([[ 1,  0, -1], [ 2,  0, -2], [ 1,  0, -1]])
r_5 = np.array([[ 0, -1, -2], [ 1,  0, -1], [ 2,  1,  0]])
r_6 = np.array([[-1, -2, -1], [ 0,  0,  0], [ 1,  2,  1]])
r_7 = np.array([[-2, -1,  0], [-1,  0,  1], [ 0,  1,  2]])
robinsons = [r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7]
robinson_img = compassOperator(image, 43, robinsons) # threshold = 43
```

(g) Nevatia-Babu 5x5 Operator

```
n_0 = np.array([[ 100, 100, 100, 100, 100],
                [ 100, 100, 100, 100, 100],
                [   0,   0,   0,   0,   0],
                [-100,-100,-100,-100,-100],
                [-100,-100,-100,-100,-100]])
n_30 = np.array([[ 100, 100, 100, 100, 100],
                 [ 100, 100, 100,  78, -32],
                 [ 100,  92,   0, -92,-100],
                 [  32, -78,-100,-100,-100],
                 [-100,-100,-100,-100,-100]])
n_60 = np.array([[ 100, 100, 100,  32,-100],
                 [ 100, 100,  92, -78,-100],
                 [ 100, 100,   0,-100,-100],
                 [ 100,  78, -92,-100,-100],
                 [ 100, -32,-100,-100,-100]])
n_270 = np.array([[-100,-100, 0, 100, 100],
                  [-100,-100,  0, 100, 100],
                  [-100,-100,  0, 100, 100],
                  [-100,-100,  0, 100, 100],
                  [-100,-100,  0, 100, 100]])
n_300 = np.array([[-100,  32, 100, 100, 100],
                  [-100, -78,  92, 100, 100],
                  [-100,-100,   0, 100, 100],
                  [-100,-100, -92,  78, 100],
                  [-100,-100,-100, -32, 100]])
n_330 = np.array([[ 100, 100, 100, 100, 100],
                  [ -32,  78, 100, 100, 100],
                  [-100, -92,   0,  92, 100],
                  [-100,-100,-100, -78,  32],
                  [-100,-100,-100,-100,-100]])
nevatias = [n_0, n_30, n_60, n_270, n_300, n_330]
nevatia_img = compassOperator(image, 12500, nevatias, 2) # threshold = 12500
```