Computer Vision_HW9

- Command Line

  python main.py Robert
  python main.py Prewitt
  python main.py Sobel
  python main.py FAC
  python main.py Kirsch
  python main.py Robinson
  python main.py NevatiaBabu

- Threshold

  Enter Threshold: (*choose any threshold you want*)

- Two Main Function

  1. 平方和開根號

```
def Magnitude(pixel, mask):
    num = len(mask)
    sizeY = len(mask[0])
    sizeX = len(mask[0][0])
    magnitude = []

    for i in range(num):
        r = 0
        for y in range(sizeY):
            for x in range(sizeX):
                r += pixel[y][x] * mask[i][y][x]

        magnitude.append(r**2)

    return math.sqrt(sum(magnitude))
```

  2. 最大值

```python
def MaxMagnitude(pixel, mask):
    num = len(mask)
    sizeY = len(mask[0])
    sizeX = len(mask[0][0])
    magnitude = []

    for i in range(num):
        r = 0
        for y in range(sizeY):
            for x in range(sizeX):
                r += pixel[y][x] * mask[i][y][x]

        magnitude.append(r)

    return max(magnitude)
```

- Robert's Operator

```python
def Robert(img, threshold):
    img_new = np.full(img.shape, 255, np.int)
    mask = [[[-1,0],[0,1]],[[0,-1],[1,0]]]
    for y in range(img.shape[0]-1):
        for x in range(img.shape[1]-1):
            neighbors = []
            neighbors.append([img[y][x],img[y][x+1]])
            neighbors.append([img[y+1][x],img[y+1][x+1]])
            G = Magnitude(neighbors, mask)

            if G > threshold:
                img_new[y][x] = 0
            else:
                img_new[y][x] = 255

    return img_new
```

- Prewitt's Edge Detector

Same as Robert's Operator, except the difference of masks

```
mask = [[[-1,-1,-1],[0,0,0],[1,1,1]],[[-1,0,1],[-1,0,1],[-1,0,1]]]
```

- Sobel's Edge Detector

Same as Robert's Operator, except the difference of masks

```
mask = [[[-1,-2,-1],[0,0,0],[1,2,1]],[[-1,0,1],[-2,0,2],[-1,0,1]]]
```

- Frei and Chen's Gradient Operator

Same as Robert's Operator, except the difference of masks

```
value = math.sqrt(2)
img_new = np.full(img.shape, 255, np.int)
mask = [[[-1,-value,-1],[0,0,0],[1,value,1]],[[-1,0,1],[-value,0,value],[-1,0,1]]]
```

- Kirsch's Compass Operator

```
def Kirsch(img, threshold):
    img_new = np.full(img.shape, 255, np.int)
    mask = [[[-3, -3, 5], [-3, 0, 5], [-3, -3, 5]],
                    [[-3, 5, 5], [-3, 0, 5], [-3, -3, -3]],
                    [[5, 5, 5], [-3, 0, -3], [-3, -3, -3]],
                    [[5, 5, -3], [5, 0, -3], [-3, -3, -3]],
                    [[5, -3, -3], [5, 0, -3], [5, -3, -3]],
                    [[-3, -3, -3], [5, 0, -3], [5, 5, -3]],
                    [[-3, -3, -3], [-3, 0, -3], [5, 5, 5]],
                    [[-3, -3, -3], [-3, 0, 5], [-3, 5, 5]]]
    for y in range(1,img.shape[0]-1):
        for x in range(1,img.shape[1]-1):
            neighbors = []
            neighbors.append([img[y-1][x-1],img[y-1][x],img[y-1][x+1]])
            neighbors.append([img[y][x-1],img[y][x],img[y][x+1]])
            neighbors.append([img[y+1][x-1],img[y+1][x],img[y+1][x+1]])
            G = MaxMagnitude(neighbors, mask)

            if G > threshold:
                img_new[y][x] = 0
            else:
                img_new[y][x] = 255

    return img_new
```

✧ Robinson's Compass Operator

Same as Kirsch's Compass Operator, except the difference of masks

```
mask = [[[-1, -2, -1], [0, 0, 0], [1, 2, 1]],
                [[0, -1, -2], [1, 0, -1], [2, 1, 0]],
                [[1, 0, -1], [2, 0, -2], [1, 0, -1]],
                [[2, 1, 0], [1, 0, -1], [0, -1, -2]],
                [[1, 2, 1], [0, 0, 0], [-1, -2, -1]],
                [[0, 1, 2], [-1, 0, 1], [-2, -1, 0]],
                [[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]],
                [[-2, -1, 0], [-1, 0, 1], [0, 1, 2]]]
```

✧ Nevatia-Babu 5x5 Operator

Same as Kirsch's Compass Operator, except the difference of masks

```
mask = [[[-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]],
[[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]],
[[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]],
[[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]],
[[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]],
[[-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]]]
```

- Results

Robert's Operator(threshold=12)	Prewitt's Edge Detector(threshold=36)

	
Sobel's Edge Detector(threshold=50)	Frei and Chen's Gradient Operator(40)
	
Kirsch's Compass Operator(150)	Robinson's Compass Operator(60)
	
Nevatia-Babu 5x5 Operator(12500)