R09945020 朱彥如

Homework 9

You are to implement following edge detectors with thresholds :

(a) Robert's Operator: 12

(b) Prewitt's Edge Detector: 24

(c) Sobel's Edge Detector: 38

(d) Frei and Chen's Gradient Operator: 30

(e) Kirsch's Compass Operator: 135

(f) Robinson's Compass Operator: 43

(g) Nevatia-Babu 5x5 Operator: 12500

The requirements above are accomplished by Python with the help of matplotlib package, cv2 package, numpy package and math package. The input image is padded first for the out-of-range problem occurring when the the masks are applied on the boundary pixels.

(a) Robert's Operator

The function, Robert(), is for implementing the Robert's operator, and its inputs are a 3D (RGB layers) image and a threshold value. It applies two 2x2 masks with different directions on each pixel. The formulas below are the core calculation of the function.

$$\begin{cases} r_1 = img[l,p] - img[l+1,p+1] \\ r_2 = img[l,p] - img[l+1,p-1] \\ mag = \sqrt{r_1^2 + r_2^2} \end{cases}$$

$$robert\_img[l,p] = \begin{cases} return\ 0, if\ mag \geq threshold \\ return\ 255, otherwise \end{cases}$$

By this function, I created two processed images with two threshold values, 12 and 30.

(b) Prewitt's Edge Detector

For all the requirements needed gradient edge detector such as (b), (c) and (d). I made the function, Gradient_Edge(), whose inputs are a 3D (RGB layers) image, a threshold value, and a weight value. The formulas below are the core calculation of the function.

$$\begin{cases} w = weight \\ p_1 = (img[l+1,p+1] + w \times img[l+1,p] + img[l+1,p-1]) \\ \qquad -(img[l-1,p+1] + w \times img[l-1,p] + img[l-1,p-1]) \\ p_2 = (img[l+1,p+1] + w \times img[l,p+1] + img[l-1,p+1]) \\ \qquad -(img[l+1,p-1] + w \times img[l,p-1] + img[l-1,p-1]) \\ mag = \sqrt{p_1^2 + p_2^2} \end{cases}$$

$$Grad\_Edge\_img[l,p] = \begin{cases} return\ 0, if\ mag \geq threshold \\ return\ 255, otherwise \end{cases}$$

To create the image with Prewitt's edge detector, I use Gradient_Edge() function with the weight = 1, and the threshold value, 24.

(c) Sobel's Edge Detector
To create the image with Sobel's edge detector, I use Gradient_Edge() function with the weight = 2, and the threshold value, 38.

(d) Frei and Chen's Gradient Operator
To create the image with Frei and Chen's gradient operator, I use Gradient_Edge() function with the weight = √2, and the threshold value, 30.

(e) Kirsch's Compass Operator
For all the requirements needed a compass operator such as (e), and (f).
I made the function, Compass(), whose inputs are a 3D (RGB layers) image, a threshold value, and a weight array. The formulas below are the core calculation of the function.

$$\begin{cases} R_{mask} = \{the\ masks\ created\ by\ rotation\ of\ weight\ array\} \\ sum\ score = max_{mask \in R_{mask}} \left(\sum_{(i,j) \in mask} img[l+i, p+j] \times mask[i,j] \right) \end{cases}$$

The function uses the input array to create 8 masks by the rotation, and applied every mask to the target pixel to get the max sum score. In the implementation of the function, the judgement of an edge pixel is determined when there is a score greater than the threshold that means the max score will also greater than the threshold.

$$Compass\_img[l,p] = \begin{cases} return\ 0, if\ sum\ score \geq threshold \\ return\ 255, otherwise \end{cases}$$

To create the image with Kirsch's compass operator, I use Compass() function with the weight array = [-3,-3,5,5,5,-3,-3,-3], and the threshold value, 135.

(f) Robinson's Compass Operator
To create the image with Robinson's compass operator, I use Compass() function with the weight array = [-1,0,1,2,1,0,-1,-2], and the threshold value, 43.

(g) Nevatia-Babu 5x5 Operator
The function, Nevatia_Babu(), is for implementing the Nevatia-Babu 5x5 operator, and its inputs are a 3D (RGB layers) image, a threshold value, weight arrays list. The formulas below are the core calculation of the function.

$$\begin{cases} arr_{mask} = \{the\ masks\ in\ the\ input\ list\} \\ sum\ score = max_{mask \in arr_{mask}} \left(\sum_{(i,j) \in mask} img[l+i, p+j] \times mask[i,j] \right) \end{cases}$$

R09945020 朱彥如

In the implementation of the function, the judgement of an edge pixel is also determined when there is a score greater than the threshold that means the max score will also greater than the threshold.

$$Nevatia\_Babu\_img[l,p] = \begin{cases} return\ 0, if\ sum\ score \geq threshold \\ return\ 255, otherwise \end{cases}$$

To create the image with Nevatia-Babu 5x5 operator, I use Nevatia_Babu() function with the weight arrays list, and the threshold value, 12500.
There are 6 weight arrays in the input weight arrays list. The matrixes below are two of them and the others are created by flip and transpose of a matrix implemented by numpy.

$$\begin{bmatrix} 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 78 & -32 \\ 100 & 92 & 0 & -92 & -100 \\ 32 & -78 & -100 & -100 & -100 \\ -100 & -100 & -100 & -100 & -100 \end{bmatrix} \begin{bmatrix} 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 \\ 0 & 0 & 0 & 0 & 0 \\ -100 & -100 & -100 & -100 & -100 \\ -100 & -100 & -100 & -100 & -100 \end{bmatrix}$$

Robert's Operator: 12

Robert's Operator: 30

R09945020 朱彥如

Prewitt's Edge Detector: 24	Sobel's Edge Detector: 38
	
Frei and Chen's Gradient Operator: 30	Kirsch's Compass Operator: 135
	
Robinson's Compass Operator: 43	Nevatia-Babu 5x5 Operator: 12500