Computer Vision HW9 Report

B01902044 Steven Lin

2015-12-10

Contents

1 Source Code

This part of the report will go through some important snippets of my source code. For source code file, please check out the `hw9.py` file in the directory submitted.

1.1 Principle Code Sneak Peek

First, we will take a look at the overall structure of my code. And furthermore we will have some brief explanation for the implementation of the 7 operators in this homework.

1.1.1 Functions Overview

In this homework, we are asked to implement 7 different kinds of edge detection methods. And for every method, I created a function for it.
The following snapshot shows a list of 7 functions corresponding to one of the detection methods respectively.

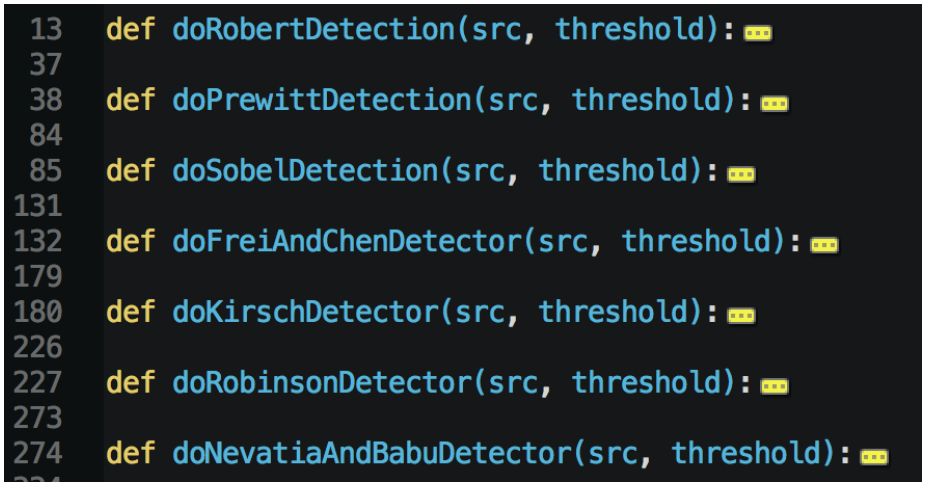

Figure 1: Essential Functions Overview

All of the functions above take 2 parameters: `src` and `threshold`. For more information about these two required parameters, please see to the description in the following list.

`src`
: The source image for the functions to detect edges.

`threshold`
: The threshold for the operator with regard to the specific edge detection method.

1.1.2 Common Gradient Magnitude

The most common type of edge detection process uses a gradient operator which calculate the gradient magnitude `g(x, y)` from the intensity values of neighboring pixels. Mathematically, it is computed this way:

$$g(x, y) \cong (\Delta x^2 + \Delta y^2)^{\frac{1}{2}}$$

and the gradient direction would thus be:

$$\theta(x, y) \cong atan(\frac{\Delta y}{\Delta x})$$

Detection methods of this type Robert's, Prewitt's, Sobel's, Frei and Chen's.

For this type of operators, I first created two empty 2D matrices with each of them representing Δx and Δy, and after the completion of computing both of the matrices, I utilize the code fragment as in the figure below to determine the final gradient value.

```
# create new image, set the value, and return it
newImage = Image.new(src.mode, src.size)
newImagePixels = newImage.load()
for i in range(src.size[0]):
    for j in range(src.size[1]):
        # if > threshold: black; otherwise, white.
        newImagePixels[i, j] = 0 if sqrt(Gx[i][j] ** 2 + Gy[i][j] ** 2) > threshold else 255
return newImage
```

Figure 2: Common Gradient Calculation

1.1.3 Maximum Magnitude Type

The other type of magnitude calculation involves multiple (usually greater than two) masks. With convolving every one of the masks respectively, we determine the final gradient value `g(x, y)` by this formula:

$$g(x, y) = \max_{i=1..N}(convolved\_value(mask_i))$$

where `N` is the number of masks defined/used this operator.

Detection methods of this type Kirsch's, Robinson's, Nevatia-Babu 5x5 Operator.

As for this type of operators, when calculating `g(x, y)` for every pixel on the source image, I first retrieve the intensity values of its neighbors that are required according to the definition of the operator. I will then have two `list` (i.e., array), one for intensity values needed, and the other for coefficient defined by the operator. The gradient value will therefore be computed this way:

$$g(x, y) = \max_{i=1..N}(compasses_i)$$
$$compasses_i = coeff_i \cdot neighbors$$

where `N` is the number of masks (i.e., compasses), `coeff` is the coefficient array, `neighbors` is the array storing intensity values of neighbors needed. The actual implementation of these equations are as shown in the following code snippet.

```
# compute the 8 compasses
for k in range(8):
    compasses[k] = sum([coefficientList[k][tmp] * neighborList[tmp] for tmp in range(9)])
# put the max of 8 compasses into gradientMap
gradientMap[i][j] = max(compasses)
```

Figure 3: Max Gradient Calculation

1.2 Parameters (Threshold)

Robert's Operator
my threshold = 30

Prewitt's Edge Detector
my threshold = 24

Sobel's Edge Detector
my threshold = 38

Frei and Chen's Gradient Operator
my threshold = 30

Kirsch's Compass Operator
my threshold = 135

Robinson's Compass Operator
my threshold = 43

Nevatia-Babu 5x5 Operator
my threshold = 12500

2 Results

All the resulted images are properly saved and submitted along with this report document. You may go check them out if you'd like to.

2.1 Robert's Operator

Figure 4: Robert's with threshold=30

2.2 Prewitt's Edge Detector

Figure 5: Prewitt's with threshold=24

2.3 Sobel's Edge Detector

Figure 6: Sobel's with threshold=38

2.4 Frei and Chen's Gradient Operator

Figure 7: Frei and Chen's with threshold=30

2.5 Kirsch's Compass Operator

Figure 8: Kirsch's with threshold=135

2.6 Robinson's Compass Operator

Figure 9: Robison's with threshold=43

2.7 Nevatia-Babu 5x5 Operator

Figure 10: Nevatia-Babu's with threshold=12500