Image and Video processing
Laboratory 3
Edge and contour detection

Prof. Touradj EBRAHIMI

Reports submission – !!! read carefully !!! Students will have to produce and submit a report two weeks after completion of each laboratory session. An electronic version of the report is submitted via Moodle's interface for uploading assignments. There will be a hard deadline for each submission. Late submissions will have to be justified and explained to responsible assistant by email. A structure of the report is explained below. In addition to the report, the source code produced should also be submitted in electronic form. The report and the source code should be submitted in **ONE ZIP** file using Moodle platform.

Structure of the reports and m-files

- the name of the final ZIP file will be: `lab_`*number-of-lab_your-surname*`.zip`
- the final ZIP file will contain
  - final m-file called `run_lab_`*number-of-lab_your-surname*`.m`
  - all created m-files and functions which are necessary to obtain the results
  - all source images and data needed for successful running of the final m-file
  - an electronic version of your report in **PDF** format
- the structure of the final m-file
  - the final m-file will contain all necessary commands and functions, by running this m-file one must get all results[1]
  - submitted m-files will be commented
  - each figure will be properly titled
  - the final m-file will be divided into cells[2] according to the example bellow
- the parts of program codes should not be included in the final report
- description of the results should be a part of the final report
- don't forget to answer all the questions in your report
- check whether your final m-file can be launched without problems - only then submit your report

[1] of course it is upon you whether you want to include everything in the final m-file, or create different functions which you will call in the final m-file

[2] those who do not know how to use the cells in Matlab to create the program code more transparent and easier to read, they should look at `http://www.mathworks.com/demos/matlab/developing-code-rapidly-with-cells-matlab-video-tutorial.html`

An example of final m-file

```
% Lab (number of lab) - your name and date

%% Exercise (number of exercise) - "Name of exercise"

% your own program code with your coments
a = imread('picture.tif');
% all figures will be properly titled
figure('name','Name Of The Figure')
imshow(a,[]);

%% Exercise (number of exercise) - "Name of exercise"

% your own program code with your coments
[output1, output2, ..., outputN] = function_name (input1, input2, ..., inputN);
```

1 Template method

Template methods try to model the shape of edges and give an approximation of the gradient. Implement a function that performs edge detection using the following *templates*:

- Sobel
$$g_1 = \frac{1}{4}\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \textit{ and } \qquad g_2 = \frac{1}{4}\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$
- Prewitt
$$g_1 = \frac{1}{3}\begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} \textit{ and } \qquad g_2 = \frac{1}{3}\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$
- Roberts
$$g_1 = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \textit{ and } \qquad g_2 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Where the template and the threshold value are the function parameters. Additionally, the function must allow different types of magnitude approximation :

- L_2-norm $y(k,l) = \sqrt{(x**g_1(k,l))^2 + (x**g_2(k,l))^2}$
- L_1-norm $y(k,l) = |x**g_1(k,l)| + |x**g_2(k,l)|$

2 Compass operator

Compass operators approximate the gradient in a number of given directions, typically eight. Then,the gradient magnitude is taken as the maximum absolute value among all directions. Finally, the gradient magnitude is used to classify the edges. Implement a function that performs edge detection using the Kirsch operator. The kirsch masks are given by

$$k_0 = \begin{pmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{pmatrix}$$

,

and $k_1, \ldots, k_7$, where k_i is obtained by rotating k_{i-1} by 45 degrees counter-clockwise.

3 Laplace operator

Edge detectors based on the Laplace operator look for parts of the image where the second derivative (i.e. the Laplacian) changes its sign (goes through zero). These correspond to extrema of the gradient. Since the Laplacian is very sensitive to noise, the image is pre-filtered with a Gaussian filter to reduce the noise before applying the Laplacian. Instead of using two filters, we will combine both, Laplacian and Gaussian, to compose one filter, Laplacian of Gaussian (LOG). We apply this filter to the image and detect points where the Laplacian goes through zero.

Write a function that implements the LOG filter. This filter can be obtained by the Matlab function `fspecial`. The LOG filter size of $2\lceil 3\sigma \rceil + 1$ is typically used because the Gaussian distribution takes negligible values after 3σ. Typical values for σ are around 2 or 3.

The detection of points where the Laplacian goes through zero can be done by looking at two neighboring pixels (for example a pixel and its upper neighbor). For instance, if they are of opposite signs and their difference is greater than a certain threshold, the negative pixel is considered as a part of the edge. This operation is applied to both, lines and columns.

4 Frei-Chen method

This method uses a slightly different approach than the previous methods. More precisely, the previous methods try to find maxima of the first derivative, or detect where the second derivative goes through zero. Instead, the Frei-Chen method projects the image into a nine-dimensional space. Two dimensions capture structures representing the edges while the others capture other types of structures (lines and uniform values). We obtain the edge's intensity information by calculating the angle between the nine-dimensional vector corresponding to each pixel and its projection on the two dimensions representing the edge structures.

First, the image is filtered with the following masks:

$$f_0 = \frac{1}{2\sqrt{2}} \begin{pmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{pmatrix} \quad f_1 = \frac{1}{2\sqrt{2}} \begin{pmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{pmatrix} \quad f_2 = \frac{1}{2\sqrt{2}} \begin{pmatrix} 0 & -1 & \sqrt{2} \\ 1 & 0 & -1 \\ -\sqrt{2} & 1 & 0 \end{pmatrix}$$

$$f_3 = \frac{1}{2\sqrt{2}} \begin{pmatrix} \sqrt{2} & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -\sqrt{2} \end{pmatrix} \quad f_4 = \frac{1}{2} \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \quad f_5 = \frac{1}{2} \begin{pmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{pmatrix}$$

$$f_6 = \frac{1}{6} \begin{pmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{pmatrix} \quad f_7 = \frac{1}{6} \begin{pmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{pmatrix} \quad f_8 = \frac{1}{3} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

We obtain the image y_n for each mask f_n. The required angle at each pixel is given by

$$\alpha(k,l) = \arccos \sqrt{m(k,l)/s(k,l)},$$

where

$$m(k,l) = \sum_{n=0}^{1} (y_n(k,l))^2, \qquad s(k,l) = \sum_{n=0}^{8} (y_n(k,l))^2,$$

and k, l are the pixel coordinates.

Given that we will threshold this angle, we are only interested in its magnitude. Thus, we can threshold the $\cos(\alpha)$ values directly(i.e. there is no need to apply the arccos function).

Implement a function that performs edge detection using this method.

5 Evaluation

1. Apply the methods above to `lena.png`, `road.png` and `rice.png`. Compare their efficiency and complexity.

2. Repeat the operation to `lena.png` or `road.png` disturbed by a Gaussian noise with standard deviations 5, 11 et 25 (with respect to the dynamic range $[0, 255]$). We can use the Matlab function `imnoise`. What are your conclusions?