LAB 11

Fall 2010, BESE- 13 & 14

Image Processing with MATLAB

Objective

Today's lab covers a well-known technique of Local Adaptive Thresholding for Segmentation named as Niblack Algorithm. Also it explores a new function to implement the edge detection on gray-scale images in MALTAB. By the end of this lab you should be able to work with edge detectors, such as Canny, Laplacian, and Sobel, and Niblack algorithm for image Segmentation.

Instructions for LAB Report Submission

You are required to submit the lab assignment in a document (**.docx or .doc**) format. This document **MUST** include: *1)* piece of code (MATLAB script) you programmed to implement the assigned task and *2)* snapshots of your work to show results.

Name your reports as: ***Lab#_Rank_YourFullName***

- '#' replaces the lab number
- '*Rank'* replaces Maj/Capt/TC/NC/PC
- '*YourFullName*' replaces your complete name.

Tasks for Today

1. Edge Detection – Sobel, Laplacian and Canny

Edge detection, as discussed in Lab-08, covered tutorial about implementing edge detectors using **fspecial** and **imfilter** functions. This tutorial is about implementing the edge detectors using **edge** function in MATLAB. The difference between the two approaches is that former can be implemented on RGB images while **edge** function works only with grayscale images.

Sobel Edge Detector…

The Sobel method finds edges using the Sobel approximation to the derivative. It returns edges at those points where the gradient of image is maximum.

Using **edge** function in MATLAB, Sobel method can be applied as follows:

```
>> BW = edge(I,'sobel', thresh, direction);
```

Where `I` is the image on which sobel filter is to be applied. `thresh` specifies the threshold for the Sobel method. **edge** function ignores all the edges that are not stronger than `thresh`. If `thresh` is not specified or if thresh is empty (`[]`), **edge** chooses the value automatically. And `direction` is a constant string specifying whether to look for `'horizontal'` or `'vertical'` edges or `'both'` (the default).

Zero Corssing of 2nd Order Derivative – Laplacian…

The zero-cross method finds edges by looking for zero crossings after filtering image with a specified filter. Using **edge** function in MATLAB, Laplacian method can be applied as follows:

```
>> BW = edge(I, 'zerocross', thresh, h);
```

Where `I` is the image to be filtered for zero-crossing using filter `h`; `thresh` specifies the sensitivity threshold which is if empty (`[]`) **edge** automatically chooses the sensitivity threshold. If you specify a threshold of 0, the output image has closed contours, because it includes all the zero crossings in the input image.

Example:

```
>> img = imread('moon.tif');
>> h = [0 1 0; 1 -4 1; 0 1 0];
>> BW = edge(img, 'zerocross', [], h);
```

Canny Edge Detector …

The Canny method finds edges by looking for local maxima of the gradient of image. The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be fooled by noise, and more likely to detect true weak edges. Using **edge** function in MATLAB, Canny method can be applied as follows:

```
>> BW = edge(I, 'canny', thresh);
```

Where `I` is the image to be filtered; `thresh` specifies the sensitivity threshold which is a *'two-element vector'* in which the first element is the low threshold, and the second element is the high threshold. If you specify a scalar for thresh, this value is used for the high threshold and 0.4*thresh is used for the low threshold. For empty vector (`[]`) i.e. no threshold, **edge** automatically chooses low and high values.

Exercise 1:

Read an image 'moon.tif'. Write a program to apply Canny and Sobel edge detectors separately.

- State your findings about the results from Sobel and Canny edge detectors. Which one is robust to detect weak edges while going unresponsive to noise?

2. Local Adaptive Thresholding – Niblack Algorithm

Niblack algorithm is one of the well-known algorithms for segmenting out the image information based on Local Adaptive Threshold. Mathematically, it is defined as:

$$T(x, y) = m(x, y) + k * s(x, y)$$

Where, **m(x, y)** and **s(x, y)** are the average of a local area and standard deviation values, respectively. The size of the neighborhood should be small enough to preserve local details, but at the same time large enough to suppress noise.
The **k**, Niblack's constant, is used to adjust how much of the total object boundary is taken as a part of the given object.

Implementing Niblack algorithm in MATLAB…
[Step – 01]: Copy the original image into a new image such that the boundaries (top, bottom, left and right) of original image are replicated.
(Hint: For 5x5 *filter, each side of original image is replicated into* **x = (n-1)/2** *rows and columns of the new image at corresponding side; where n is size of filter i.e.* 5 *in this particular case)*.

[Step – 02]: For each pixel `img(i, j)` compute the Threshold as follows:

a. Copy elements under the current window into a separate matrix **W** as follows:
```
W = img(i-x: i+x, j-x: j+x);
```

b. Compute mean of the pixels values in current window into **mean_W**:
```
mean_W = mean(mean(W));
```

c. Compute the standard deviation of pixel values under current window as follows:
```
stdd_W = sqrt(mean(mean((w – mean_w).^2)));
```

d. Compute threshold for the current pixel `img(i, j)` as follows:
```
thresh_w = mean_w + k * stdd_w;
```
where k is Niblack's constant which specifies how much of the total object boundary will be taken as a part of the object.

e. Apply the results on to current pixel `img(i, j)` as follows:

```
if img(i, j) > thresh_w
     new_img(i, j) = 0;
else
     new_img(i, j) = 1;
end
```

Exercise 2:

Read an image 'rice.png'. Write a function named 'myNiblack' to implement Niblack algorithm for segmentation based on local adaptive threshold. Show the results of your implementation.

[HINT]: Follow aforementioned steps as an aid to this exercise.