LAB 8

Fall 2010, BESE- 13 & 14

Digital Image Processing with MATLAB

Objective

The purpose of today's lab is to introduce you to the process of Filtering. This lab spotlights the built-in MATLAB IPT functions for *Smoothing* and *Sharpening* filters. Also, it gives a brief overview for manually coding/implementing these filters in MATLAB on mathematical grounds. By the end of this lab you should be able to use Smoothing/Sharpening filters and code them at your own.

Instructions for LAB Report Submission

For this LAB (i.e. Lab-08) 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's LAB

1. Filtering

Filtering is the process of applying a transformation matrix *h(u, v)* over some input intensity *f(x, y)* such that we have transformed intensity *g(x, y)* while considering the effect of 4- or 8- neighbors of *f(x, y)*.

$$g(x, y) = h(u, v) * f(x, y)$$

where the transformation matrix *h(u, v)* is called as kernel, mask, or **filter**. *g(x, y)* is the resultant intensity obtained after *h(u, v)* is **convolved** with *f(x, y)*.

Applying Filters in MATLAB

In MATLAB applying a filter to an image is a 2-step procedure:

1. Create a filter using **`fltr = fspecial(fltr_typ, fltr_siz)`** function. Where `fltr_typ` specifies type of a filter e.g. 'average', 'laplacian', 'sobel' etc. `fltr_siz` specifies the filter size and its value can be a scalar (e.g. 3 mean 3x3 filter) or it can be a vector (e.g. [5 5] means 5x5 filter).
   NOTE: in case of 'laplacian' and 'sobel' filters `fltr_siz` is not required.
2. Apply filter, as created in step-1, to the desired image using **`fltrd_img = imfilter(img, fltr, options)`** function. Where `img` is a 2D matrix to which we want to apply the filter. `fltr` is the filter matrix created using **`fspecial`**. `option` can be 'replicate' (i.e. replicate padding) or '0' (zero padding).

1.1 Smoothing Filters:

Smoothing filters blur out the image or reduces the effect of noise based upon 4- or 8- neighbors. Examples include: **Average** filter and **Median** filter.

a. **Average Filter:**

   Example:

```
>> img = imread('cameraman.tif');
>> fltr = fspecial('average', [3 3]);
>> fltrd_img = imfilter(img, fltr, 'replicate');
>> imshow(img), figure, imshow(fltrd_img)
```

b. **Median Filter:** Median filter is a statistical filter and cannot be created/applied using **`fspecial`** and **`imfilter`** functions. So, MATLAB IPT defines a special function, named **`medfilt2`**, to implement Median filter.

   `fltrd_img = medfilt2(img, fltr_siz)`

   where `img` is the image and `fltr_siz` is a vector defining filter size e.g. [3 3]. **`medfilt2`** performs filtering by applying zero padding.

   Example:

```
>> img = imread('cameraman.tif');
>> fltrd_img = medfilt2(img, [3 3]);
>> imshow(img), figure, imshow(fltrd_img)
```

Exercise 1:

Read an image ‘coins.png’. Apply average and median filters of size 5x5 to individually identify the differences b/w their results.

1.2 Sharpening Filters:

Sharpening filters are analogous to derivative and the response of a derivate at some *f(x, y)* is proportional to discontinuity (e.g. noise or sudden change in gray-intensity). Thus derivates (most particularly 2^{nd} order derivate) are used to sharpen the boundaries of objects/regions and/or finding the edges (change in shades). Examples include: **Laplacian** filter and **Sobel** filter.

a. **Laplacian Filter – 2^{nd} Order Derivative of Image**

Example:

```
>> img = imread(‘cameraman.tif’);
>> fltr = fspecial(‘laplacian’);
>> fltrd_img = imfilter(img, fltr, ‘replicate’);
>> imshow(img), figure, imshow(fltrd_img)
```

b. **Sobel Filter:**

Example:

```
% extract horizontal edges
>> img = imread(‘cameraman.tif’);
>> fltr = fspecial(‘sobel’);
>> fltrd_img = imfilter(img, fltr, ‘replicate’);
>> imshow(img), figure, imshow(fltrd_img)

% extract vertical edges – transpose of fltr i.e. fltr’
>> img = imread(‘cameraman.tif’);
>> fltr = fspecial(‘sobel’);
>> fltrd_img = imfilter(img, fltr’, ‘replicate’);
>> imshow(img), figure, imshow(fltrd_img)
```

Exercise 2:

Read an image 'moon.tif'. Write a function named 'mylaplacian' to MANUALLY code/implement 2nd order derivate of above read image in order to extract horizontal and vertical edges, collectively. Also, compare your results with 'Sobel' filter and state your findings.

[HINT]:

Vertical Edges:

```
g(x, y) = f(x + 1, y) + f(x – 1, y) – 2f(x, y)
```

Horizontal Edges:

```
g(x, y) = f(x , y + 1) + f(x, y - 1) – 2f(x, y)
```