

Computer Assisted Image Analysis II

Spring 2015

Excercise 1: Filter Design

Local filtering operations are some of the most useful and important tools in all computerized image analysis tasks. The aim of filtering is to enhance information in an image. The information of interest may vary from case to case, e.g., edges or lines in certain directions may be extracted, the image may be sharpened, or noise may be suppressed. There exist many well-known general filters that perform these operations, but there is also a need to design special purpose filters and edit existing ones. The aim of the exercise is to try various predefined filters, implementing your own filters and tune the filters for specific applications.

*Every part of the exercise has some instructions, and sometimes also some questions. Read these **before** actually performing the part. Answer the questions explaining **why** you got a particular result, rather than **how**. If you get stuck, do not spend too much time on a particular problem, it is better to continue and ask us later on.*

Formality

- You need an account for the computer systems at the Dept. of Information Technology or the Division of Visual Information and Interaction.
- You should work together in groups of two people. It may be advantageous to have someone to discuss issues with.
- You are requested to prepare a written report in pdf format (one per group) of the exercise answering all questions including explanatory illustrations. The report is not required if you are able to finish all the exercises and get them graded by the instructor during the lab session.
- The report is handed in via the Student Portal under 'File areas' → 'Lab reports'. Make sure that you belong to a group before starting the lab. You join a group in the Student Portal by going to 'Group divisions' → 'Lab/Project groups'.
- Status of your reports will be found at the Student portal under 'Progress' → 'Progress, mandatory tasks'.
- **Deadline: Before the next lab**

Getting started

In a lab at the Dept. of Information Technology running WINDOWS XP: Log on to one of the workstations. MATLAB is found under G:\program\MatlabR2013b\

Download the lab material from the Student portal, 'File areas' → 'Lab material'

1 Filtering

Gaussian filters

One can use the `conv2` function in MATLAB to implement the convolution. To generate the Gaussian kernels, one can use the `fspecial` function from MATLAB. Use the `help` command followed by function name in MATLAB to read more about these functions. Read the image 'van.tif' using `imread` function in MATLAB. Syntax for creating a Gaussian kernel:

```
h = fspecial('gaussian', [a b], sigma);
```

where `a` and `b` defines the size of the Gaussian kernel and `sigma` defines the standard deviation. However, the Gaussian filter is separable and by utilizing this some computing time can be saved. To use `conv2` with two 1D filters type:

```
result = conv2(hx,hy,image);
```

1. *How do you choose the size of the filter kernel with regards to the standard deviation (sigma) of the Gaussian distribution?*
2. *How much time do you save when using separated Gaussian filters with sigma = 1 on 'van.tif'? Please use conv2 to compute the convolution. HINT: In MATLAB you can use the tic, toc function pair to measure the CPU time.*

Difference of Gaussians

Difference of Gaussians is a grayscale image enhancement algorithm that involves the subtraction of one blurred version of an original image from another, less blurred version of the original. The blurred images are obtained by convolving the original grayscale image with Gaussian kernels having different standard deviations. This can be written mathematically by:

$$DoG = f \otimes h_1 - f \otimes h_2, \quad (1)$$

where h_1 and h_2 are two Gaussian kernels with different standard deviations, and f represents the original image, while \otimes represents the convolution operator. In the course book DoG is discussed in chapter 5.3.3.

3. *Implement equation 1 with f being the image 'van.tif'. How does changing the standard deviation (sigma) affect the result?*
4. *Does it make any difference if we switch position of h_1 and h_2 ? If yes, explain the difference.*
5. *As you are aware of the distributive property of the convolution, i.e.*

$$f_1 \otimes (f_2 + f_3) = f_1 \otimes f_2 + f_1 \otimes f_3 \quad (2)$$

Using the same law, one can write difference of Gaussian as:

$$DoG = f \otimes h_1 - f \otimes h_2 \quad (3)$$

$$= f \otimes (h_1 - h_2) \quad (4)$$

Implement equation 4 and apply it on the image and compare your results with Q7 for the same values of sigma. Is there any benefit (computationally) to using equation 4 instead of equation 1?

Noise reduction in 3D

Reducing noise is one very common image processing task. Before designing a noise suppression filter, one needs to have some knowledge about noise type and its properties (like mean, standard deviation etc.).

In this exercise you will be working with an image of the electron density in a hydrogen molecule, H_2 . We will begin by looking at one 2D slice of the 3D image. Run `noise_demo_2D.m` for an example of noise in this 2D slice. In the example median filtering is used to remove 'salt & pepper' noise and mean filtering is used to remove Gaussian noise.

6. *Why are these filters suitable for removing these types of noise?*

Viewing a 3D image is not as trivial as viewing a 2D image. One can render each voxel as a semitransparent cube with each voxel value mapped to a color, this is referred to as volume rendering, alternatively render an isosurface by setting a threshold and creating a surface at this threshold. A convenient way of adding noise to an image is to use the `imnoise` function in MATLAB. Type `help` to see how to specify Gaussian or 'salt & pepper' noise.

Load the 3D image by typing `v = readVTK('hydrogen.vtk');` the image is stored as a VTK image which is one of many ways of storing 3D images. Use the `volrender` function supplied to you for volume rendering, the 'Rotate 3D' tool in the figure toolbar is used to rotate the image.

Create two 3D images, one having Gaussian noise (mean=0, and sigma=0.001) and the other having 'salt & pepper' noise with density 0.01.

7. *Try to remove the noise using the median and the mean filters (use `ordfilt3D` function for 3D median filtering and `imfilter` function for mean filtering with filter size $3 \times 3 \times 3$). Which method will achieve the best result and why?*

Another way of reducing noise is to, if possible, acquire several images (with random noise) of the same scene and then taking the mean or median value of each voxel among the set of images. You can create such a set of images by creating, let's say, 27 images with Gaussian noise added. An example code for achieving this could be:

```
array = cell(27,1);
for i = 1 : 27
    array{i} = imnoise(image,"options");
end
```

Use the same settings for the Gaussian noise as used above.

8. *Take the mean value of each voxel among the 27 images. Compare this result with the mean filtered image in the previous question.*

2 Gabor filtering

Please run the `gabor_filter_example.m`. In this example you can mark positions in the Fourier space to generate a Gabor filter. The size of the Gabor filter can be changed in the m-file. Go through the code line by line so you understand what is actually happening at each step.

9. *How many objects or grid patterns do you find? Show the resulting images.*

10. *In the `gabor_filter_example.m` the image is filtered with the sine and cosine part separately. Take a look at the two resulting images, `b1` and `b2`, and describe how they differ. Play with the `sigma` parameter and look at the filter that is generated.*

3 Canny edge detector and Hough transform

Detecting edges

The original Canny edge detector was presented in 1983 by John Canny at MIT. For a concise description, study Chapter 5.3.5 in [?]. It is optimal for step edges corrupted by white noise. The method is based on the following three criteria:

Detection There should be a high probability of detecting *true* edge pixels (high sensitivity) and a low probability of detecting *false* edge pixels (high specificity).

Localization The pixels detected as edge pixels should be as close as possible to the *true* edge.

One response If multiple responses are detected, then only one is true, the others are false.

To accomplish these criteria a “hybrid” combination of linear and non-linear approaches is performed:

- Gaussian gradient computation
- non-maximal suppression
- hysteresis thresholding

One can use `edge` function in MATLAB instead of implementing your own. Use the `help` function to read more about the `edge` function. The syntax is `BW = edge(I,'canny',thresh,sigma)`, where `thresh` is a 1x2 matrix with high and low threshold. `sigma` is the standard deviation of the Gaussian filter. Open the image `'coins.png'` and display it.

11. *What is your opinion on what an edge is in this particular image?*
12. *Try using different parameter values for the Canny edge detection algorithm. After trying a number of them decide which parameter settings are best for finding the edges between the coins and the background. Show the detected edges in white on top of the original image.*

4 Implementing circular Hough transform

Your final task is to find the circular objects in the edge map computed in the previous task. You should do this by implementing a Hough transform that parametrizes circles. Please use a three dimensional parameter space with x and y position as well as radius. Make sure you understand the relationship between the image space and the parameter space. For some help see Fig. 1.

You have access to a function that will draw a circle in an image, named `generate_circle`. The rest is up to you. HINT: Try to keep the radius interval small and use the fact that the output from the Canny method is binary.

13. *Explain the relationship between the image space and the parameter space.*
14. *Using pseudocode, explain how you implemented the Hough transform.*
15. *Find the coins in the Hough parameter space. Illustrate your result by drawing the borders of the detected circles from the Hough space as well as plotting the size on top of the gray scale image, see Fig. 2.*

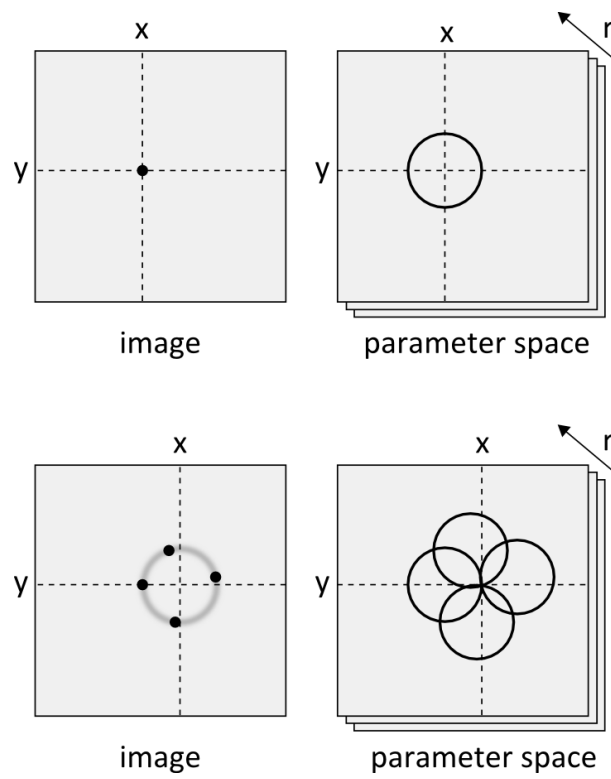


Figure 1: The relationship between the image space and the parameter space.



Figure 2: Example result