Computer Assisted Image Analysis II
Spring 2018

Excercise 1: Image restoration, Object detection

Image pre-processing methods, aiming at enhancing relevant information in the images (such as edges, or characteristic details), while suppressing non-relevant or non-desired details (such as noise), may, or may not, utilize a priori knowledge about the image formation/degradation model. Local filtering operations do most often not rely on the knowledge about the degradation. Image restoration methods, on the other hand, utilize the degradation model to estimate the original image from the observed (degraded) one. This usually requires estimation of some parameters of the degradation model, as well, which is in general a rather difficult task.

The aim of this exercise is to try to use some predefined filters, to implement your own and tune the filters for specific applications. Further, you will try to estimate and measure noise in the images, and to restore a degraded image by utilizing several techniques. Finally, you will try to detect some object in the images by suitable filtering.

Every part of the exercise has some instructions, and sometimes also some questions. Read these **before** *actually performing the task(s) of that part. When answering the questions, explain* **why** *you got a particular result, rather than* **how**. *If you get stuck, do ask us!*

Formality

- You need an account for the computer systems at the Dept. of Information Technology.
- 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 of the exercise answering all questions and including explanatory illustrations. Only one report needs to be turned in per group, with names of group members mentioned in it.
- Try to spend reasonable time on the task(s) in the beginning and move forward, more important task/tasks often come later and will often require more time and effort.
- The report is handed in via the Student Portal under 'File areas' $\rightarrow$ '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' $\rightarrow$ 'Lab/Project groups'.
- Status of your reports will be found at the Student portal under 'Progress' $\rightarrow$ 'Progress, mandatory tasks'.
- **Deadline: 30, Jan 2018**

Getting started

In a lab at the Dept. of Information Technology running Windows: Log on to one of the workstations. Matlab is found under `ITC-Application Explorer(Zenworks)`, use the latest version available.

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 perform convolution. To generate Gaussian kernels, one can use the `fspecial` function from Matlab. Use the `help` or `doc` 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 fact 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. Discuss this in terms of estimated number of operations. One way to verify this analysis is to measure the CPU time spent on both these approaches.*** Hint: ***In*** Matlab ***you can use the*** `tic`, `toc` ***function pair to measure the CPU time.***

Difference of Gaussians

Difference of Gaussians is a gray-scale image enhancement approach 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 gray-scale image with Gaussian kernels having different standard deviations. This can be written mathematically as:

$$DoG(f) = f \otimes h_1 - f \otimes h_2, \tag{1}$$

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.***

$$f_1 \otimes (f_2 + f_3) = f_1 \otimes f_2 + f_1 \otimes f_3\,. \tag{2}$$

Using that property, one can write difference of Gaussian as:

$$DoG(f) = f \otimes h_1 - f \otimes h_2 \quad (3)$$
$$= f \otimes (h_1 - h_2) \quad (4)$$

Implement equation 4 and apply it on the image and compare your results with equation 1 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 the 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 for Matlab 2011 sigma=0.001 is recommended. Matlab 2013 and later may require sigma=1.0×10^{-5}) 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*** $3\times3\times3$***). 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, lets 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 Noise modeling and image restoration

Image restoration relies on knowledge about the degradation model – we need to know (estimate) the type and amount of blur and noise in the image to chose/design an appropriate restoration method. It is also important to be able to measure the quality of a restored image (restoration is an objective process).

Start with adding different types and different levels of noise to a noise-free (original) image. Use the image 'cameraman.tif' and further experiment with the `imnoise` function in Matlab. Add 4 different levels of each of the Gaussian and 'salt & pepper'noise to the image. Analyze the parameters used by `imnoise` to specify the amount of noise added. Estimate quality of the degraded images by measuring PSNR and SSIM (yes, functions exist in Matlab) and plot the measured quality values as functions of the level of degradation applied.

9. ***Do you notice any difference in how image quality is quantified by PSNR and SSIM? How does that correspond to your visual assessment of the images? Include the plots and selected images to support your arguments!***

Now try to estimate the noise parameters from the images, and compare them with the known parameters you used for degradation. Use the 'cameraman.tif' image degraded by Gaussian noise with sigma=0.05. Denote it by 'cameraman noisy.tif' Specify a suitable homogeneous region by Matlab function `roipoly`. The provided function `histroi` can be used to compute the histogram of the selected region, and the function `statmoments` (also included in the Lab 1 material) can provide estimated parameters of the noise distribution. Use `help` to learn how to use these functions.

10. ***Explain the main idea used for the noise estimation. How do the estimated parameters correspond to the correct values you used to degrade the image? Can you improve the result if you use more than one region to estimate the noise and take the average of the parameters estimated per region?***

Image restoration by regularized energy minimization

Calculus of variations is a powerful tool for minimizing (or maximizing) functionals, which are mappings from a set of functions to the real numbers. Seeing a 2D-image as a 2D-function we can use tools from this branch of mathematics to help us perform image restoration. We can define a suitable objective function (a.k.a. energy function) and then find the image which optimizes the function and best fulfills our objective. This can be used to turn our restoration problem into an energy minimization problem – *for all possible images X, find the one ($\hat{X}$) which has minimal energy*, or in maths-language:

$$\hat{X} = \arg\min E(X) \tag{5}$$

Of crucial importance is of course to formulate a good energy function E, so that the found minimizer $\hat{X}$ is actually (close to) the image we wish to get.

Denosing

A "classic" way to denoise an image in this way is Total variation denoising, a.k.a. ROF denoising (from the Authors of the pioneering paper [1].) In the lab material, in the directory `NoiseModeling and ImageRestoration`, you can find an implementation of TV-denoising `EM_DenoiseTV.m`. Read the `help` and try the *Example* given in the help text.

11. ***What is the role of the parameter mu? Can you manage to tune it better than the value given in the example?***
12. ***Do the two performance measures (PSNR and SSIM) agree?***
13. ***How does the parameter mu relate to the level of image noise that we wish to remove?***

Deblurring (combined with denoising)

Regularized energy minimization can be used for solving many problems. Image blur from poor focus or camera motion can be modelled as a convolution of the image with a point spread function (PSF). Adding some noise η, we have the following image formation model:

$$S = I \otimes PSF + \eta \tag{6}$$

In this exercise, generate a synthetically blurred version of 'cameraman.tif' by convolving (filtering) it with a Gaussian PSF, then add Gaussian noise to create a blurred and noisy image 'cameraman noisy blurred.tif'. The code `EM_DeblurTV.m` is similar as `EM_DenoiseTV.m` but taking into consideration the PSF as well. Read the `help` and test-run the *Example* provided there.

Inverse filtering is another way to restore a degraded image. Try to apply iy on 'cameraman noisy blurred.tif' by using Matlab function `deconvwnr`. If you assume that NSFR (noise-to-signal ratio) is zero, the function performs direct inverse filtering. An improved result can be obtained if this parameter is set to a value different from 0 - then Wiener filtering is applied. Try to tune NSFR and observe its effect (you can get a hint how to do that from `help`).

14. ***What is the main obstacle for successful image restoration by inverse filtering?***
15. ***Compare the image quality measures (PSNR and SSIM) of the restored 'cameraman noisy blurred.tif' obtained by (1) Wiener filter and (2) energy minimization. Can you tune the Wiener filter to outperform energy minimization approach?***
16. ***What are, in your opinion advantages and disadvantages of the two compared restoration approaches?***

3 Template matching in frequency domain

Template matching is a technique used to find a location in an image where a template - smaller image, image patch - "fits best". It is often performed by computing, at different positions, correlation between the template and the image, and detecting the position of its maximal value. Template matching can be performed in frequency domain as well. In that case, `Phase Correlation` is utilized. Use the files in the `TemplateMatching` directory.

17. ***Find out how phase correlation is defined and explain how it works. What are the advantages if template matching is performed in frequency domain instead of doing it in spatial domain?***
18. ***Try to implement template matching based on phase-correlation.***
19. ***Can you detect the location of the*** `template1.png` ***in the image*** `search.png`***?***
20. ***Can you use the same approach for locating*** `template2.png` ***in*** `search.png`***? Why?***

4 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 [2]. 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.

21. ***What is your opinion on what an edge is in this particular image?*** Hint: ***Define a basic notion first and try extending it to variations such as scaling; inclusion of an edge, should it be part of an object or background; intensity gradation etc.***

22. ***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.***

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.

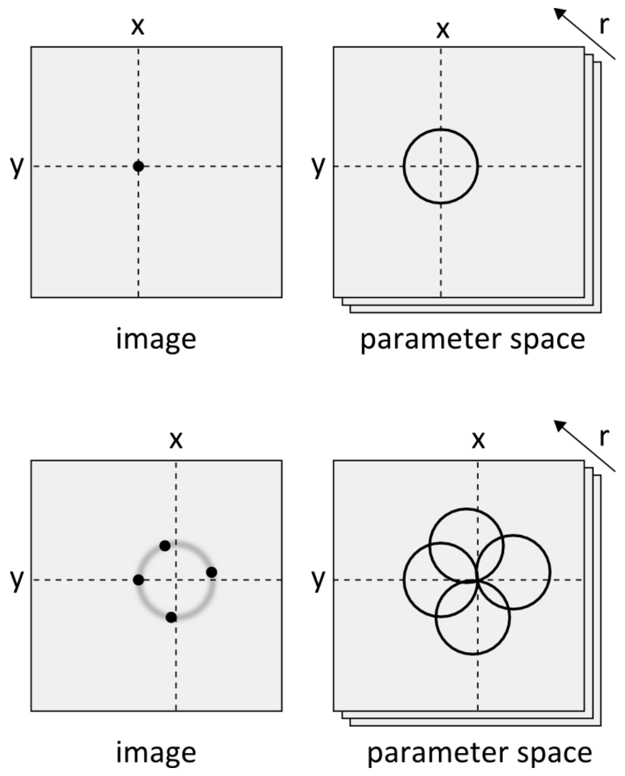

Figure 1: The relationship between the image space and the parameter space.

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. Do not use Matlab function imfindcircles.

23. ***Explain the relationship between the image space and the parameter space.***

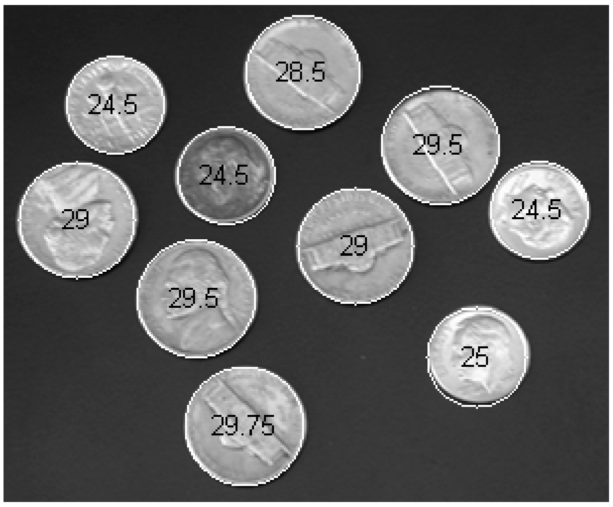

Figure 2: Example result

24. ***Using pseudocode, explain how you implemented the Hough transform.***
25. ***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.***

References

[1] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.

[2] Milan Sonka, Vaclav Havlac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision.* Thomson Learning, 3rd edition, 2008.