Introduction to Image Analysis

HT 2024

Computer Exercise 2:
Filtering and Restoration

Image preprocessing 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 most often do 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 encourage you to explore some predefined local filters, both in spatial and in frequency domain, as well as to implement your own filters, and to tune them to best fit specific applications. Further, you will estimate noise in the images, and restore a degraded image by utilizing several techniques. Finally, you will apply suitable filtering to detect some object in the images by template matching.

Every part of the exercise includes instructions that will guide you through the tasks, some additional explanations and pointers to further reading, and finally some concrete tasks you should complete and questions you should answer. Read the instructions and the complete list of questions for each exercise **before** *actually performing the tasks. When answering, focus on understanding and explaining* **why**, *rather than* **how**, *you got a particular result. If you get stuck, do not hesitate to contact the Teaching Assistant, preferably during the scheduled lab session. Try to prepare for the scheduled session, to get the most of it!*

Formalities

- Take a look at the 'Introduction to the Lab work', available in Studium, under 'Modules' → 'Labs - general', to recall what to focus on when doing lab exercises.
- **Collaborate within your team!** Start with assigning yourself into one of the teams for this exercise, under People → Computer Exercise 2. Initiate communication as soon as possible!
- **Prepare a lab report** in pdf format – one per group. Include answers to all the questions, include explanatory illustrations and, when suitable, your (pseudo-)code. Your report should follow the template provided in Studium, under 'Modules' → 'Labs - general' → 'Lab report template'.
- Please, hand in your report via Studium. If you submit the report by the indicated deadline, the TA will take a look at it, give feedback, and grade it rather quickly (within a week). Assessment of late submissions may take much longer time.
- Follow the status of your reports in Studium, under 'Grades'.

- **Deadline: October 4, 2024**

Getting started

Download the material from Studium, 'Modules' → 'Lab assignment 2' → 'Assignment Lab 2'.

1 Filtering

1.1 Gaussian filters

You have implemented convolution by mean filter in Computer Exercise 1. It is, however, possible (and often more convenient) to use the `conv2` function in MATLAB to perform convolution. Read the image 'van.tif' using MATLAB's `imread` function. Start by generating a Gaussian kernel, using the `fspecial` function from MATLAB. To read and learn more about the existing functions in MATLAB, type the `help` or `doc` command followed by the function name. Syntax for creating a Gaussian kernel is

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

where `a` and `b` define the size of the spatial filter and `sigma` defines the standard deviation of the Gaussian function. However, the Gaussian filter is separable and by utilizing this fact some computing time can be saved. To determine if a (2D) filter is separable, and to get its decomposition to two 1D filters, you can use MATLAB's function

```
[S, HCOL, HROW] = isfilterseparable(H).
```

To apply `conv2` with two 1D filters, use:

```
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? Are the coefficients in the generated Gaussian filter the same as the values of a continuous Gaussian function if sampled at the same discrete spatial locations? If not, how/why do they differ?***

2. ***How much time do you save when using separated Gaussian filters on 'van.tif' image? Use*** `conv2` ***to compute the convolution and explore different sizes of Gaussian filter, e.g., σ taking values*** $1, 7, 27$***.***
   (Note: When filtering images, you may also consider the 'specialized' functions `imfilter(im, h)` ***and*** `imgaussfilt(im, sigma)`***.)***

   Discuss this in terms of the estimated number of operations. One way to verify such analysis is to measure the CPU time spent on each of these approaches (direct convolution with the 2D kernel, and convolution with the two 1D filters after the separation). As the execution speed of such computations depends on

your available computational resources at that very moment, the results can fluctuate when you repeat the same calculation twice. To get a better estimate, run this empirical time analysis 100 times and report the mean and standard deviation of the run time.

Recall that you can use the* MATLAB*'s* `tic` *and* `toc` *function pair to measure the CPU time.

1.2 Difference of Gaussians - DoG filter

Laplace operator approximates the second derivatives and is used to detect edges in an image (as the positions of zero-crossing of the second derivative). Laplacian is, however, very sensitive to noise and is therefore often applied after a smoothing filter; Gaussian filter is a popular choice. The resulting Laplacian of Gaussian - LoG filter - has several good properties, but it is not straightforward to compute it by using a sequence of 1D filters (i.e, separability). At the same time, its size grows quickly with increase of σ, which slows down the computation. A good approximation of LoG is the *difference of Gaussians* - DoG filter. The filter is a difference of two Gaussian filters with suitably (and substantially) different σ values. This can be formally expressed as:

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

where h_1 and h_2 are two Gaussian kernels with different standard deviations, f represents the original image, and $*$ represents the convolution operator. DoG is a good approximation of LoG if the ratio between the two standard deviations of the Gaussian filters is approximately 1.6. Much bigger ratios are often used for other applications of the DoG filter.

3. ***Implement the equation (1) and apply the DoG filter to the 'van.tif' image. How does the choice the two values of σ affect the result? Try some combinations, such that the ratio of the two σ values varies.***

4. ***The distributivity of convolution w.r.t. summation***

$$g_1 * (g_2 + g_3) = g_1 * g_2 + g_1 * g_3 \tag{2}$$

enables to apply DoG as two consecutive convolutions with two Gaussian filters, and to subtract the resulting images. More precisely, we can write

$$\begin{aligned} DoG(f) &= f * (h_1 - h_2) && (3)\\ &= f * h_1 - f * h_2. && (4) \end{aligned}$$

Implement equation (4) and apply it on the same image ('van.tif'), using one of the pairs of σ values you tried with (1). Compare the two DoG-filtered images. Are they the same? Is there any difference in the computational time of the two approaches? Which equation, (1) or (4), leads to the more efficient approach? Do you expect that the answer to this question may vary? If so, what can cause different results?

1.3 Non-linear filters

Median filter (you may read about it in Szeliski [2022], Section 3.3.1), is not included in `fspecial` function. To compute this filter you can use MATLAB's function `medfilt2`.

5. ***Open the image* wagon_shot_noise.png*. Perform median filtering on the image using different sizes of the filter masks.***

6. ***Compare visually the effect of median filtering on the image* wagon_shot_noise.png *with the effect of mean and Gaussian filtering (consider 2-3 different sizes of the filters). Which filter performs best on the task of noise removal from this image?***

7. ***Can median filtering be implemented using convolution?***

8. ***Comment on the computational time required for the three types of filters (mean, Gaussian, and median). What can cause the difference in speed? (How and why?)***

9. ***Implement your own* 3×3 *median filtering. You may use the* MATLAB*'s function* `median` *that computes the median element of a vector. Use for instance two nested for-loops to iterate your filter for every neighborhood in the image. The exact behavior on the borders is not so important for this exercise and you may 'cut some corners' here if it helps you. Include the code in your report.***

It is often said that the median filter preserves edges in the images better than, e.g., mean or Gaussian filters. What about small details or thin structures?

10. ***Open the image* `'fundus.png'` *(a noise-free image is available as* `'fundus_reference.png'`*). Apply Median, Mean and Gaussian filters of size 3x3 (with reasonable value of* σ*). Explain the main reasons for the different results. Include the difference images between the noisy and the filtered images in your report and comment on the noise removed.***

We have mentioned bilateral filter as one type of adaptive filters. Try MATLAB's bilateral filter on 'fundus.png' image, by using `imbilatfilt(image, DegreeOfSmoothing, SpatialSigma)`. A reasonable starting point for setting the parameters is a `DegreeOfSmoothing` of 0.001 (for an image with intensities in the range [0,1]) and a `SpatialSigma` of 3.

11. ***What is the objective of adaptive filtering?***

12. ***If you use* `imbilatfilt(image,'NeighborhoodSize', s)`*, you can set* s *to the same (odd!) value as the size of the used Median, Mean, and Gaussian filters.***

    Do you observe the expected behaviour of the bilateral filter? How does this filter achieve image smoothing with edge preservation?

2 Noise reduction in 3D

Reducing noise is one very common image processing task. When designing a noise suppression filter, it may be beneficial to have, and use, some knowledge about the noise type and its properties (like mean, standard deviation etc.).

Assuming that a known reference image exists, the quality of the output of a noise suppression method can be assessed quantitatively. Two examples of such quality measures (introduced during the lecture), are PSNR and SSIM. These measures are both available as MATLAB functions `psnr(restored, reference, max_value)` and `ssim(restored, reference)`. Both of these

Matlab functions support both 2D and 3D images. Read the documentation for more details; you may also take a look at Szeliski [**2022**], Section 2.3.3. and 3.4.2. for brief explanation of PSNR and SSIM).

In this part of the exercise you will be working with an image of the electron density in a hydrogen molecule, H_2. You will try to process a 3D image and will notice some similarities and some differences compared to 2D images processing.

Let us begin by looking at one 2D slice of the 3D image. Run `noise_demo_2D.m`, in the folder "Filtering", for an example of noise addition and removal 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. Check the provided code and note how these types of noise are added to the image.

Now, let us work with the 3D image. Load it by typing `v = readVTK('hydrogen.vtk');` (ReadVTK is located in the filter folder). 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. 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, you can render an isosurface by setting a threshold and creating a surface at this threshold.

We will now turn to artificial degradation of the 3D image by application of noise, and then recovery of the original image. A convenient way of adding noise to an image is to use the `imnoise` function in Matlab (as you could see in the `noise_demo_2D.m` example). Type `help` to see more details on how to specify Gaussian or 'salt & pepper' noise. Create two 3D images: (1) one with added Gaussian noise (mean=0 and sigma=1.0×10^{-5}); (2) the other with added 'salt & pepper' noise, with density 0.01.

13. ***Try to remove the added 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 filtering method achieves better result and why? Assess the quality of the results both by visual inspection and quantitatively - by computing and comparing the*** `psnr` ***and*** `ssim` ***values. How do the quantitative results (PSNR and SSIM values) that you have obtained correspond to your visual assessment of the images - which measure do you find better matching your subjective evaluation of the restoration performance?***

We discussed the main idea behind denoising by averaging (by computing mean or median) over a small neighbourhood: It approximates averaging over a number of repeated image acquisitions of the same scene (with assumed same imaging conditions). Inevitably present random noise can, by that, be removed. In this approach, the mean or median values are repeatedly computed over a sequence of voxels at the same spatial location in each image in the whole set (instead of averaging over a neighbourhood at that position in *one* image.

You can create such a set of images by creating, let us say, 27 images with Gaussian noise added. An example code for achieving this is:

```
array = zeros([size(v),27]);
for i = 1 : 27
  array(:,:,:,i) = imnoise(v,"options");
end
```

Use the same settings for adding the Gaussian noise as recommended above.

14. ***For each spatial location in the image, take the mean value over the 27 voxels at that same location in the generated 27 noisy images. Compare the resulting image with the image you obtained by the mean filtering (in the previous question). Perform both qualitative evaluation (by visual inspection of the two results), and quantitative evaluation - by computing and reporting the*** `psnr` ***and*** `ssim` ***values.***

3 Filtering in Frequency Domain - Fast Fourier Transform

Images can be filtered in the frequency domain, by applying the *Convolution Theorem* which states that convolution in the spatial domain corresponds to multiplication in the frequency domain, and vice versa. By applying forward and inverse Fourier Transform, we can move the images from one domain to the other, and perform the filtering in the preferred one, for each particular situation. Fast Fourier Transform (FFT) enables efficient ($\mathcal{O}(N \log N)$), for an image with N pixels) computation of the Discrete FT (DFT), as well as its inverse. A very brief overview of the topic can be found in Szeliski [2022], Section 3.4.

Open the image **lines.png** (in the Filtering in Frequency Domain folder). Transform the image using FFT and display the magnitude of the obtained transform in the Fourier domain. Useful commands are:

- `double` - the integer data type is not suitable input for FT;
- `fft2` - the transform itself;
- `fftshift` - the function which re-centres the obtained transform so that the origin (0 frequency) is shown in the centre of the transform;
- `abs` - to compute (and then visualise) the magnitude of the resulting complex-valued Fourier transform;
- `log` - to stretch the intensity range of the obtained FT values, especially around the zero-level;
- `imagesc` - to display the result.

15. ***Repeat the same procedure with the image*** **cameraman.tif** ***(available in*** MATLAB***), as you followed on*** **lines.png**. ***Comment on the spectrum that you see and find correspondences with the original image representation in the spatial domain. Can you explain the different bright straight-line-like structures in the Fourier domain? What causes the appearance of the vertical and horizontal bright lines? How do you interpret the bright region/spot in the centre of the image?***

    Can the original **cameraman.tif** ***image be reconstructed from the image you have displayed? Why?***

You can use the DFT to perform filtering in the frequency domain. To transform back to the image or spatial domain, you use the command `ifft2`. However, if you make modifications

to the FFT representation, you need to take extra care in order to ensure that the inverse-transformed image is a real-valued function. Also, remember to apply `ifftshift` again before you apply `ifft2`. The `i/fftshift` command is useful mainly for visualization purposes, to place the origin (zero-frequency) in the centre of the image; inverse transform is computed on the original main period and therefore requires the shift to be applied again ("undone"). In other words, either skip `i/fftshift` or make sure to apply it twice, before and after visual inspection and manipulation of the Fourier spectrum, i.e. after the forward DFT, and before the inverse DFT.

16. ***Modify the DFT representation of* cameraman.tif, setting certain frequencies to 0, to create its low-pass filtered version. Use, e.g., a circular symmetric filter for best result, but feel free to simplify the task by using a square-shaped filter instead. You may try a band-stop/pass filter, by setting, e.g. `A(20:30, 50:60) = 0`. Once inverse-transformed to the spatial domain, the resulting image should be real valued (after performing `ifft`). You should achieve that by ensuring suitable symmetries when filtering in the Fourier domain, and not by forcing a real valued result by e.g., using the functions `real` or `abs`.**

17. ***Look at the FFT representation of the Gaussian filter and the DoG filter which you have previously used in this lab (Task 1.2). Include the corresponding figures in your report and discuss if they can be considered high-pass, low-pass, or band-pass filters.***

Now open the image **freqdist.png**. There is a pattern (structured noise) present in the image that should be filtered out. Remove it using notch filters in the frequency domain. This is similar to the task (above) that you just addressed, but you may need to be slightly more artistic when selecting which frequencies to attenuate. Remember to ensure the proper symmetry, so that your end result is a real-valued image after inverse-transformed by `ifft2`. Sometimes filtering produces signals that are outside the [0,255] range, keep that on mind when displaying the results by using `caxis`, or perhaps if using the command `imcontrast`. The image you are aiming to "discover" (by filtering) is an aerial image of a city. Your goal is to filter out the periodic pattern well enough to recognize four bridges spanning across the river going through the city.

18. ***Create a filter in the frequency domain that suppresses the pattern in* freqdist.png*, but leaves the rest of the image as intact as possible. What does the filter look like? What do you see in the filtered image? Like the previous question, the resulting image should be a real-valued function after performing* `ifft` *and returning to the spatial domain. You should achieve this by ensuring suitable symmetries when filtering in the Fourier domain, and not by forcing a real valued result by e.g., using the functions* `real` *or* `abs`*.***

4 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. Considering that the aim of restoration is to reverse the degradation process and generate the image as similar as possible to the original, initially degraded one, it is very important to establish quantitative evaluation criteria - i.e., to establish a suitable measure of the quality of a restored image (restoration is an objective process). The measures introduced in Task 2

above, PSNR and SSIM, are often used for this purpose. Recall: you can use MATLAB functions `psnr(restored, reference, max_value)` and `ssim(restored, reference)` to compute these measures.

Let us 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 and plot the measured quality values as functions of the level of degradation applied.

19. ***PSNR and SSIM rely on different assessment criteria (have different objectives). Which are these criteria for each of the two measures? How do the quantitative results (PSNR and SSIM values) that you have obtained correspond to your visual assessment of the images - which measure do you find better matching your subjective evaluation of the restoration performance? Include the plots and selected images to support your arguments! Check your lecture notes regarding the relationship between high PSNR and visual image quality assessment.***

Now try to estimate the noise parameters from the images. Then compare the estimated parameters with the known parameters that you used for degradation, to assess performance of the applied noise estimation approach. Generate the 'cameraman.tif' image degraded by Gaussian noise with $\sigma = 0.05$. Denote it by 'cameraman noisy.tiff'. Specify a suitable homogeneous region in this image, and compute some statistics on that region to derive relevant parameters of the noise distribution. Useful MATLAB functions, as well as functions provided in the lab material, are:

- `roipoly` - selection of a region;
- `histroi` - computation of the histogram of a selected region;
- `statmoments` - computation of the statistical parameters of a distribution;
- `[row, col] = find(Z)` - returns the row and column subscripts of each nonzero element in array Z;
- `help` - always good to have on mind!

20. ***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?***

5 Image restoration by regularized energy minimization

Calculus of variations is a powerful tool for minimizing (or maximizing) functionals. Recall: functionals are mappings from a set of functions to the set of real numbers. If we interpret a 2D-image as a 2D-function, we can use tools from this branch of mathematics to help us perform image restoration (and a number of other image processing tasks). We can define a

suitable objective function (a.k.a. energy function) and then find the image which optimizes this function and best fulfills our objective. We can follow this approach and formulate our restoration problem as an energy minimization problem: *for all possible images X, find the one ($\hat{X}$) which has minimal energy.* More formally, find $\hat{X}$ such that

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

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 restore.

5.1 Denoising

A "classic" way to denoise an image by energy minimization is known as Total Variation (TV) Denoising. It is also referred to as ROF denoising, acknowledging 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.

21. What is the role of the parameter mu? Can you manage to tune it better than the value given in the example? How does the parameter mu relate to the level of image noise that we wish to remove?

22. Do the two performance measures (PSNR and SSIM) agree on which restored image has the highest quality?

5.2 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:

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

Let us start with generating a synthetically blurred version of 'cameraman.tif' by convolving (filtering) it with a Gaussian PSF, and then adding Gaussian noise to the blurred intermediate result. Name the created blurred and noisy image 'cameraman noisy blurred.tif'.

The provided code `EM\_DeblurTV.m` (in the directory `NoiseModeling and ImageRestoration`) is similar as `EM\_DenoiseTV.m` but it takes into consideration the PSF, and the blur, as well. Tune the parameter *mu* in the provided code and comment on its effect. Read the `help` and test-run the `Example` provided there.

Inverse filtering is another way to restore a degraded image. Try to apply it on 'cameraman noisy blurred.tif' by using MATLAB function `deconvwnr`. If you assume that NSR (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; by this change we define, and apply, a Wiener filter. Try to tune NSR and observe the effect (you can get a hint how to do that from `help`).

23. What is the main obstacle for successful image restoration by inverse filtering?

24. ***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?***

25. ***What are, in your opinion advantages and disadvantages of the two compared restoration approaches?***

6 Template matching in frequency domain

Template matching is a technique used to find a location (or sometimes multiple locations) in an image where a template - smaller image, image patch - "fits best". It is often performed by computing, at different positions, (normalized cross-) correlation between the template and the image, and detecting the position of its maximal value. Challenges are imposed by, e.g., imaging conditions and small (or big) differences between the template and the image content, presence of noise, occlusions, but also the properties of the used similarity/distance measure (e.g., normalized cross-correlation) which, due to existence of many local minima, in general make optimization challenging.

Template matching can be performed in the frequency domain as well. One way to do it is to compute `Phase Correlation`, which corresponds to the Normalized Cross-Correlation, now in the frequency domain. Application of the inverse DFT provides the location of the matched template. Let us try it!

Use the files in the `TemplateMatching` directory.

26. ***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?***

27. ***Try to implement template matching based on phase-correlation.***

28. ***Can you detect the location of the*** `template1.png` ***in the image*** `search.png`***?***

29. ***Can you use the same approach for locating*** `template2.png` ***in*** `search.png`***? Why?***

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.