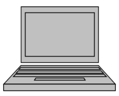
LAB 3: Operations on grayscale images

Maria Magnusson and Michael Felsberg, 2017
Division of Computer Vision (CVL), Dept of Electrical Engineering (ISY),
Linköping University

Latest uppdate: Maria Magnusson, Februari 2024

1 Introduction

Read the booklet before the laboratory exercise. Exercises/questions marked with a pointing hand should be resolved in preparation before the laboratory exercise. Suggested answers for all questions, as well as a multiple choise table is in the end of this lab booklet. In case of any wrong answer, the problem should be discussed with the teacher.



A computer symbol means that a Python script should be created and demonstrated for the teacher.

A double teacher symbol means that you are advised to fill in parts of the multiple choise table and show some demos to the teacher. Continue with new exercises while you wait!

2 Getting started with the lab environment

Copy the images
`baboon.tif, circle.tif, pirat.npy, pirat2.npy, pattern.npy`
from `/courses/TSKS24/imageLab/` (or Lisam) to your home directory. Then start a suitable editor e.g. `emacs`) open a terminal and start the interactive Python shell:

```
ipython3
```

3 Show images

3.1 Basics about images in Python

In order to work with images in Python, the following packages must be imported:

```
import numpy as np
from scipy import signal, misc
from matplotlib import pyplot as plt
plt.rcParams['image.interpolation'] = 'nearest'
```

The first package (numpy) is a Python package for managing matrices. The other two packages are Python packages to provide more advanced scientific operations (scipy) and plots (pyplot). The fourth line is needed only for `matplotlib` with version older than 2. All Python scripts must contain this *preamble*. It is recommended to save these lines in a file tt preamble.py, which is then executed with `from preamble import *`.

3.2 Show images in Python

There are several commands for displaying images in Python, but the simplest variant (which is also similar to Matlab) is `plt.imshow()`:

```
plt.imshow(Im,'gray')
plt.show(block=False)
```

In this example, `Im` contains the (grayscale) image itself. The commands automatically scales the image so that the pixels become quadratic, the smallest pixel value becomes black and the largest pixel value becomes white in a linear grayscale. There are color tables other than `gray`, e.g. `jet`. It is also possible to create your own color table. If you want to map the values to another range, you can set this as:

```
plt.imshow(Im,'gray',clim=(min,max))
```

so that the image is displayed with a linear color scale between the values `min` and `max`. With the command:

```
plt.colorbar()
```

you get a gradation next to the image that indicates which colors and pixel values correspond to each other. In PYTHON you must end all plot commands with a special command to create or update the current plot, namely:

```
plt.show(block=False)
```

The parameter `block=False` ensures that all figure windows are displayed, followed by an immediate return to the command shell.

3.3 Exercises

Create a file `DemoA.py` with preamble and the contents below.

```
from preamble import *
Im = np.double(plt.imread('baboon.tif'))
plt.figure(1)
plt.subplot(121)
plt.imshow(Im,'gray',clim=(0,255))
plt.title('original image')
plt.colorbar()
plt.show(block=False)
```

Execute it in the interactive PYTHON shell with:

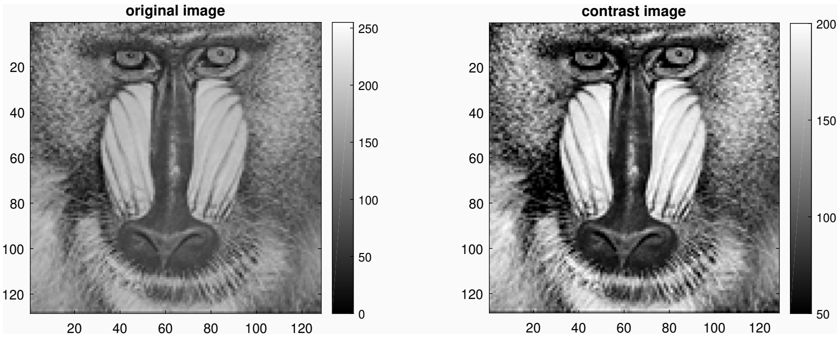
```
run DemoA
```

The command `plt.figure(1)` selects the Figure 1 window. The command `plt.subplot(121)` divides the window into 1 row and 2 columns, i.e. 2 squares, and shows the image in the first of them. Image `Im` contains only one color channel. A color table and range are needed to display the image using `plt.imshow()`. If the image contains three channels, this is perceived as RGB values and the color table is ignored. Enter the commands below. `Im` causes the image matrix to be printed on the screen. Since the matrix is very large, PYTHON will automatically remove most of it. It still gives a good feeling that an image is actually just a matrix. Now give the following commands within the interactive PYTHON shell:

```
Im
np.min(Im)
np.max(Im)
```

QUESTION 1: What are the min and max values of `baboon`?

Unlike Matlab, Python only needs a simple `np.min()` call. If you want to get the minimum value for each column, specify that the operation is to be used on dimension 0: `np.min(Im,0)`.



DEMO A: Extend the file `DemoA.py` so that the monkey appears to the right with higher contrast, for example between 50 and 200 as shown above. Also give the image a suitable title.

QUESTION 2: You can move the cursor over the image and get the value of the current position. Select a position between the monkey's eyes. Which coordinate (X,Y) and what grayscale value (Index) do you get? Also check that you get the same values in the left and right images.

---

4 Color tables

Create a file `DemoB.py`, where you show the monkey in the full window:

```
from preamble import *
Im = np.double(plt.imread('baboon.tif'))
plt.figure(2)
plt.subplot(111)
plt.imshow(Im,'gray',clim=(0,255))
plt.title('original image')
plt.colorbar()
plt.show(block=False)
```

To get the color table as a matrix, give the commands

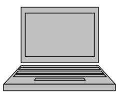
```
graycmap = plt.get_cmap('gray',256)
gray_vals = graycmap(np.arange(256))
```

QUESTION 3: Look at `gray_vals`. Compared to the regular grayscale table shown in the Lecture slides, `gray_vals` has a 4th channel (alpha channel). There is also another small difference. Which one?

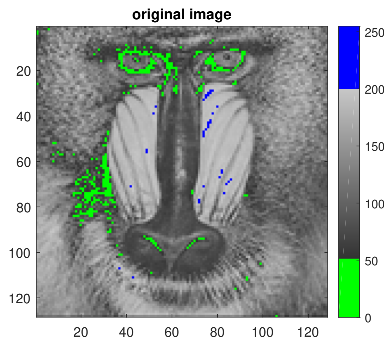
To modify the colortable, give the commands

```
gray_vals[200:] = [1, 0, 0, 1]
plt.register_cmap('modgray',graycmap.from_list('modgray', gray_vals))
```

QUESTION 4: Look at the modified `gray_vals` and give the command `plt.imshow()` with `'modgray'`. Explain how this color table affects the image to the left.



DEMO B: Make your own color table, based on the grayscale color table, but let the values ≥ 200 appear blue and the values ≤ 50 appear green.



QUESTION 5: Finally, test the color table `'jet'` on the monkey. What color does the monkey's nose get?

---

5 Convolution (Swedish: Faltning)

5.1 Weighted averaging filter (low-pass filter)

Create a file `DemoC.py` with the content below and execute it.

```
from preamble import *
plt.figure(3)
Im = np.double(plt.imread('baboon.tif'))
plt.subplot(121)
plt.imshow(Im,'gray',clim=(0,255))
plt.title('original image')
plt.colorbar()
plt.show(block=False)
```

Different filter kernels can be constructed from the two base filters,
`b =` $\begin{array}{|c||c|}\hline 1 & 1 \\ \hline\end{array}/2$ and `d =` $\begin{array}{|c||c|}\hline 1 & \text{-1} \\ \hline\end{array}$.
Note that these two filters have their origin on the border in the middle, i.e. they shift the signal by a 1/2 sample.

On the other hand, `b2 = b * b =` $\begin{array}{|c|c|c|}\hline 1 & [2] & 1 \\ \hline\end{array}/4$ has its origin in the center of the filter kernel.

The filter kernel

`aver = b2 * b2`$^{\mathrm{T}}$ $= \begin{array}{|c|c|c|}\hline 1 & [2] & 1 \\ \hline\end{array}/4 * \begin{array}{|c|}\hline 1 \\ \hline [2] \\ \hline 1 \\ \hline\end{array}/4 = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & [4] & 2 \\ \hline 1 & 2 & 1 \\ \hline\end{array}/16$

can be created in Python as follows:

```
b = np.array([0.5,0.5])
b2 = np.convolve(b,b).reshape(1,3)
aver = signal.convolve2d(b2,b2.T)
```

The command `reshape` is needed to convert the array to a 1×3 matrix of the convolution result.

Run the code and verify that the kernel is correct. Also look at the intermediate results. Then apply the kernel to the monkey with the code:

```
Imaver = signal.convolve2d(Im,aver,'same')
```

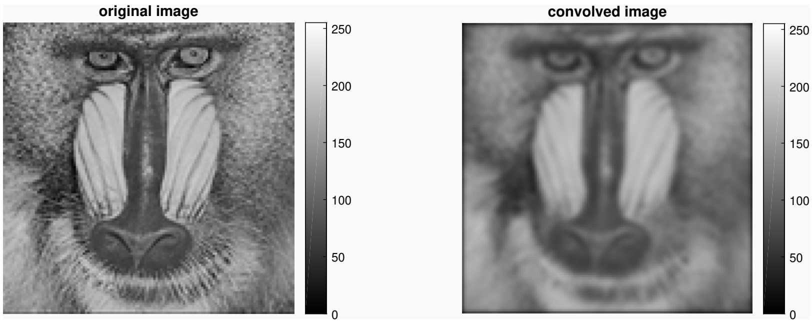
Extend the `DemoC.py` file with this code and display the filtered monkey `Imaver` to the right of the original monkey.

QUESTION 6: Averaging filters are often used for noise reduction, but what happens to fine details, such as edges and lines, in the image?

QUESTION 7: There is a parameter `same` in the `signal.convolve2d()` command. It causes the output image to get the same size as the input image. But how is data outside the image handled before convolution? Check by typing `help(signal.convolve2d)`.

DEMO C: Then try to convolve several times with `aver` for a stronger effect. Show the 3 times convolved monkey to the right. Be sure to have the same contrast window on both images, so that it becomes educational to compare the two images.

QUESTION 8: What happens when the `aver` filter is applied repeatedly?



QUESTION 9: Note that `aver` is divided by the normalization factor 16, which is the sum of the filter coefficients. What happens to the resulting image if the normalization factor is set to a lower value? (You can simulate this by e.g. multiplying `aver` with a value larger than 1, e.g. 1.1.)

Fill in the first column in the multiple choise table in the end of this lab booklet and show demo A, B, C to the teacher!

5.2 Derivation in x- and y-direction, gradient

The filter kernel `d` calculates the finite difference. To avoid shifting with a 1/2 sample, the central difference is often used instead:

`cd =` $\boxed{1}\boxed{[0]}\boxed{-1}/2,$

where

`b * d` $= \boxed{1}\boxed{1}/2 * \boxed{1}\boxed{-1} = \boxed{1}\boxed{[0]}\boxed{-1}/2 =$ `cd`

Write down the corresponding PYTHON code and verify that the result becomes `cd`. Note: Use `1.0` instead of `1`, so that the filter kernel gets a floating point format.

To calculate partial derivatives of an image, you must select one coordinate system. A common choice is to identify the column index with the x coordinate and row index with the y coordinate. The derivative in the x direction of an image $f(x, y)$ can then be calculated according to

$$\frac{\partial f(x,y)}{\partial x} = \frac{\partial}{\partial x} * f(x,y) \approx \text{sobelx} * f(x,y).$$

Similarly, the derivative in the y-direction, $\frac{\partial f(x,y)}{\partial y}$, can be calculated. The selected coordinate system results in sobely = sobelx$^{\mathrm{T}}$. The Sobel filters are shown below.

$$\text{sobelx} = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & [0] & -2 \\ \hline 1 & 0 & -1 \\ \hline\end{array}/8\ , \quad \text{sobely} = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & [0] & 0 \\ \hline -1 & -2 & -1 \\ \hline\end{array}/8.$$

Create a file `DemoD.py` with the content below. Then complete it and execute.

```
from preamble import *
plt.figure(4)
Im = np.double(plt.imread('circle.tif'))
plt.subplot(221)
plt.imshow(Im,'gray',clim=(0,255))
plt.title('original image')
plt.colorbar()

b = np.array([0.5,0.5])
b2 = np.convolve(b,b).reshape(1,3)
d = ...
cd = ...
sobelx = ...

Imsobelx = signal.convolve2d(Im,sobelx,'same')
plt.subplot(223)
plt.imshow(Imsobelx,'gray',clim=(-128,127))
plt.title('sobelx image')
plt.colorbar()

plt.show(block=False)
```

Add code to `DemoD.py` to display the result of sobely convolved with the circle. Show the result image at the bottom right.

When an image contains only positive values from 0 to 255, the **gray** color table works like this:

color:	black	gray	white
pixel value:	0	128	255

The sobel-filtered images contain both positive and negative values. When such an image is displayed in the range [-128,127], `clim=(-128,127)`, this is how the **gray** color table works:

color:	black	gray	white
pixel value:	-128	0	127

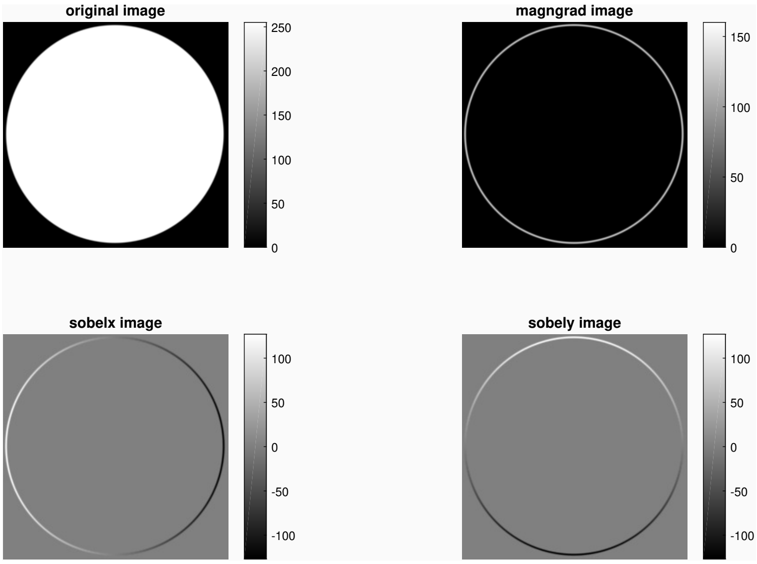
Consequently, negative values appear dark and positive values bright. Values close to 0 are displayed in gray. Values ≤ -128 are displayed in black and values ≥ 128 are displayed in white.

QUESTION 10: Look at your images and tell why the edge of the circle sometimes becomes dark, sometimes bright, and sometimes gray in the sobel-filtered images.

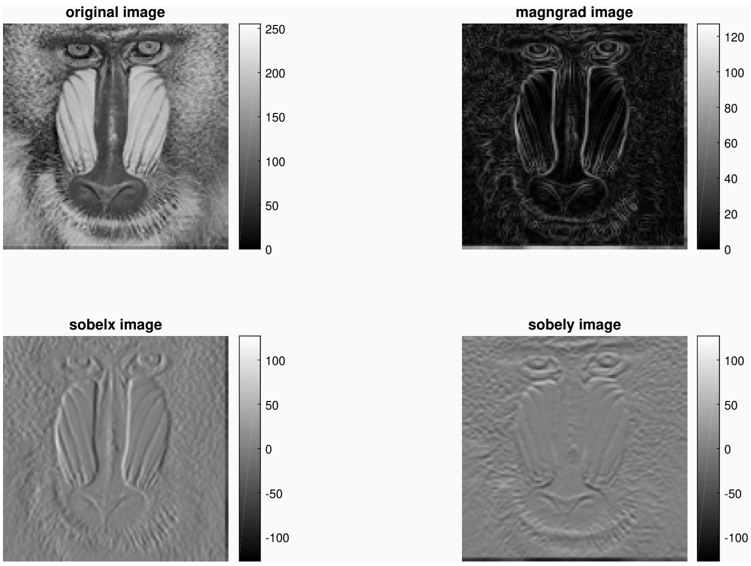
The gradient $\left(\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y}\right)$ is a two-dimensional vector that points in the direction where the intensity of the image $f(x, y)$ increases fastest.

QUESTION 11: Write down the mathematical expression for the magnitude of the gradient of the image $f(x, y)$! (Alternative terms for magnitude are the length or the absolute value.)

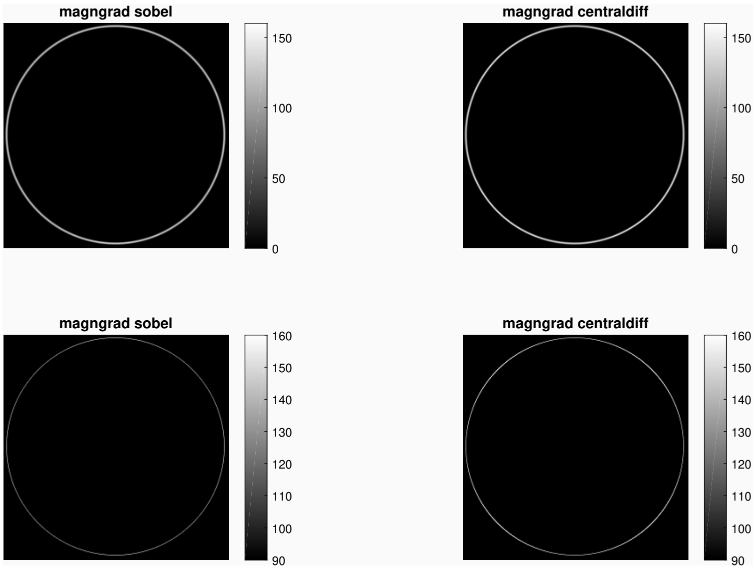
DEMO D: Add code in `DemoD.py` so that the magnitude of the gradient on the circle image is shown at the top right.



DEMO E: Also write a similar script for the monkey, `DemoE.py`, alternatively have the file name as an argument to the script, and study the images.



DEMO F: It is often good enough to use central differences `cd` instead of `sobel`, but if you want to be careful, sobel is better. For the circle, its edge is just as strong all around. Thus, the magnitude of the gradient should also be equally strong around. This is better fulfilled for `sobel` than `cd`. Show this with the script `DemoF.py`. To see clearly, you need to change the contrast interval. Note the difference between the two lower images below.

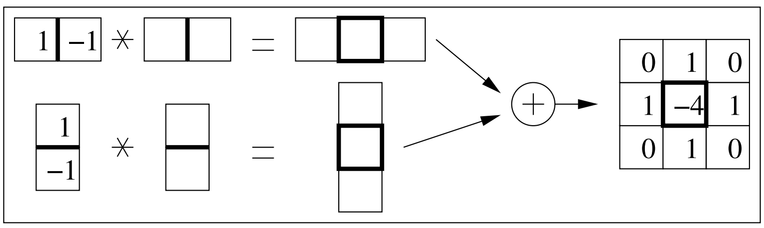


5.3 Laplace filter (*negative* high pass filter)

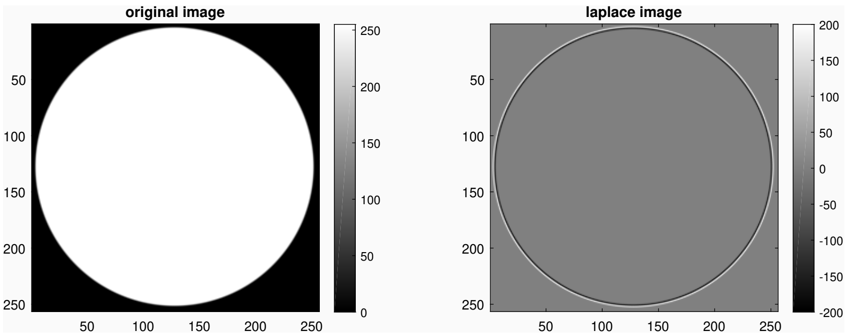
The Laplace operator is definied as

$$\nabla^2 = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) = \frac{\partial}{\partial x} * \frac{\partial}{\partial x} + \frac{\partial}{\partial y} * \frac{\partial}{\partial y}$$

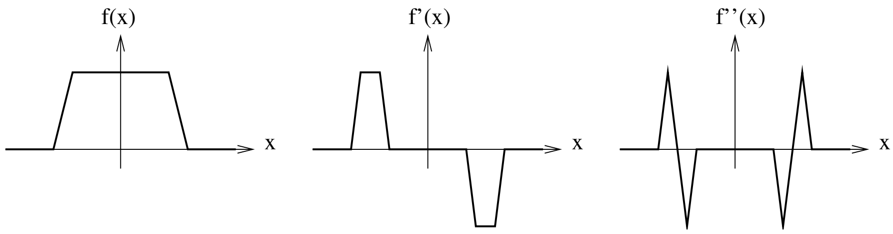
QUESTION 12: Design a Laplace filter by filling in the boxes below. As you can see, the Laplace filter is given. Its center is marked with a thicker frame. The derivative operators used to derive the Laplace filter, have their centers between the two pixels.



DEMO G: Create a file `DemoG.py` where the circle is convolved with the Laplace filter so that the figure below is obtained.



The Laplace operator thus estimates a kind of 2-D second derivative. Below is shown how a 1-D second derivative reacts to an (approximate) 1D rectangular function $f(x)$.

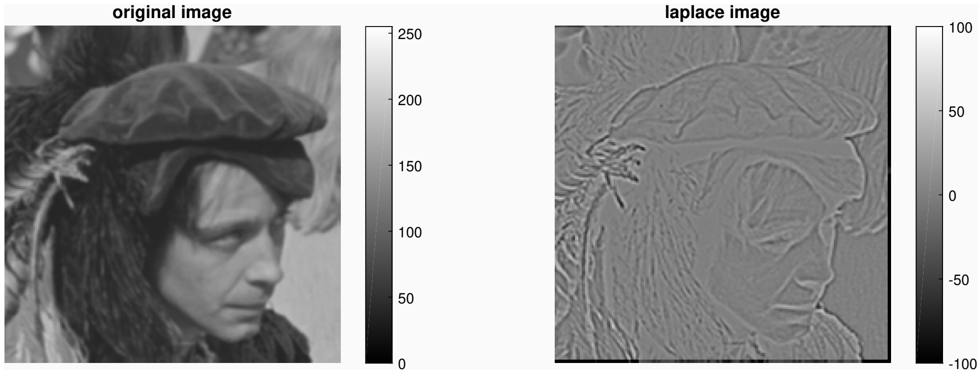


QUESTION 13: Does the Laplace image from DEMO G match the 1D sketch above? Motivate your answer.

Create a file `DemoH.py` with similar content as `DemoG.py`, but load another image:

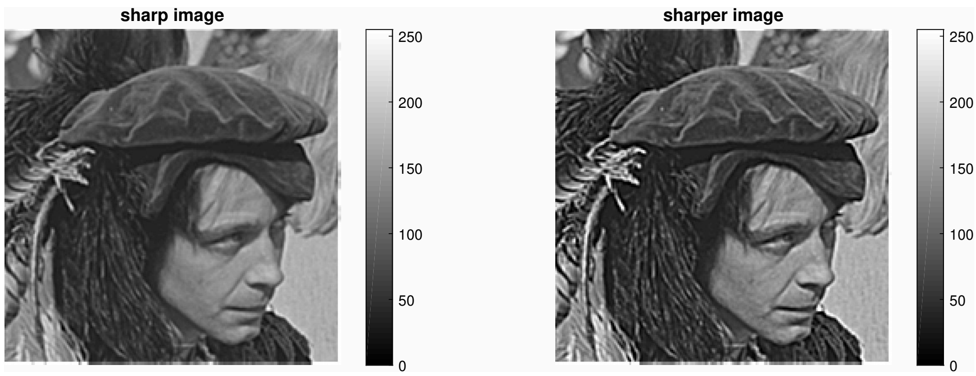
```
Im = np.load('pirat.npy')
```

Call the filtered image `ImLaplace`. You may also need to adjust the limits `(-200,200)` to `(-100,100)` to get a better contrast, see below:



QUESTION 14: Look at the filtered Laplace image and compare it with the original image. What happens in smooth regions (e.g. cheek)? What happens at edges? What is the effect in highly dynamic regions (e.g. feather)?

DEMO H: Consequently, the Laplace filter amplifies rapid changes in the image, like a high-pass filter (HP). However, the Laplace filter has a negative sign compared to a normal high-pass filter. Therefore form `ImHP = -ImLaplace`. Then complete `DemoH.py` with an image that is the sum of the original image and the high-pass image, i.e. `Imsharp = Im + ImHP`. Also create `ImSharp2 = Im + 2 * ImHP`. Note that the edges become more distinct with an increasing proportion of ImHP. However, the noise is amplified.



QUESTION 15: It is important with correct sign! What happens if the sign is changed, i.e. `Imsharp = Im - ImHP`?

Fill in the second column in the multiple choise table in the end of this lab booklet and show demo D, E, F, G, H to the teacher!

6 Aliasing (Swedish: Vikningsdistorsion)

Create a file `DemoI.py` with the contents below and execute it.

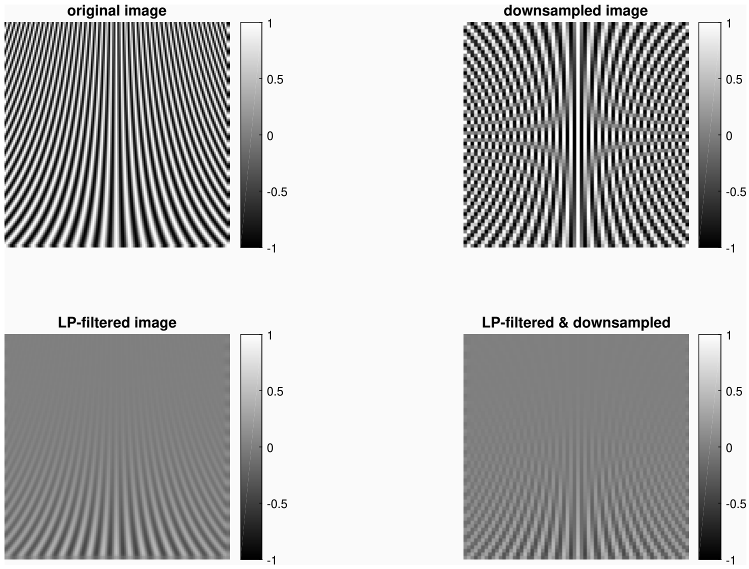
```
from preamble import *
Im = np.load('pattern.npy')
plt.figure(9)
plt.subplot(221)
plt.imshow(Im,'gray',clim=(-1,1))
plt.title('original pattern')
plt.show(block=False)
```

Downsample the pattern a factor of 2 in both dimensions with `Im[::2,::2]` and show the result to the right.

QUESTION 16: Explain what has happened to the upper part of the right image?

The observed effect is undesirable and is due to the image not being preprocessed properly. Images should be preprocessed with low-pass filtering before downsampling! The filter `aver` used in DEMO C is a suitable low-pass filter.

DEMO I: Before downsampling, perform repeated convolutions with `aver`. Vary the number of convolutions so that the aliasing disappear.



QUESTION 17: How many convolutions were needed?

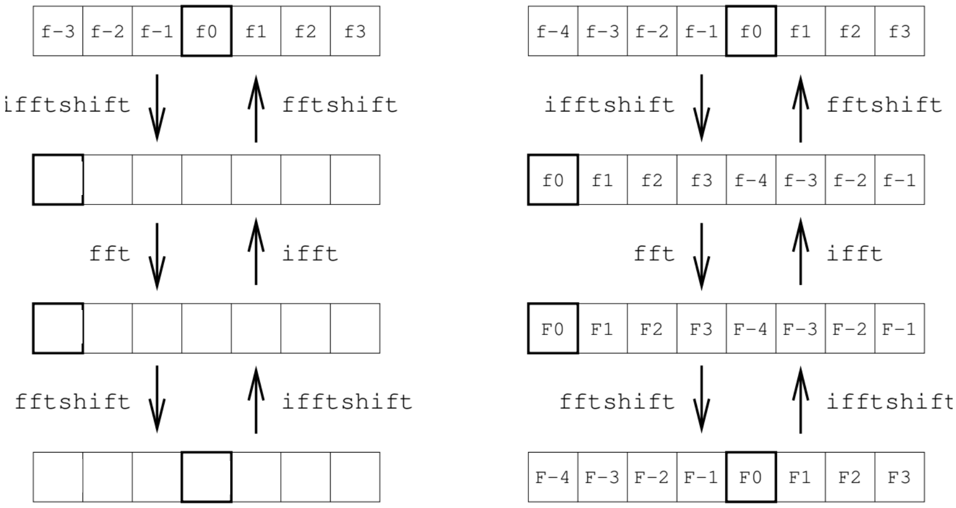
QUESTION 18: Also explain, in terms of frequencies, why repeated low-pass filtering was useful here.

---

7 Visualization of the Fourier spectrum

In the lecture it was mentioned that `np.fft.fft2` can be combined with `np.fft.fftshift()` and `np.fft.ifftshift()` to get the origin in the middle of both the image and the Fourier transform. Both functions give the same result for an even number of samples, but different for an odd number of samples.

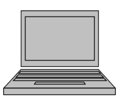
QUESTION 19: Complete the following figure by testing in PYTHON. (Hint: create a test array with values `[0,1,2,3,4,5,6]`)



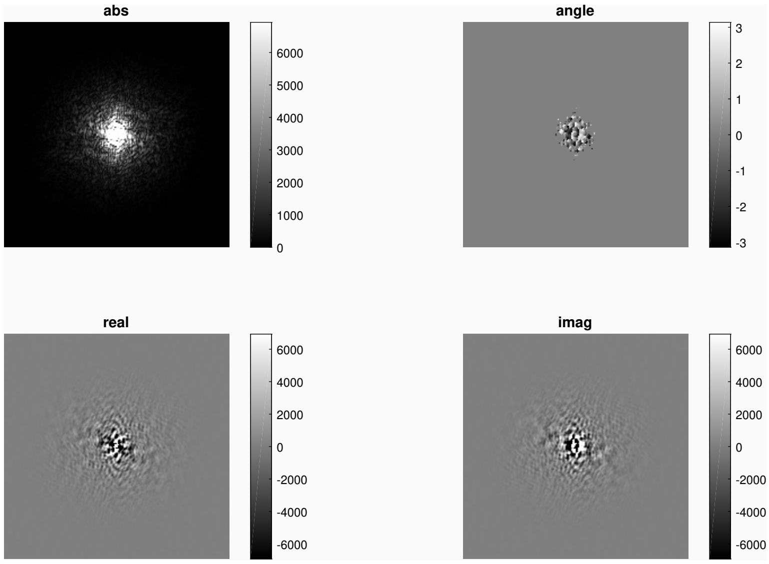
Create a file `DemoJ.py` and execute it:

```
from preamble import *
Im = np.load('pirat2.npy')
IM = np.fft.fftshift(np.fft.fft2(np.fft.ifftshift(Im)))
plt.figure(10)
plt.subplot(221), plt.imshow(np.abs(IM),'gray'), plt.colorbar()
plt.subplot(222), plt.imshow(np.angle(IM),'gray'), plt.colorbar()
plt.subplot(223), plt.imshow(np.real(IM),'gray'), plt.colorbar()
plt.subplot(224), plt.imshow(np.imag(IM),'gray'), plt.colorbar()
plt.show(block=False)
```

QUESTION 20: Why do images 1 and 3 show only a white dot in the middle?



DEMO J: Set -0.1 times the minimum value in the real part to `maxV`, i.e. `maxV = -0.1*np.min(np.real(IM))`.
For the abs image: Let 0 be shown in black and `maxV` be shown in white.
For the real and imag images: Let `-maxV` appear in black and `maxV` appear in white.
For the phase image (angle): Set it to 0 where the amplitude (abs) is less than 10 times the mean of the abs.



QUESTION 21: Which of the images are even and which are odd? If necessary, zoom in the images to see better.

Fill in the third column in the multiple choise table in the end of this lab booklet and show demo I, J to the teacher!

8 Suggested answers

ANSWER 1:
a) 2 and 207.
b) 0 and 255.
c) 5 and 243.

ANSWER 2:
a) (X,Y) $\approx$ (95,30), index $\approx$ 103.
b) (X,Y) $\approx$ (62,16), index $\approx$ 84.
c) (X,Y) $\approx$ (23,58), index $\approx$ 92.

ANSWER 3:
a) `gray_vals` has values between 0 and 100 instead of between 0 and 255.
b) `gray_vals` has values between 0 and 10 instead of between 0 and 255.
c) `gray_vals` has values between 0 and 1 instead of between 0 and 255.

ANSWER 4:
a) Values = 200 are shown in red.
b) Values $\leq$ 200 are shown in red.
c) Values $\geq$ 200 are shown in red.

ANSWER 5:
a) magenta
b) yellow
c) cyan

ANSWER 6:
a) They become sharper.
b) They become blurred.
c) They change orientation.

ANSWER 7:
a) Before convolution, data outside the image are padded with zeros.
b) Before convolution, data outside the image are extrapolated.
c) Before convolution, the image is repeated in the x- and y-directions.

ANSWER 8:
a) The resulting image gets more and more bright.
b) The resulting image gets more and more dark.
c) The resulting image gets more and more blurred.

ANSWER 9:
a) The resulting image is still blurred, but also brighter.
b) The resulting image is still blurred, but also darker.
c) The resulting image is still blurred, but have a better contrast.

ANSWER 10:
a) Positive derivative (transition from dark to bright) gives a bright pixel. Negative derivative (transition from bright to dark) gives a dark pixel.
b) Positive derivative (transition from dark to bright) gives a dark pixel. Negative derivative (transition from bright to dark) gives a bright pixel.
c) Positive derivative (transition from dark to bright) gives a bright pixel. Negative derivative (transition from bright to dark) gives a bright pixel.

ANSWER 11:
a) $\left|\frac{\partial f(x,y)}{\partial x}\right| + \left|\frac{\partial f(x,y)}{\partial y}\right|$
b) $\sqrt{\frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}}$
c) $\sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2}$

ANSWER 12:

a) $\begin{array}{|c||c|}\hline 1 & -1 \\ \hline\end{array} * \begin{array}{|c||c|}\hline -1 & 1 \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline 1 & \mathbf{-2} & 1 \\ \hline\end{array}$

$$\begin{array}{|c|}\hline 1 \\ \hline\hline -1 \\ \hline\end{array} * \begin{array}{|c|}\hline -1 \\ \hline\hline 1 \\ \hline\end{array} = \begin{array}{|c|}\hline 1 \\ \hline \mathbf{-2} \\ \hline 1 \\ \hline\end{array}$$

b) $\begin{array}{|c||c|}\hline 1 & -1 \\ \hline\end{array} * \begin{array}{|c||c|}\hline 1 & -1 \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline -1 & \mathbf{2} & -1 \\ \hline\end{array}$

$$\begin{array}{|c|}\hline 1 \\ \hline\hline -1 \\ \hline\end{array} * \begin{array}{|c|}\hline 1 \\ \hline\hline -1 \\ \hline\end{array} = \begin{array}{|c|}\hline -1 \\ \hline \mathbf{2} \\ \hline -1 \\ \hline\end{array}$$

c) $\begin{array}{|c||c|}\hline 1 & -1 \\ \hline\end{array} * \begin{array}{|c||c|}\hline 1 & -1 \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline 1 & \mathbf{-2} & 1 \\ \hline\end{array}$

$$\begin{array}{|c|}\hline 1 \\ \hline\hline -1 \\ \hline\end{array} * \begin{array}{|c|}\hline 1 \\ \hline\hline -1 \\ \hline\end{array} = \begin{array}{|c|}\hline 1 \\ \hline \mathbf{-2} \\ \hline 1 \\ \hline\end{array}$$

ANSWER 13:
a) Yes, a bright (positive) line are obtained to the left of the circle and a dark (negative) line are obtained to the right of the circle.
b) Yes, a bright (positive) line are obtained at the edge of the circle.
c) Yes, double lines are obtained at the edge of the circle (one positive and one negative).

ANSWER 14:
a) Smooth regions get values close to 0.
Edges of the e.g. the shoulder gets double-lines (one positive and one neg.).
Edges and details which contain high frequencies are enhanced.
b) Smooth regions look the same.
Edges of the e.g. the shoulder gets double-lines (one positive and one neg.).
Edges and details which contain high frequencies are enhanced.
c) Smooth regions look the same.
Edges of the e.g. the shoulder get negative values.
Edges and details which contain high frequencies are enhanced.

ANSWER 15:
a) High frequencies are amplified and the resulting image becomes sharper than the original image.
b) High frequencies are cancelled and the resulting image becomes similar to the original image.
c) High frequencies are removed and the resulting image becomes more blurred than the original image.

ANSWER 16:
a) The upper half of the right image has been exposed to aliasing. Downsampling a factor of 2 lowers the sampling frequency a factor of 2 and the risk for aliasing increases.
b) The upper half of the right image has been exposed to aliasing. Downsampling a factor of 2 increases the sampling frequency a factor of 2 and the risk for aliasing increases.
c) The upper half of the right image has been exposed to aliasing. Downsampling a factor of 2 increases the contrast of the image and the risk for aliasing increases.

ANSWER 17:
a) Approximately 2
b) Approximately 4
c) Approximately 6

ANSWER 18:
a) Lowpass filtering attenuates the high frequencies so that they give less aliasing.
b) Lowpass filtering amplifies the high frequencies so that they give less aliasing.
c) Lowpass filtering inverts the high frequencies so that they give less aliasing.

ANSWER 19:

a)
```
f-3 f-2 f-1 f0  f1  f2  f3
f0  f1  f2  f3  f-3 f-2 f-1
F0  F1  F2  F3  F-3 F-2 F-1
F-3 F-2 F-1 F0  F1  F2  F3
```

b)
```
f-3 f-2 f-1 f0  f1  f2  f3
f-1 f0  f1  f2  f3  f-3 f-2
F-1 F0  F1  F2  F3  F-3 F-2
F-3 F-2 F-1 F0  F1  F2  F3
```

c)
```
f-3 f-2 f-1 f0  f1  f2  f3
f0  f-1 f-2 f-3 f3  f2  f1
F0  F-1 F-2 F-3 F3  F2  F1
F-3 F-2 F-1 F0  F1  F2  F3
```

ANSWER 20:
a) The frequency 101 Hz in both directions dominates so that all other values are mapped to black.
b) This is normal for 'abs' and 'real' of the Fourier transform. All black values correspond to 0.
c) The DC frequency (0 frequency) dominates so that all other values are mapped to black.

ANSWER 21:
a) abs and real are even. angle and imag are odd. This always applies when a real (non-complex) image is Fourier transformed!
b) real and imag are even. abs and angle are odd. This always applies when a real (non-complex) image is Fourier transformed!
c) abs and angle are even. real and imag are odd. This always applies when a real (non-complex) image is Fourier transformed!

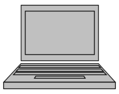
9 Examination

Multiple choise table

	a	b	c
1:			
2:			
3:			
4:			
5:			
6:			
7:			
8:			
9:			

	a	b	c
10:			
11:			
12:			
13:			
14:			
15:			

	a	b	c
16:			
17:			
18:			
19:			
20:			
21:			



Questions and Demonstrations approved by the teacher

Column 1 Demo A,B,C	Column 2 Demo D,E,F,G,H	Column 3 Demo I,J