Lab 1: Basic operations on images in Matlab

Maria Magnusson with contributions by Michael Felsberg, 2017,
updated 2019,
Computer Vision Laboratory, Department of Electrical Engineering,
Linköping University

1 Introduction

Read all instructions before the lab-exercise. Exercises marked with a pointing hand are supposed to be solved as preparation before the lab-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 indicates that a MATLAB-script has to be written. Save all your MATLAB-scripts and demonstrate them to the teacher.

An exercise marked with 'Extra' can be performed in terms of time and interest.

Start by extracting the zip-archive containing the files `baboon.tif, circle.tif, pirat.mat, mathlogo.mat, peppersmall.mat` into a suitable folder.

2 Displaying images

2.1 Displaying images in MATLAB

There are two commands for displaying images in MATLAB, `imagesc` and `imshow`. The second one automatically scales the image so that the pixels become rectangular, removes the axes and sets the colortable to grayscale.

(with 256 different values). Consequently

```
imagesc(im);                    or          imshow(im, []);
axis image;                                 colormap(gray(256));
axis off;
colormap(gray(256));
```

give the same result. In the commands above, `im` contains the image and `[]` automatically finds the minimum and maximum of the image and displays it with a linear scale in between. Note that `[]` is not needed in `imagesc`. There exists also other colortables, i.e. `jet`, or it is possible to design your own colortable.
If another range of pixel values is desired, you can write

```
imagesc(im, [min max]);          or          imshow(im, [min max]);
```

giving that the image is displayed with a linear scale between the values `min` and `max`. The command

```
colorbar;
```

is useful in connection with displaying images. It shows how the colors in the image correspond to pixel values. The command

```
imshow(im);
```

displays the image with a linear scale between 0 and 1. One exception to what have been said above is 8-bit images of class uint8, which may contain values between 0 and 255. For such images this command works fine:

```
imshow(im);
```

2.2 Exercises

Create a file `ShowMonkey.m` with the subsequent contents and execute it.

```
im = double(imread('baboon.tif'));
figure(1)
colormap(gray(256))
subplot(1,2,1), imagesc(im, [0 255])
axis image; title('original image')
colorbar('SouthOutside')
```

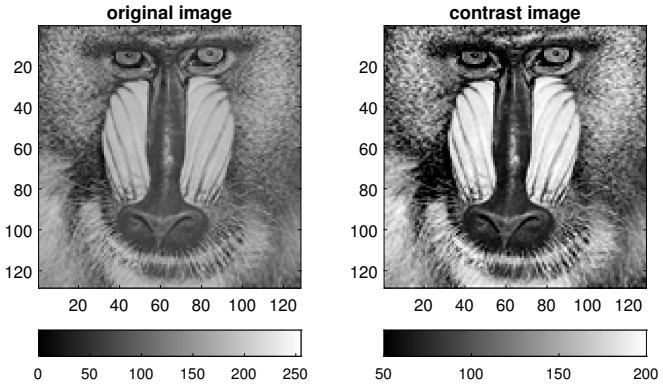
The command `subplot(1,2,1)` divides the figure window into 1 row and 2 columns, i.e., 2 cells, and shows the image in the first one. Note that the commas can be omitted `subplot(121)`. Enter the commands below. Writing `im` without semicolon results in the whole image matrix being printed on the screen and helps to understand that an image is a matrix in MATLAB.

```
>> im
>> min(min(im))
>> max(max(im))
```

QUESTION 1: What is the min- and max-value of `baboon`?

We need to write `min(min( )` because a single `min` outputs the minimum value for each column. The second `min` outputs the minimum of the resulting row vector. Check out the details by typing `help min` or `doc min`.

DEMO A: Extend the file `ShowMonkey.m` such that the monkey is shown on the right with higher contrast, e.g. between 50 and 200. Name the image with a suitable title.



QUESTION 2: Choose the symbol 'data cursor' (yellow with +-symbol) in the figure-window. Click in the middle between the monkey's eyes. Which coordinates (X,Y) and grayscale (index) do you get? Verify that you get the same value in the left and right image.

---

3 Colormaps

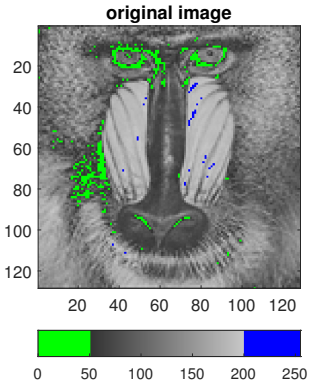
Give the commands

```
>> mycolormap0 = gray(256);
>> mycolormapR = mycolormap0;
>> mycolormapR(201:256,:) = ones(56,1)*[1 0 0];
```

QUESTION 3: Look at `mycolormap0` (by removing the semicolon) and compare it with the normal gray scale color map presented in the first lecture. The principal is the same, but they do differ slightly, e.g. has `mycolormap0` addresses between 1 and 256 instead of between 0 and 255. Mention one more difference!

QUESTION 4: Look at `mycolormapR` and give the command:
`colormap(mycolormapR)`
Explain how this colormap influences the image to the left.

DEMO B: Add code in `ShowMonkey.m` so that the Monkey is shown also in `figure(2)`. Make your own colortable, based on the gray scale color map but let values ≥ 200 be shown in blue and values ≤ 50 be shown in green.



QUESTION 5: Finally check the colortable `jet` on the monkey. What color does the monkey's nose get?

---

4 Convolution (Swedish: Faltning)

4.1 Weighted averaging filter (lowpass-filter)

Create a file `ConvolveMonkey.m` with the subsequent content and execute it.

```
im = double(imread('baboon.tif'));
figure(1)
colormap(gray(256))
subplot(1,2,1), imagesc(im, [0 255])
axis image; title('original image')
colorbar('SouthOutside')
```

The filter kernel `aver =`

1	2	1
2	[4]	2
1	2	1

/16,

where the origin has been marked with square brackets. It can be implemented in MATLAB and be applied to the monkey with the code:

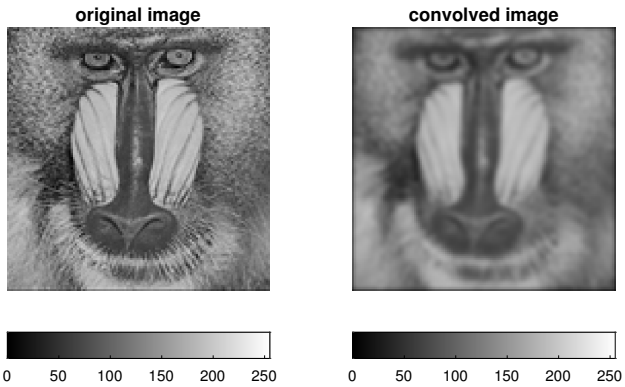
```
aver = [1 2 1; 2 4 2; 1 2 1] /16
imaver = conv2(im,aver,'same');
```

Extend the file `ConvolveMonkey.m` with this code and display the filtered monkey `imaver` to the right of the original monkey.

QUESTION 6: Averaging filters are often applied to regularize or reduce noise. However, what happens to fine details, edges and lines, in the image?

QUESTION 7: When using `same` in the `conv2`-command, the output image gets the same size as the input image. How are data outside the input image handled? Verify by typing `help conv2`.

DEMO C: In order to increase the effect of the filtering, convolve the image multiple times with the filter kernel `aver`. Modify `ConvolveMonkey.m` such that the 3 times convolved monkey is shown to the right. Make sure that you use the same contrast window for both images.



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

QUESTION 9: The filter kernel `aver` is divided by the normalizing factor 16, the sum of the filter coefficients. Reduce the normalizing factor and observe the result. What happens?

QUESTION 10: The parameters `full` and `valid` are alternatives to `same` in the `conv2` command.
`'full'` returns the full 2-D convolution.
What does `'valid'` mean?

EXERCISE I: Perform the convolution below by hand as preparation before the lab-exercise. Verify your result by comparing with the `conv2` command.

1	2	0
1	[3]	0
1	4	1

$*$

1	2	0
0	[2]	0
0	1	3

$=$

5 Fourier transform

5.1 Displaying Fourier transforms in MATLAB

The discrete Fourier transform (DFT) can be computed fast by the fast Fourier transform (FFT). The MATLAB function `fft` assumes the origin of the data to be at the first position of the input vector. When dealing with images we usually want the origin of the data to be in the center instead. When the data is of even size, the central position can be located at the pixel slightly to the right of the geometrical center. The `ifftshift` shifts the origin in the signal domain from the middle position to the first position as shown in Figure 1. The result from `fft` has the Fourier origin, that is the DC-component, in the first position of the output vector. The `fftshift` command shifts it to the middle position where we expect to find it when we plot the function. Consequently, if f is the function in the signal domain, its Fourier transform is calculated as

```
F = fftshift(fft(ifftshift(f)));
```

and if F is the function in the Fourier domain, its inverse Fourier transform is calculated as

```
f = fftshift(ifft(ifftshift(F)));
```

The 2D case for even sized data is illustrated in Figure 2. Consequently, if f is the function in the spatial domain, its Fourier transform is calculated as

```
F = fftshift(fft2(ifftshift(f)));
```

and if F is the function in the Fourier domain, its inverse Fourier transform is calculated as

```
f = fftshift(ifft2(ifftshift(F)));
```

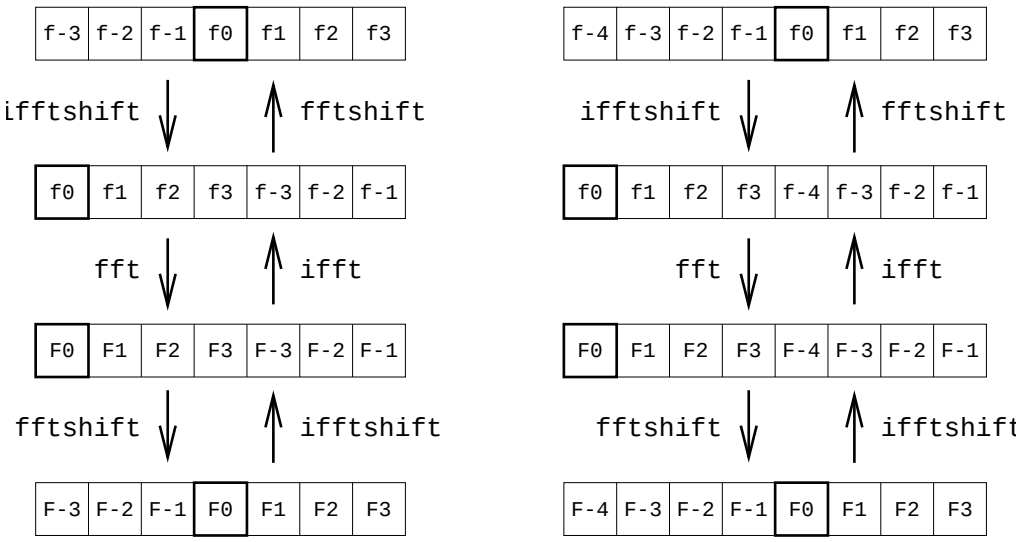


Figure 1: The functions `fftshift` and `ifftshift` are used to place the origin at the correct positions when transforming vectors. Left: odd number of samples. Right: even number of samples.

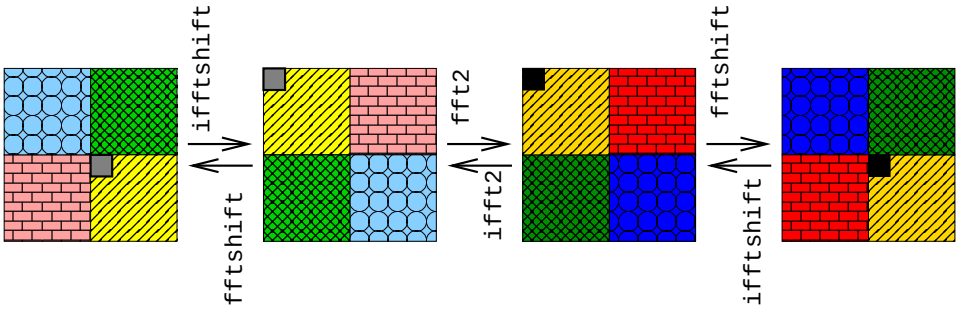
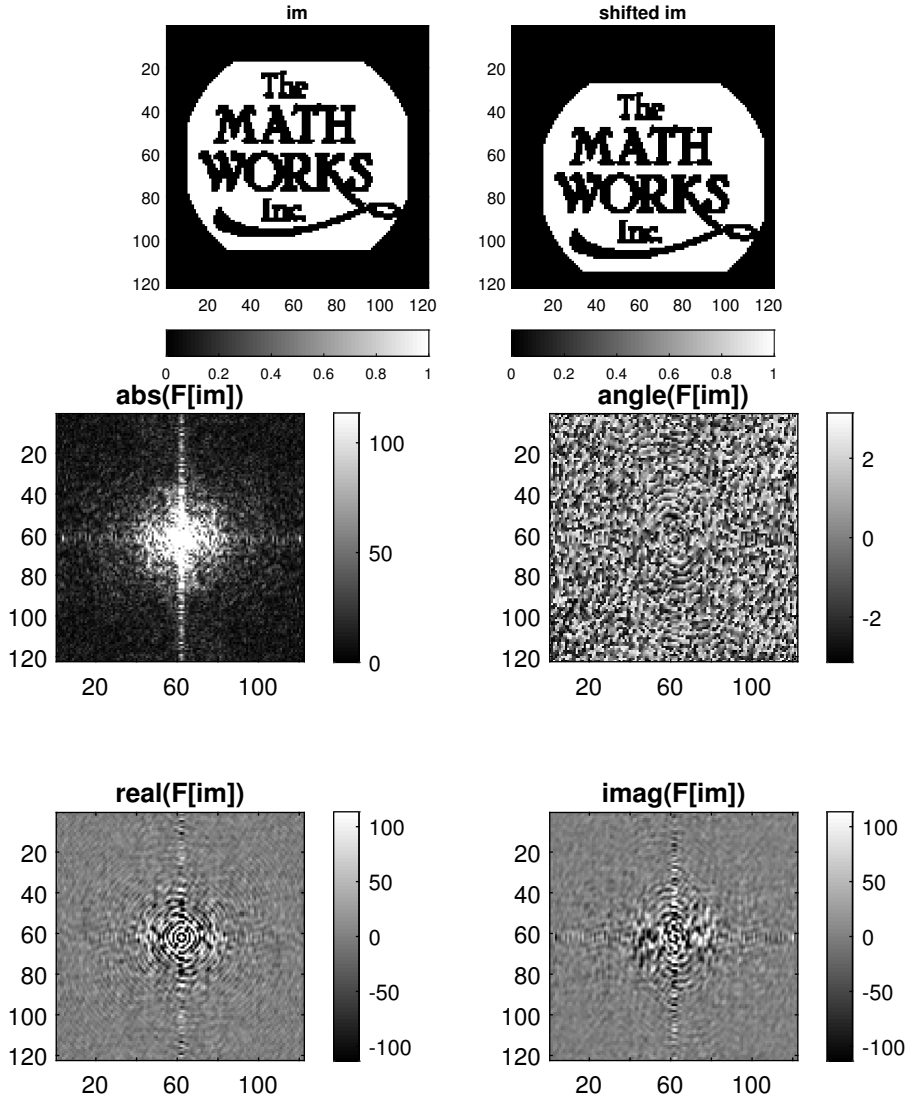


Figure 2: The functions `fftshift` and `ifftshift` are used to place the origin at the correct positions when transforming matrices. Illustrated for even number of samples.

5.2 Analysis of the Fourier transform

Create and execute a file `FFTmathlogo.m` containing the commands below.

```
load math_logo
im = math_logo;
IM = fftshift(fft2(ifftshift(im)));
maxIM = max(max(abs(IM)));
shiftim = circshift(circshift(im,5,2),10,1);
figure(1); colormap gray;
subplot(121); imagesc(im);
axis image; title('im')
colorbar('SouthOutside')
subplot(122); imagesc(shiftim);
axis image; title('shifted im')
colorbar('SouthOutside')
figure(2); colormap gray;
subplot(221); imagesc(abs(IM), [0 0.02*maxIM]);
axis image; colorbar; title('abs(F[im])')
subplot(222); imagesc(angle(IM), [-pi pi]);
axis image; colorbar; title('angle(F[im])')
subplot(223); imagesc(real(IM), [-0.02*maxIM 0.02*maxIM]);
axis image; colorbar; title('real(F[im])')
subplot(224); imagesc(imag(IM), [-0.02*maxIM 0.02*maxIM]);
axis image; colorbar; title('imag(F[im])')
```



EXERCISE II: Different perspectives of the Fourier transform $F(u, v)$ of the image $f(x, y)$ are shown in `figure(2)`. The Fourier transform of a real valued image is Hermitian, i.e. $F(u, v) = F^*(-u, -v)$, where $*$ denotes the complex conjugate. From this property, we can derive a set of symmetry relations. Examples of such symmetries are odd and even functions. Complete the answer below. Discuss with the teacher how the respective symmetries show up in `figure(2)`.

`abs(IM)`: Even, since $|F(u, v)| = |F(-u, -v)|$
`angle(IM)`:
`real(IM)`:
`imag(IM)`:

EXERCISE III: Formulate the translation theorem for 2D signals.

DEMO D: The image `shiftim` to the right in `figure(1)` is shifted with respect to the image `im` to the left. Add code in `FFTmathlogo.m` that calculates the Fourier transform of `shiftim` and shows the results in `figure(3)`.

QUESTION 11: Which of the images `abs( )`, `angle( )`, `real( )` and `imag( )` is not affected by the translation? Is this consistent with the translation theorem?

QUESTION 12: Regard `abs(IM)` in `figure(2)`. The zero frequency component corresponding to $(u, v) = (0, 0)$ is at position (62,62). Click in the image and give its value! Note that it is much higher than the other pixels.

QUESTION 13: Normally this frequency component dominates the frequency spectrum of natural images. It is equal to the sum of all pixel values in the image, which is realized from the DFT formula:
$F[k,l] = \sum_{n=-N/2}^{N/2-1} \sum_{m=-M/2}^{M/2-1} f[n,m] \cdot e^{-j2\pi(nk/N+ml/M)} \Rightarrow$
$F[0,0] = \sum_{n=-N/2}^{N/2-1} \sum_{m=-M/2}^{M/2-1} f[n,m]$.
Verify this (with `sum`) and compare with the previous question!

5.3 Image filtering and the Fourier transform

The filter kernel `aver` with size 3×3 can be implemented in MATLAB. By convolving `aver` by itself two times we obtain the resulting kernel `aver3`.

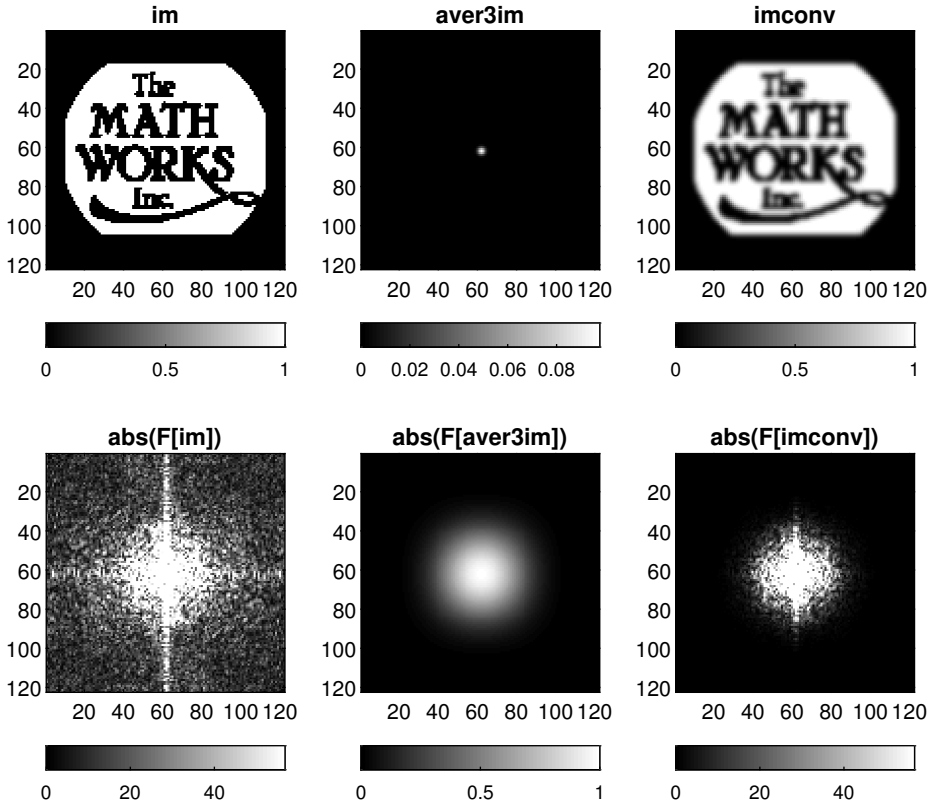
```
aver = [1 2 1; 2 4 2; 1 2 1] /16
aver3 = conv2(conv2(aver,aver,'full'),aver,'full')
```

EXERCISE IV: What is the size of the filter kernel `aver3`?

QUESTION 14: Verify your answer by running the commands above during the lab-exercise. Also note the center value of `aver3`.

Create `FFTmathlogo2.m` with the content below and execute the file. The code generates `figure(1)` containing the original image `im`, an image of the filter kernel `aver3im`, as well as the resulting filtered image `imconv`. The respective Fourier transform of the images in `figure(1)` are shown in `figure(2)`.

```
load math_logo
im = math_logo;
aver = [1 2 1; 2 4 2; 1 2 1]/16;
aver3 = conv2(conv2(aver,aver,'full'),aver,'full');
imconv = conv2(im,aver3,'same');
aver3im = 0*im;
center = size(im,1)/2+1;
aver3im(center-3:center+3, center-3:center+3) = aver3;
IM      = fftshift(fft2(ifftshift(im)));
AVER3IM = fftshift(fft2(ifftshift(aver3im)));
IMCONV  = fftshift(fft2(ifftshift(imconv)));
maxIM   = max(max(abs(IM)));
figure(1); colormap gray;
subplot(131); imagesc(im);
axis image; title('im'); colorbar('SouthOutside')
subplot(132); imagesc(aver3im);
axis image; title('aver3im'); colorbar('SouthOutside')
subplot(133); imagesc(imconv);
axis image; title('imconv'); colorbar('SouthOutside')
figure(2); colormap gray;
subplot(131); imagesc(abs(IM), [0 0.01*maxIM]);
axis image; title('abs(F[im])'); colorbar('SouthOutside')
subplot(132); imagesc(abs(AVER3IM));
axis image; title('abs(F[aver3im])'); colorbar('SouthOutside')
subplot(133); imagesc(abs(IMCONV), [0 0.01*maxIM]);
axis image; title('abs(F[imconv])'); colorbar('SouthOutside')
```



QUESTION 15: Zoom in on the image `aver3im` and verify that it contains the small filter kernel. Click on the center value and check if it is correct. What is the correct value?

EXERCISE V: What is the convolution theorem for 2D signals?

QUESTION 16: Analyze the Fourier transforms in `figure(2)` and describe how they fulfill the convolution theorem.

We will now relate properties in the spatial domain (image) to the frequency domain (Fourier transform). The zero frequency $(u, v) = (0, 0)$ is placed at coordinates (62,62) in the Fourier transform image, as we previously concluded. Closer to the edges in the Fourier transform image, $\sqrt{u^2 + v^2}$ becomes larger i.e. the frequencies increase.

QUESTION 17: Complete the sentences with *low frequencies* or *high frequencies* in the boxes below. Your answers should be consistent with the

images `im`, `imconv` and the Fourier transforms `IM`, `IMCONV`.

A normal image is dominated by smooth surfaces with different intensities. This corresponds to [*frequencies*] *i the Fourier transform image. The image* `im` *has both smooth surfaces and sharp edges. The sharp edges corresponds to* [*frequencies*] *in the Fourier transform image. The image* `imconv` *has both smooth surfaces and soft edges. Consequently, the* [*frequencies*] *are suppressed in the Fourier transform image.*

6 Convolution with derivative filters

6.1 Derivatives in the $x-$ and $y-$direction, gradient

The filter `d`= [1 | -1] calculates the finite difference in the $x-$direction. In order to avoid a shift by $1/2$ pixels, the central difference `cd`= [1 | [0] | -1]$/2$ is more commonly used. It applies that

$$\texttt{cd} = \texttt{d} * \texttt{b} = \boxed{1 \,\|\, \text{-}1} * \boxed{1 \,\|\, 1}/2 = \boxed{1 \mid [0] \mid \text{-}1}/2$$

The derivative in the x-direction (horizontal) can be calculated as

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

Create the file `DeriveCircle.m` containing the code below and execute it.

```
im = double(imread('circle.tif'));
figure(1)
colormap(gray(256))
subplot(2,2,1), imagesc(im, [0 255])
axis image; axis off; title('original image'); colorbar
cd = [1 0 -1]/2;
imdx = conv2(im,cd,'same');
subplot(2,2,3), imagesc(imdx, [-128 127])
axis image; axis off; title('imdx image'); colorbar
```

The derivative in the y-direction (vertical), $\frac{\partial f(x,y)}{\partial y}$, can be calculated in a similar way as the derivative in the x-direction, namely as

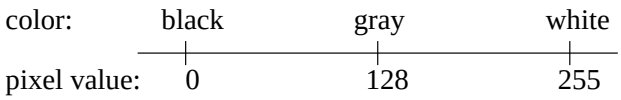
$$\frac{\partial f(x,y)}{\partial y} = \frac{\partial}{\partial y} * f(x,y) \approx \mathcal{R}_{90^\circ}\{\texttt{cd}\} * f(x,y).$$

EXERCISE VI: Construct $\mathcal{R}_{90^\circ}[\texttt{cd}]$ below.

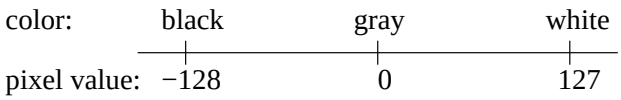
$$\mathcal{R}_{90^\circ}[\texttt{cd}] = \mathcal{R}_{90^\circ}[\texttt{d}] * \mathcal{R}_{90^\circ}[\texttt{b}] = \begin{array}{|c|}\hline\ \\ \hline\ \\ \hline\end{array} * \begin{array}{|c|}\hline\ \\ \hline\ \\ \hline\end{array}/2 = \begin{array}{|c|}\hline\ \\ \hline\ \ \\ \hline\ \\ \hline\end{array}/2$$

DEMO E: Add code in `DeriveCircle.m` for displaying the approximated derivative of the circle in the y-direction. Show the resulting image down to the right.

When an image contain positive values from 0 to 255, the resulting color table **gray** works as follows:



The images containing approximate derivatives contain negative values. Whenever an image only contain values within the interval [-128 127], the resulting color table **gray** works as follows:



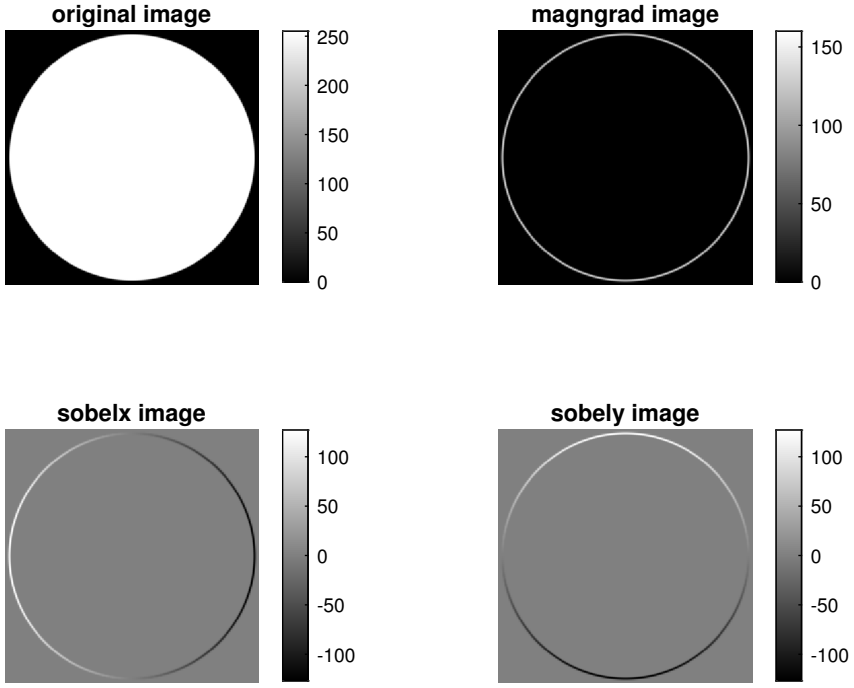
In conclusion, negative values are dark while positive values are bright. Values close to zero are gray.

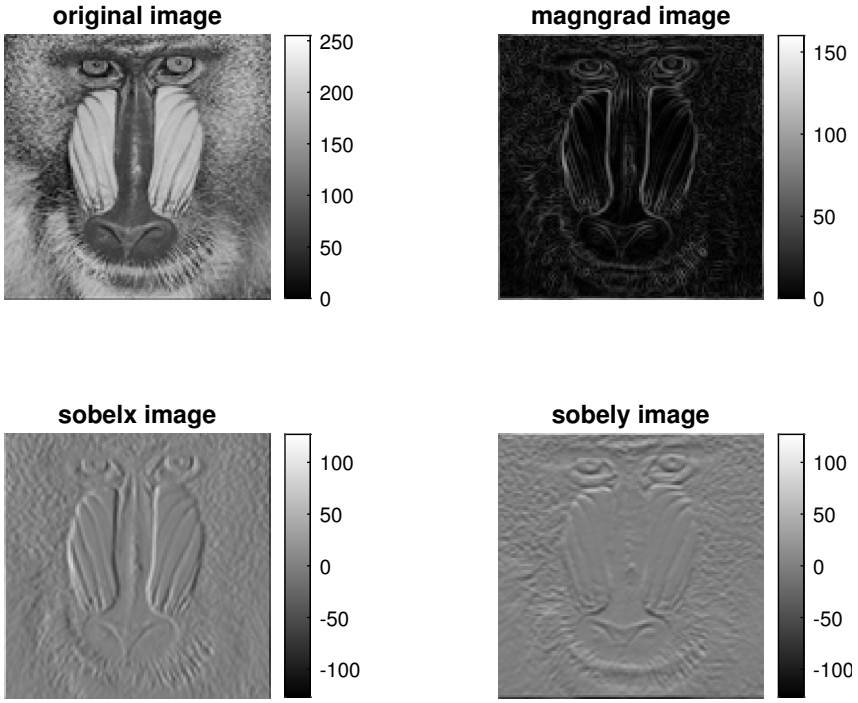
QUESTION 18: Look at the edges in the original image and the corresponding result in the derivative images. Explain why the derivative images sometimes are dark and sometimes bright.

The gradient $\left(\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y}\right)$ is a two dimensional vector pointing at the direction where the intensity level in the image has its fastest rate of change.

QUESTION 19: Write down the mathematical expression for the magnitude (length) of the gradient of the image $f(x, y)$!

DEMO F: Add code in `DeriveCircle.m` such that the magnitude of the gradient is shown up to the right. Write down a similar program for the monkey, `DeriveMonkey.m` and examine the images. Note that the magnitude of the gradient is an edge detector!



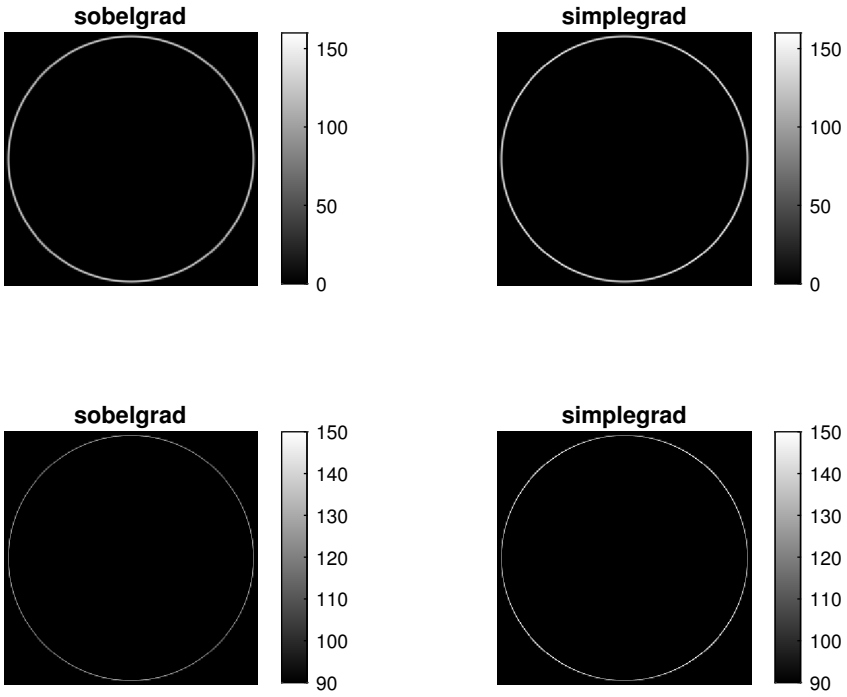


For the circle, the edge is equally steep all around. Hence, also the magnitude of the gradient should be equally large around the circle. However, this is not the case for the gradient magnitude produced by convolving the image with the central difference filters. In this sense, the Sobel filters are more suitable for approximating the derivative of the images, see below.

$$\texttt{sobelx} = \texttt{cd}* \begin{array}{|c|}\hline 1 \\ \hline [2] \\ \hline 1 \\ \hline \end{array} /4 = \begin{array}{|c|c|c|}\hline 1 & 0 & \text{-}1 \\ \hline 2 & [0] & \text{-}2 \\ \hline 1 & 0 & \text{-}1 \\ \hline \end{array} /8,$$

$$\texttt{sobely} = \mathcal{R}_{90^\circ}[\texttt{cd}]* \begin{array}{|c|c|c|}\hline \text{-}1 & [\text{-}2] & \text{-}1 \\ \hline \end{array} /4 = \begin{array}{|c|c|c|}\hline \text{-}1 & \text{-}2 & \text{-}1 \\ \hline 0 & [0] & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} /8.$$

DEMO G: Copy your code in `DeriveCircle.m` to `MagnitudeSimpleSobel.m`. Add code for the sobel filters. The idea is now to visualize the difference between the simple central difference filters and the sobel filters when they are used to calculate the gradient. To magnify the difference you can change the contrast window. Produce similar images as below.



6.2 The Laplace filter, a highpass-filter (with negative sign)

The Laplace operator is defined as

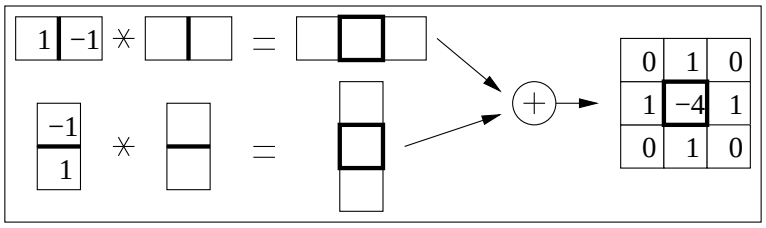
$$\nabla^2 = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) = \frac{\partial}{\partial x} * \frac{\partial}{\partial x} + \frac{\partial}{\partial y} * \frac{\partial}{\partial y}$$

We will now apply the derivative operators $\mathtt{d}$ and $\mathcal{R}_{90^\circ}[\mathtt{d}]$, in x and y, respectively,

$$\frac{\partial}{\partial x} \approx \mathtt{d} = \boxed{1}\,\boxed{\boxed{-1}}, \quad \frac{\partial}{\partial y} \approx \mathcal{R}_{90^\circ}[\mathtt{d}] = \begin{array}{|c|}\hline -1 \\ \hline\hline 1 \\ \hline \end{array}.$$

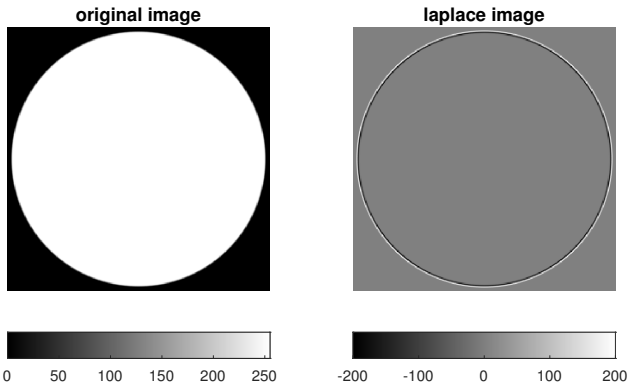
where the origin is marked by double lines.

EXERCISE VII: Build a Laplace filter by filling in the fields below. The Laplace filter itself is given. Its origin is marked by a bold frame. For the derivative filters their origin is between the pixels.

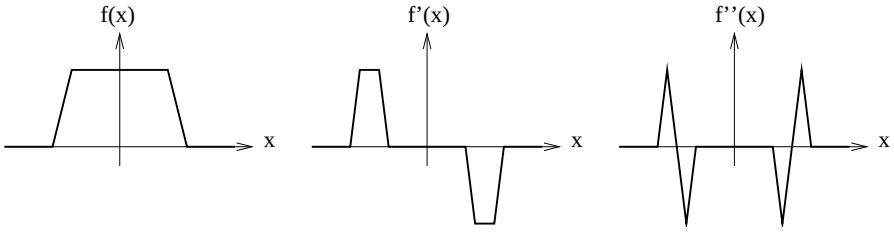


Generate a file `LaplaceCircle.m` with the following contents and execute it.

```
im = double(imread('circle.tif'));
figure(1)
colormap(gray(256))
subplot(1,2,1), imagesc(im, [0 255])
axis image; axis off; title('original image')
colorbar('SouthOutside')
laplace = [0 1 0; 1 -4 1; 0 1 0];
imlaplace = conv2(im, laplace, 'same');
subplot(1,2,2), imagesc(imlaplace, [-200 200])
axis image; axis off; title('Laplace image')
colorbar('SouthOutside')
```



The Laplace filter estimates a certain 2-D second derivative. The 1-D second derivative of an approximative 1-D rectangular function $f(x)$ is shown below.

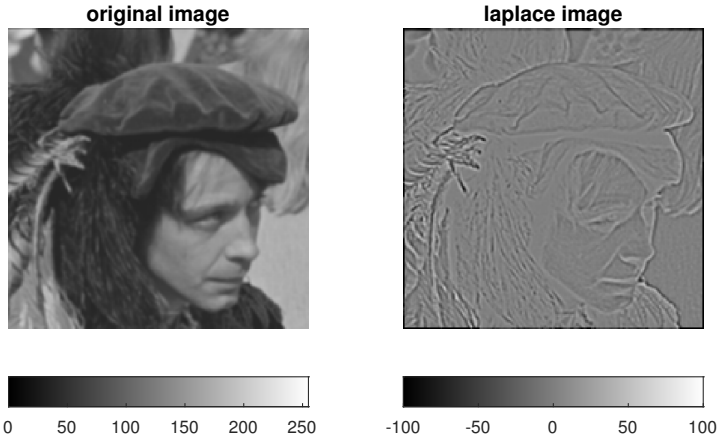


EXERCISE VIII: Does the Laplace image (`imlaplace`) agree with the plot above? Motivate your response!

Generate a file `SharpenPirate.m` with the same contents as `LaplaceCircle.m`. Replace the first lines with:

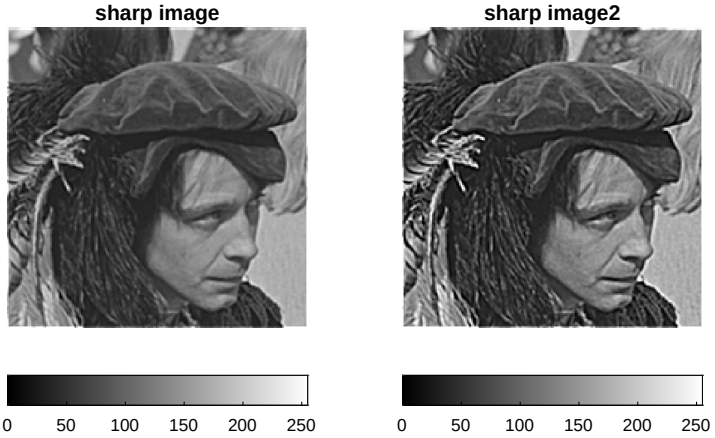
```
load('pirat.mat')
im = pirate;
```

You will also have to adjust the bounds for displaying the contrast interval to `[-100 100]`.



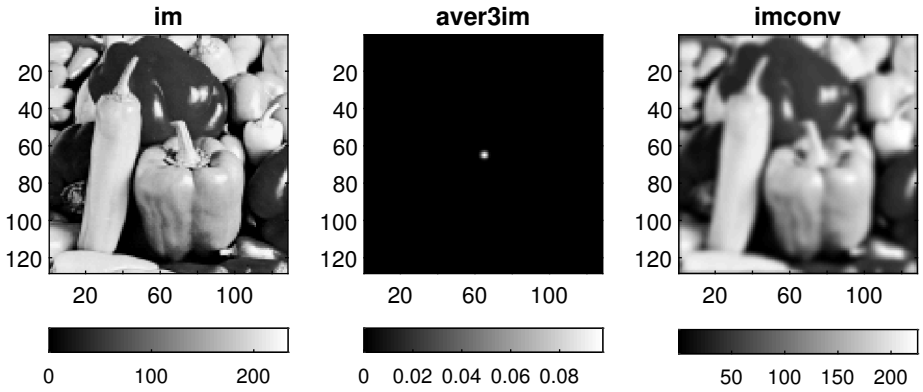
EXERCISE IX: Look at the Laplace image (`imlaplace`) and compare it to the original image (`im`). What happens in smooth regions (e.g. cheek)? What happens at edges? What is the effect in highly dynamic regions (e.g. feathers)?

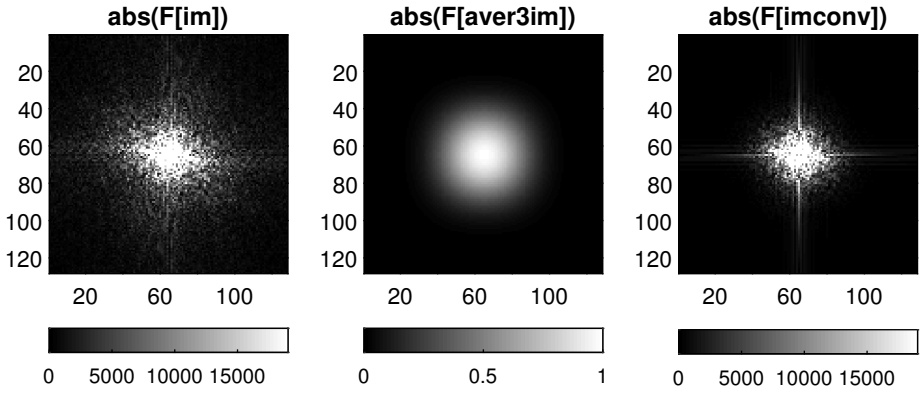
DEMO H: The Laplace filter is a highpass-filter with negative sign. Therefore, form `imHP = -imlaplace;` Then complete `SharpenPirate.m` with an image which is the sum of the original image and the high-pass filtered image, i.e. `imsharp = im + imHP`; Also make `imsharp2 = im + 2*imHP`; Note that the edges become sharper with increased amount of `imHP`. Also note that the image noise is amplified.



7 Further Exercises

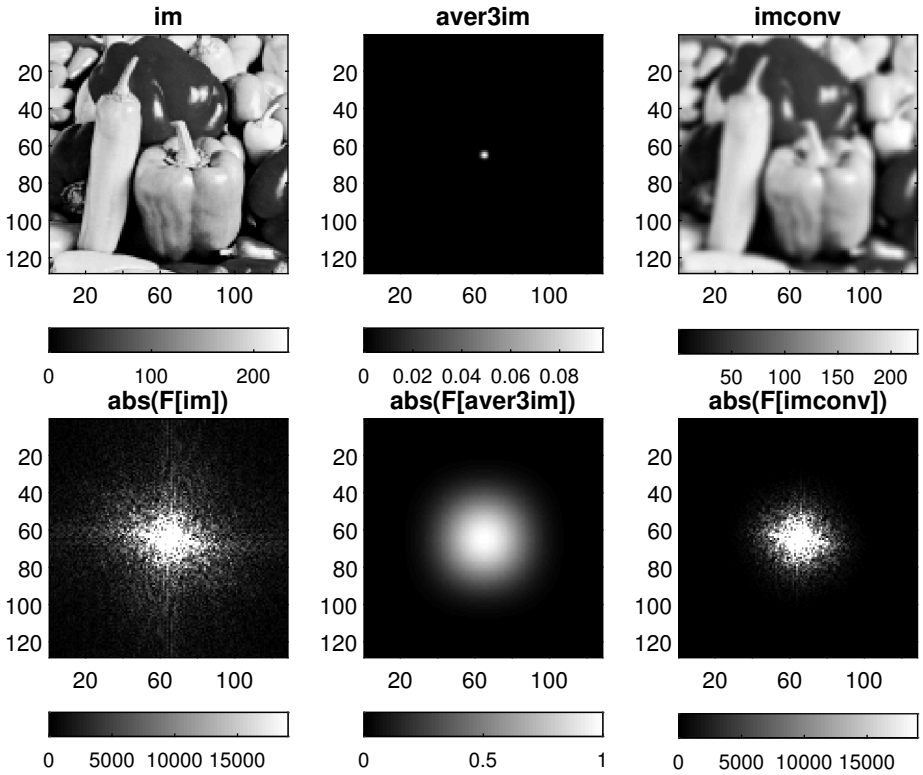
DEMO I: Copy the file `FFTmathlogo2.m` to `FFTpeppers.m` and replace the image `math_logo` with `peppersmall`.





QUESTION 20: Note that the dark frame of `imconv` gives rise to a "cross" in the Fourier domain. Why does `imconv` contain a dark frame?

DEMO J: The dark frame can be avoided in several different ways. For instance, values outside the image can be extrapolated before the convolution. The convolution can also be executed via multiplication in the Fourier domain, as stated by the convolution theorem. Implement this approach in `FFTpeppers2.m`. Reuse what you have been achieved in `FFTpeppers.m`.

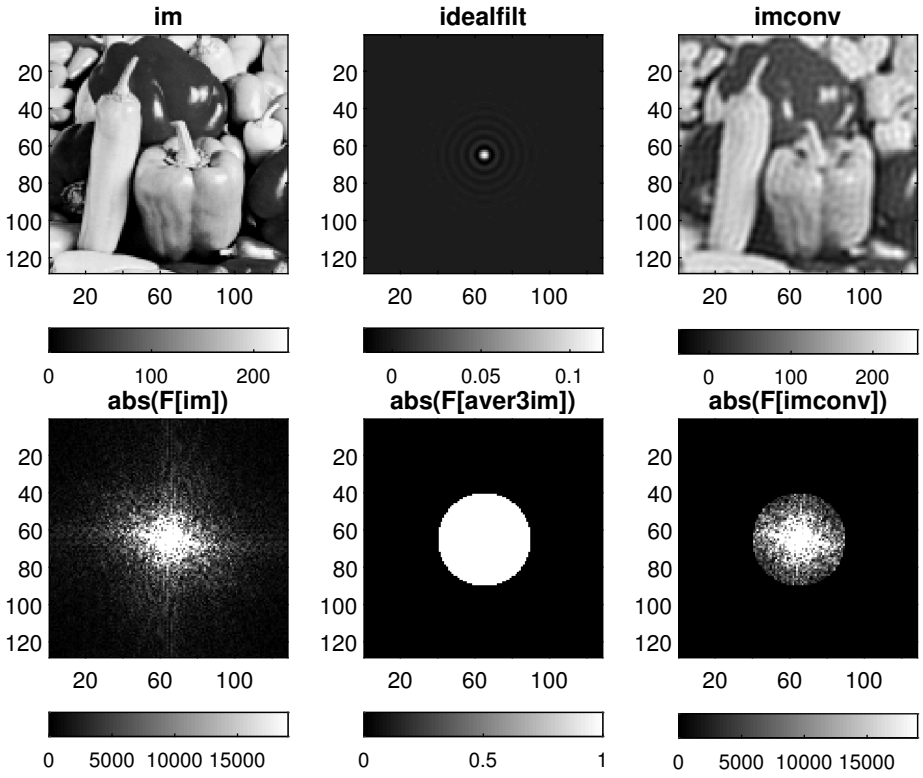


QUESTION 21: Compare the new `imconv` image with the one from `FFTpeppers.m`. Note that the interior of the images are exactly the same. The dark frame is gone. Instead, the right edge of the image is influenced by the left edge and vice versa. In addition, the top of the image is influenced by the bottom of the image and vice versa. What is this kind of convolution called?

DEMO K: If the convolution is performed in the Fourier domain anyway, one might like to apply an ideal lowpass filter. Copy `FFTpeppers2.m` into `FFTpeppers3.m`. An ideal circular lowpass filter can be generated by:

```
half = (size(im,1)-1)/2;
[u,v] = meshgrid(-ceil(half):floor(half),-ceil(half):floor(half));
IDEALFILT = sqrt(u.^2+v.^2) < myradius;
```

Choose a suitable value for `myradius` such that `IDEALFILT` is as similar to `AVER3IM` as possible. Verify that the previous lowpass filtered image `imconv` and the new lowpass filtered image look similarly blurry. Note that the new lowpass filtered image contains a distortion that is the 2-D counterpart of the Gibb's phenomenon containing oscillations and stripes.



Extra

8 Mesh plots of the Fourier transform of filter kernels

Finally we will plot the Fourier transform of some common filter kernels. You will later be able to calculate their Fourier transforms. Now, you can just check the answers in lesson 1. See the three filter kernels below.

a) $\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & \mathbf{4} & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$ /16, b) $\begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & \mathbf{-4} & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}$, c) $\begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & \mathbf{0} & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array}$ /8

QUESTION 22: Write down their Fourier transforms!

a) $F(u,v) = (1+\cos(2\pi\Delta u))/2 \cdot (1+\cos(2\pi\Delta v))/2$

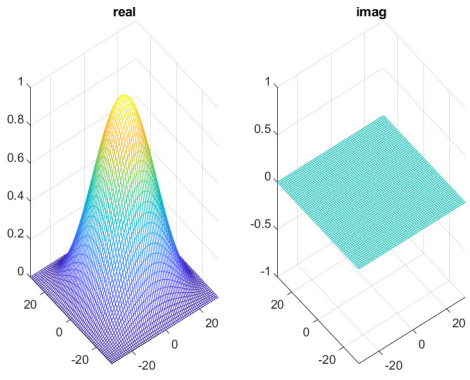
b) $F(u,v) = ...$

c) $F(u,v) = ...$

The relationship between the sampled Fourier transform $F(u,v)$ and the DFT $F(k,l)$ is $\Delta u = k/N$, $\Delta v = l/M$. Create `meshplot.m` with the content found below and execute the file.

```
N=64;
averim = zeros(N,N);
center = size(averim,1)/2+1;
aver = [1 2 1; 2 4 2; 1 2 1] /16
averim(center-1:center+1, center-1:center+1) = aver;
AVERIM = fftshift(fft2(ifftshift(averim)));
figure(1);
subplot(1,2,1), mesh(-N/2:N/2-1, -N/2:N/2-1, real(AVERIM));
subplot(1,2,2), mesh(-N/2:N/2-1, -N/2:N/2-1, imag(AVERIM));
```

A a mesh plot of the DFT of filter kernel a) is shown, see below.



DEMO L: Extend `meshplot.m` so that it shows meshplots of the DFT of filter kernel b) and c) also. Be sure that they agree with the equations given above.

---

9 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) $\approx$ (63,17), index $\approx$ 84.
c) (X,Y) $\approx$ (23,58), index $\approx$ 92.

ANSWER 3:
a) `mycolormap0` has values between 0 and 100 instead of between 0 and 255.
b) `mycolormap0` has values between 0 and 10 instead of between 0 and 255.
c) `mycolormap0` 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) Data outside the image are regarded as zeros.
b) Data outside the image are extrapolated before convolution.
c) Output pixels close to the border are not affected, since the filter kernel is then partly outside the input image.

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) When using a lower value, the resulting image gets brighter.
b) When using a lower value, the resulting image gets darker.
c) When using a lower value, the resulting image gets better contrast.

ANSWER 10:
a) It returns only those parts of the convolution that are computed without the zero-padded edges.
b) The output image gets the same size as the input image.
c) Output pixels close to the border are not affected, since the filter kernel is then partly outside the input image.

ANSWER 11:
a) `real` does not change. Yes, because $\texttt{real}(\mathcal{F}[f(x-a,y-b)]) =$
$= \texttt{real}(e^{-j2\pi(au+bv)}F(u,v)) = \texttt{real}(F(u,v))$.
b) `abs` does not change. Yes, because $|\mathcal{F}[f(x-a,y-b)]| =$
$= |e^{-j2\pi(au+bv)}F(u,v)| = |e^{-j2\pi(au+bv)}||F(u,v)| = |F(u,v)|$.
c) `imag` does not change. Yes, because $\texttt{imag}(\mathcal{F}[f(x-a,y-b)]) =$
$= \texttt{imag}(e^{-j2\pi(au+bv)}F(u,v)) = \texttt{imag}(F(u,v))$.

ANSWER 12:
a) The zero frequency component has the value 5679.
b) The zero frequency component has the value 6679.
c) The zero frequency component has the value 7679.

ANSWER 13:
a) 5679
b) 6679
c) 7679

ANSWER 14:
a) 0.07766
b) 0.08766
c) 0.09766

ANSWER 15:
a) 0.07766
b) 0.08766
c) 0.09766

ANSWER 16:
a) The images make it likely that: `IM` + `AVER3IM` = `IMCONV`.
b) The images make it likely that: `IM` $*$ `AVER3IM` = `IMCONV`.
c) The images make it likely that: `IM` $\cdot$ `AVER3IM` = `IMCONV`.

ANSWER 17:
a) high, high, low
b) high, low, high
c) low, high, high

ANSWER 18:
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 grey pixel.

ANSWER 19:
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 20:
a) Data outside the image are extrapolated before convolution.
b) When the filter kernel is somewhat outside the image, the result is set to 0.
c) Data outside the image are regarded as zeros.

ANSWER 21:
a) Border convolution.
b) Linear convolution.
c) Circular convolution.

10 Examination

EXERCISES

Home exercises approved by teacher? OK! Maria

DEMONSTRATIONS

Demonstrations approved by teacher? Covered to ex 19, demo E

QUESTIONS

	a	b	c
1:	X		
2:		X	
3:			X
4:			X
5:			X
6:		X	
7:	X		

	a	b	c
8:			X
9:	X		
10:	X		
11:		X	
12:	X		
13:	X		
14:			X

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

Questions approved by teacher? OK! Emanuel
