Image Analysis - Lecture 3

Magnus Oskarsson

Lecture 3

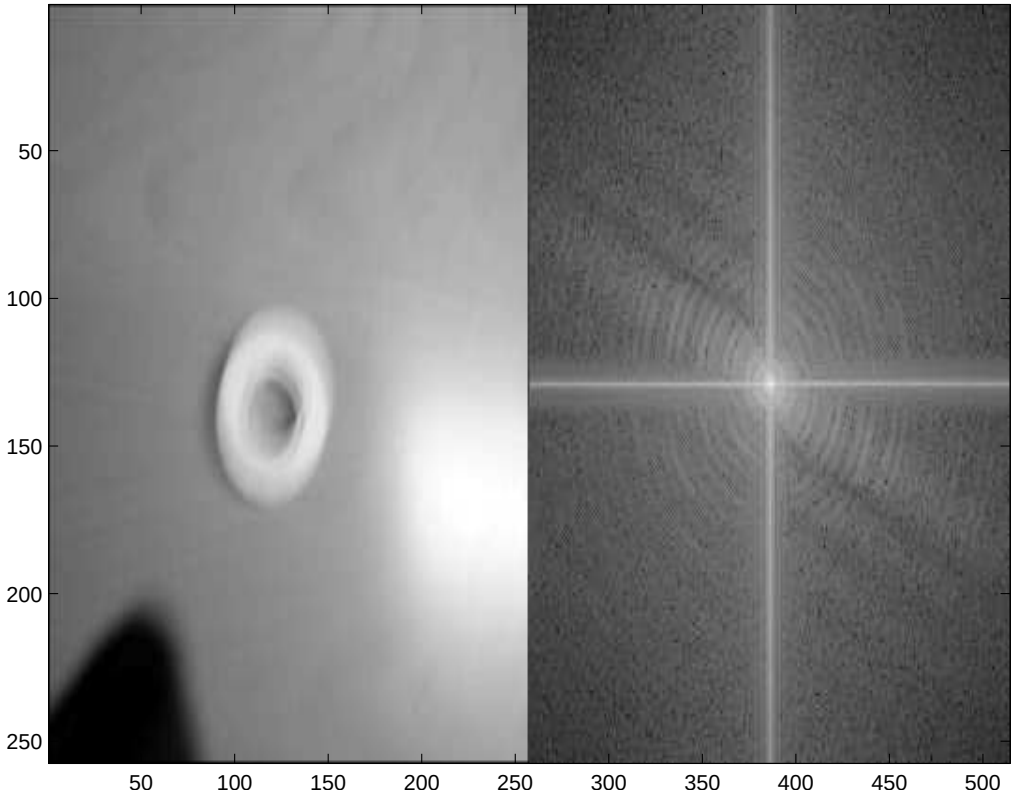
Contents

- Understanding the Fourier transform
  - Effects due to periodic continuation
  - Effects due to edges/structure in the images
- Filters and convolution
  - Low pass-, high pass-, median-, mean value-, differentiation-filters
  - Scale-space
  - Finding patterns

Understanding the Fourier transform

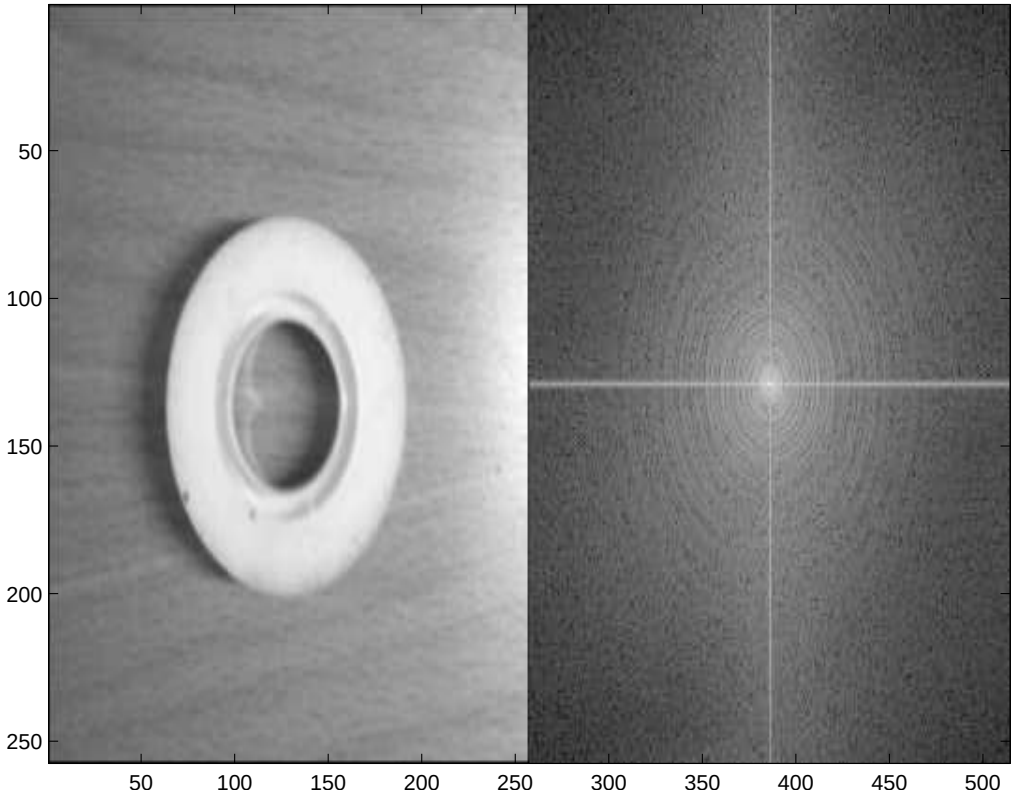
- After fftshift low frequencies are located in the centre of the FFT image
- High frequencies are located at the borders of the FFT image
- The fact that images are considered as periodic functions, the borders of the images can influence the FFT
- High frequencies in one direction (e.g. edges) can be seen as high intensities in the FFT image in the same direction.

Image b and log(abs(fftshift(fft2(b))))



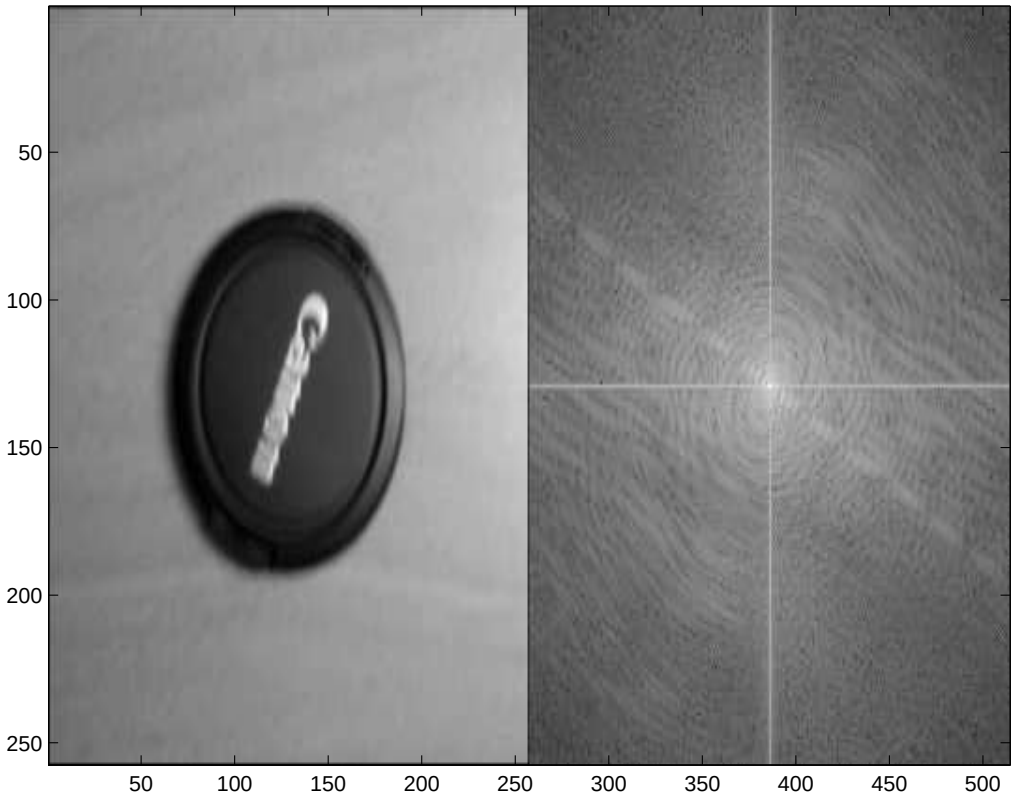
Observe the boundary effects!

Image b and log(abs(fftshift(fft2(b))))



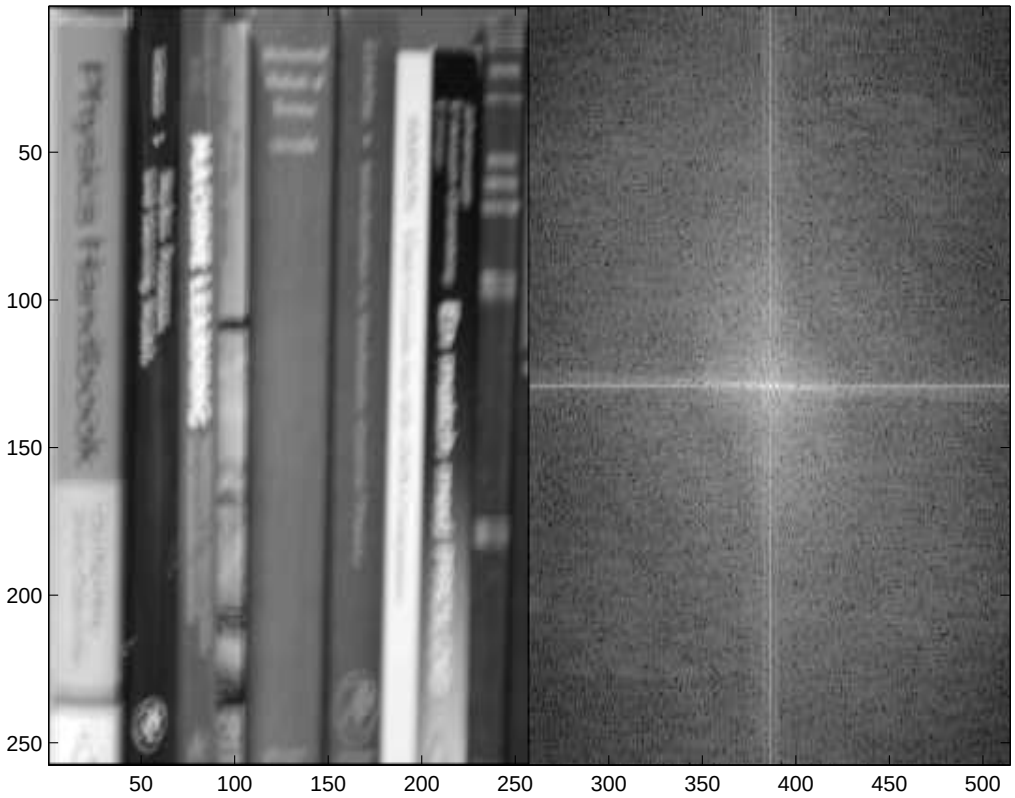
Smaller boundary effects due to more uniform background.

Image b and log(abs(fftshift(fft2(b))))



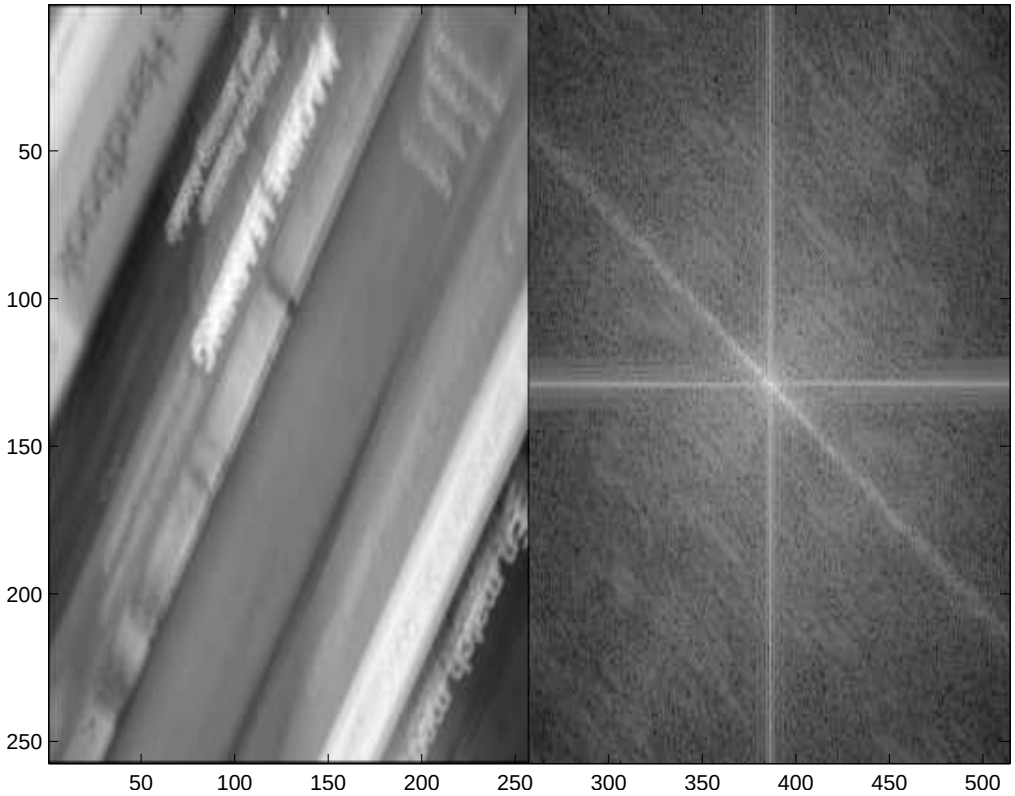
Sharp edges in the image gives high frequencies in the FFT image.

Image b and log(abs(fftshift(fft2(b))))



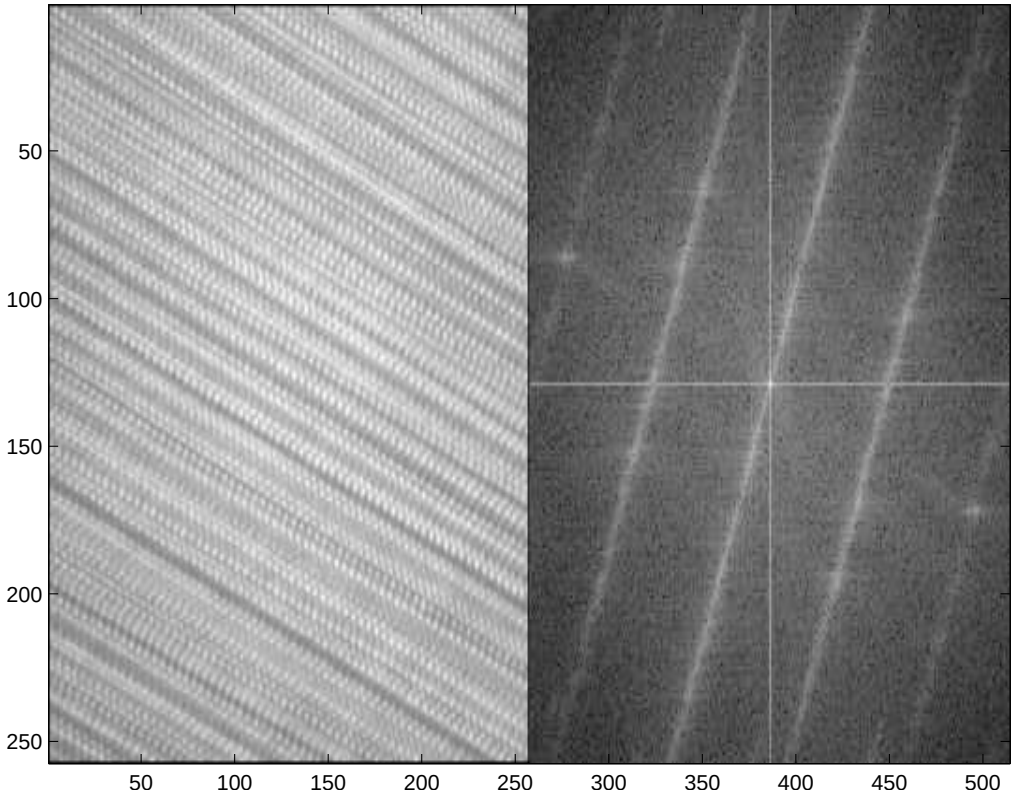
Sharp edges in the image gives high frequencies in the FFT image.

Image b and log(abs(fftshift(fft2(b))))



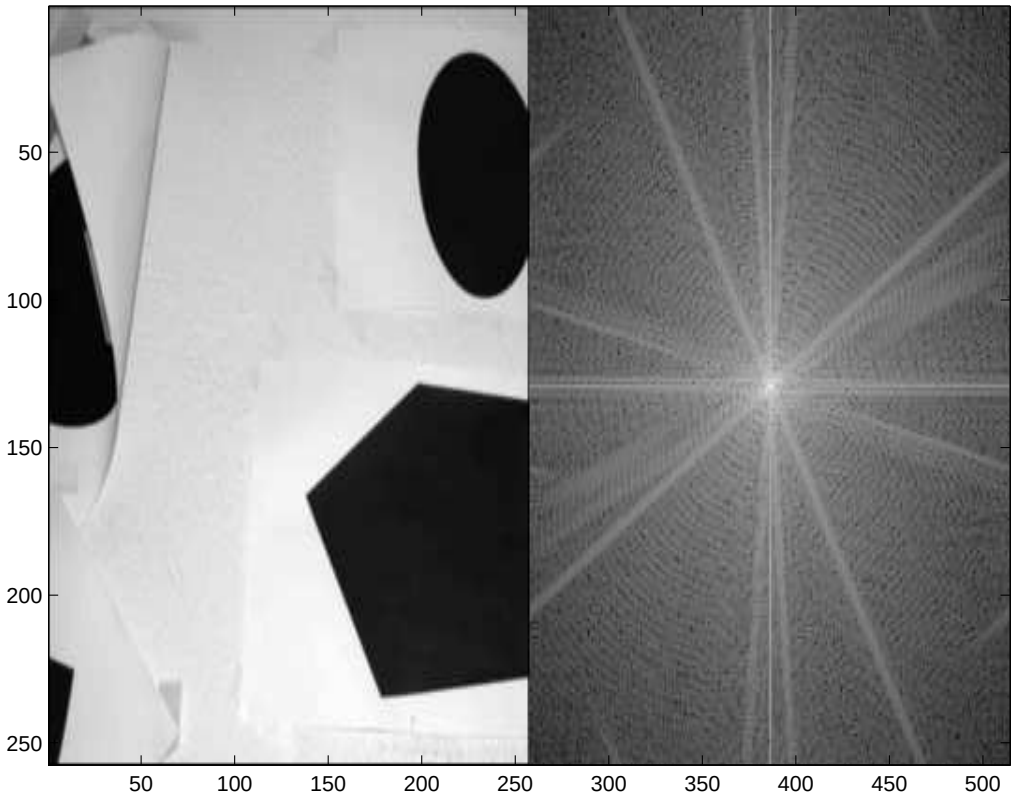
Observe the directions of the edges and the high frequency components.

Image b and log(abs(fftshift(fft2(b))))



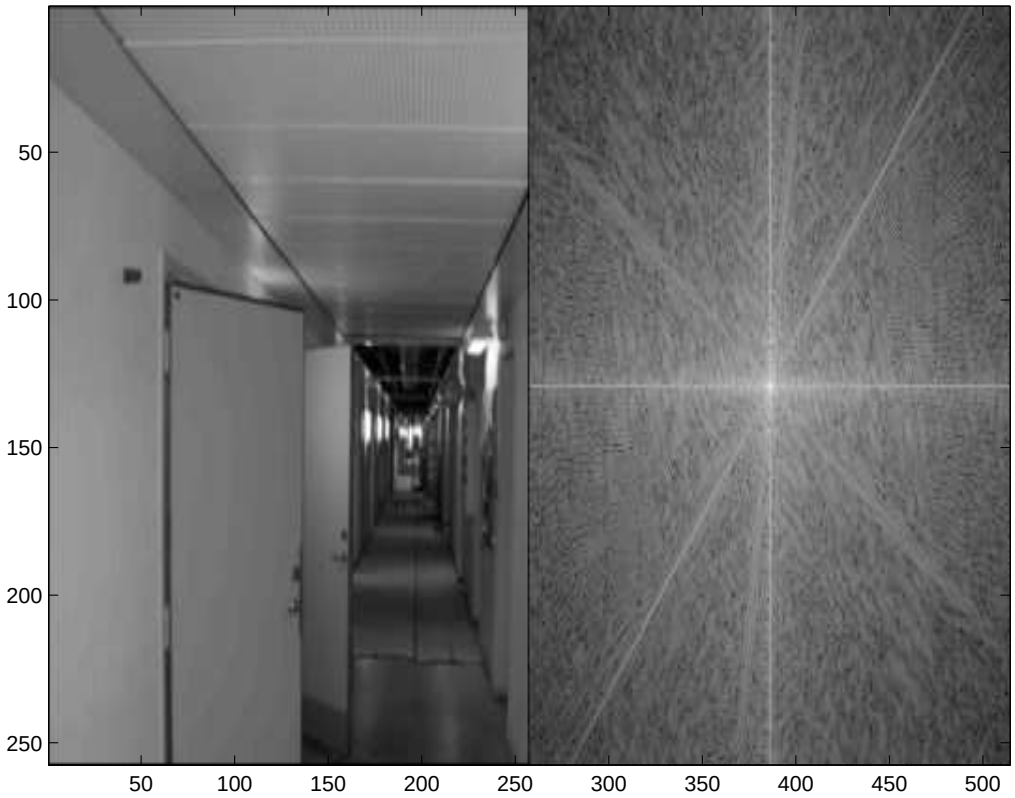
Several edges can give rise to several high frequency components.

Image b and log(abs(fftshift(fft2(b))))



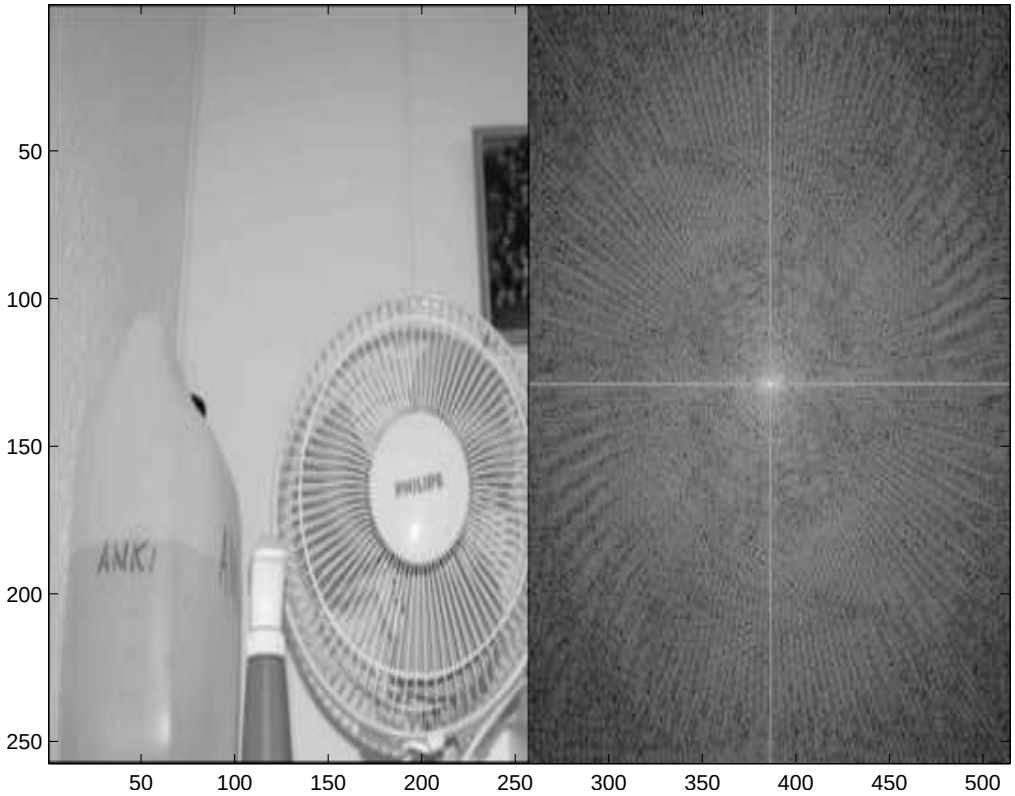
Identify the different edge directions and the corresponding high frequency components.

Image b and log(abs(fftshift(fft2(b))))



Identify the different edge directions and the corresponding high frequency components.

Image b and log(abs(fftshift(fft2(b))))



The pattern in the fan gives rise to several high frequency components.

One-dimensional convolution

The (one-dimensional) convolution of two functions f and g is defined as follows:

Definition

Let f and g be functions from $\mathbb{R}$ to $\mathbb{R}$.

$$\textbf{convolution: } (f * g)(x) = \int f(\alpha) g(x - \alpha) d\alpha$$

■

Observe that

$$f * g = g * f \quad .$$

Two-dimensional convolution

The (two-dimensional) convolution of two functions f and g is defined as follows:

Definition

Let f and g be functions from $\mathbb{R}^2$ to $\mathbb{R}$.

$$\textbf{convolution: } (f * g)(x, y) = \int f(\alpha, \beta) g(x - \alpha, y - \beta) d\alpha d\beta$$

■

In Forsythe-Ponce the notation $f **g$ is used for 2D convolution.

Example: Convolution

One-dimensional convolution:
Graduated engineers from XTH. Assume that each autumn year x one admits f students to XTH. Estimates have shown that the number of students that graduate are 5% in year $x + 3$, 25% in year $x + 4$, 50% in year $x + 5$ and 20% in year $x + 6$.

One-dimensional convolution:

$$h = f * g$$

$$h(x) = \sum_{\alpha} f(\alpha) g(x - \alpha)$$

Example: Convolution (ctd.)

Assume that 100 are admitted in year 2006 and 200 in year 2007 (but none before 2006 and none after 2007). How many graduate different years

year	2009	2010	2011	2012	2013
from 2006	5	25	50	20	0
from 2007	0	10	50	100	40
total	5	35	100	120	40

Example: Convolution (ctd.)

If you want to calculate how many graduate a certain year (e.g. year 2012), one flips g

year	2006	2007	2008	2009	2010	2011	2012
f	100	200	0	0	0	0	0
flipped g	0.2	0.5	0.25	0.05	0	0	0
$f(\alpha)g(x-\alpha)$	20	100	0	0	0	0	0

$$h(2012) = \sum_{\alpha} f(\alpha)g(2012-\alpha) = 120$$

The position of the zero'th element in sequences

It is important to know where position zero in sequences are. Compare with decimal-comma in reals.

$$12 \cdot 12 = 144, 1.2 \cdot 12 = 14.4$$

What notations should we use?

Suggestion: underline the element at position zero, e.g.

$$< 1, \underline{1} > * < 1, \underline{2}, 1 >=< 1, 3, \underline{3}, 1 >$$

but

$$< 1, \underline{1} > * < \underline{1}, 2, 1 >=< 1, \underline{3}, 3, 1 >$$

and

$$< \underline{1}, 1 > * < \underline{1}, 2, 1 >=< \underline{1}, 3, 3, 1 >$$

Edge effects

In practice we do not have infinite images.
How should we treat the edges of the image? What values should one assume 'outside' the image.
Some common choices are

1. Only calculate the result where we can be certain. The result is a smaller image.
2. Assume that there are zeros outside the image. This often means that we introduce artificial sharp edges at the border.
3. Make a periodic expansion of the image, i.e. assume that the image is periodic. This fits well with the theory for discrete fourier transform.

Example: Convolution of finite images

Assume that one would like to convolve the image

$$f = \begin{bmatrix} \underline{1} & 2 & 3 & 5 \\ 1 & 3 & 2 & 1 \\ 2 & 2 & 2 & 2 \end{bmatrix}$$

with the smoothing filter

$$h = \begin{bmatrix} \underline{1} & 1 \\ 1 & 1 \end{bmatrix}$$

Example: Convolution of finite images (ctd.)

(1) Don't let h extend outside f

$$\begin{bmatrix} 7 & 10 & 11 \\ 8 & 9 & 7 \end{bmatrix}$$

(2) Extend with zeros $\Rightarrow$ equal or larger resulting $h * f$-image

$$\begin{bmatrix} \underline{1} & 3 & 5 & 8 & 5 \\ 2 & 7 & 10 & 11 & 6 \\ 3 & 8 & 9 & 7 & 3 \\ 2 & 4 & 4 & 4 & 2 \end{bmatrix}$$

(3) Extend f and h to periodic functions with the same period: f_p, $h_p \Rightarrow$ periodic $h_p * f_p$ result with same period

$$\begin{bmatrix} \underline{10} & 7 & 9 & 12 \\ 8 & 7 & 10 & 11 \\ 6 & 8 & 9 & 7 \end{bmatrix}$$

Here we have also made a periodic function of h:

$$h = \begin{bmatrix} \underline{1} & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} .$$

Example: Convolution and applying a mask

Let h be a filter represented by a matrix. Define $\check{h}$ as the mirrored version of h, e.g.

$$h = \begin{bmatrix} \underline{1} & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad \check{h} = \begin{bmatrix} \underline{1} & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

Then the convolution of f and h can be calculated as a masking operation:

$$g(i,j) = \sum_{u,v} f(u,v)h(i-u,j-v) =$$

$$= \sum_{u,v} f(u,v)\check{h}(i+u,i+v) = \sum_{u,v} f(i+u,j+v)\check{h}(u,v) \quad . \quad (1)$$

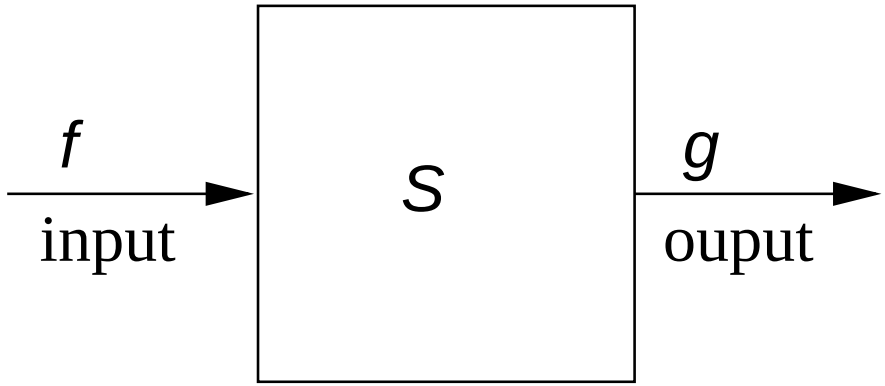
Two-dimensional periodic convolution

Periodic continuation gives:

$$g(i,j) = \sum_{u,v} f(u,v)h(i-u,j-v)$$

$$\begin{pmatrix} 1 & 1 & 2 \\ 1 & \underline{1} & 2 \\ 1 & 2 & 2 \end{pmatrix} * \begin{pmatrix} \underline{1} & -1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 1 \\ -1 & \underline{0} & 1 \\ -1 & 1 & 0 \end{pmatrix}$$

Convolution as image transform



S is called a **system** or **filter**.

- S is **linear** if $S(\lambda_1 f_1 + \lambda_2 f_2) = \lambda_1 S(f_1) + \lambda_2 S(f_2)$. Implies that

$$g(x) = \int h(x,y) f(y) dy \quad ,$$

  where h called the **impulse response**.
- S is **translation invariant** if $S(f(x)) = g(x) \quad \Rightarrow \quad S(f(x-a)) = g(x-a)$.

Linear translational invariant systems

Any linear and translation invariant system can be represented by

$$g(x) = \int h(x-y)f(y)dy \quad .$$

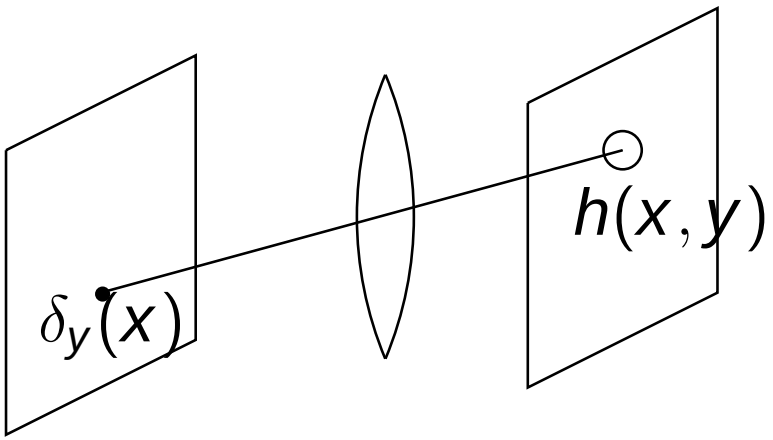
i.e. a **convolution**.

The proof is left as an exercise.

Point-spread function

Example

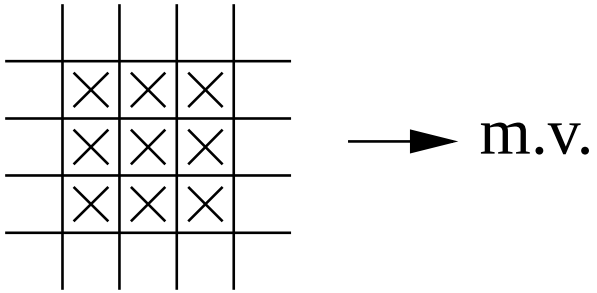
In an optical system $h(x, y) = S(\delta_y(x))$ *is called a* **point-spread function**.



The mean filter

Example

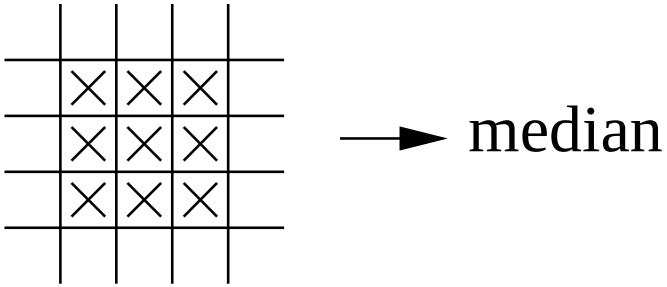
$g(x,y) =$ *mean of values* f *in a* 3×3*-neighborhood is called a* **mean filter**.



The median filter

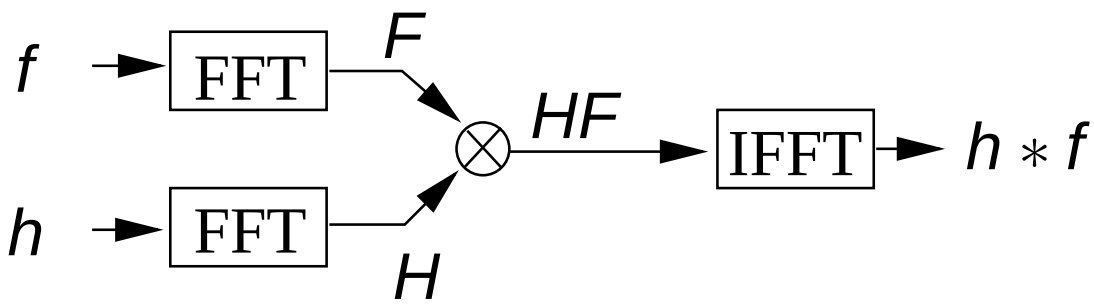
Example

$g(x,y)$ = median of f in a 3×3-neighborhood is called a **median filter** *(not linear filter).*



Computational aspects

1. $f \rightarrow FFT \rightarrow F$
2. $h \rightarrow FFT \rightarrow H$
3. $H, F \rightarrow \times \rightarrow H \cdot F$
4. $H \cdot F \rightarrow IFFT \rightarrow h * f$



Computational complexity

The computational complexity of using FFT for a convolution is:

$$2\frac{N \log N}{2} + N + \frac{N \log N}{2} \sim \frac{3}{2} N \log N$$

Calculation based on the definition gives complexity N^2

Convolution in two dimensions

f: $M \times N$-matrix (image)
Extend f to a double periodic function; "infinite matrix", f_P:

$$f(x+M, y+N) = f(x,y) \text{ for all } x, y \in \mathbb{N}$$

Definition

The convolution between two double periodic functions f and h, with the same period $M \times N$ is defined by

$$(h * f)(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} h(x-m, y-n) f(m,n)$$

■

$h * f$ is also $M \times N$-periodic.

Computational complexity

The number of multiplications needed using the convolution theorem is

$$\frac{3}{2} MN \log(MN)$$

The number of multiplications in the signal space is

$$MNmn \quad ,$$

where the kernel has size $m \times n$.
Conclusion: If $mn < \frac{3}{2} \log(MN)$ then it is better to compute the convolution in the signal plane.

Example

Example

256×256-*image:*

$$\frac{3}{2}\log(MN) \approx 24$$

5×5-*kernel is the border case*
Smaller kernel: Use the signal plane.
Larger kernel: Use the frequency plane.

The frequency function of a filter

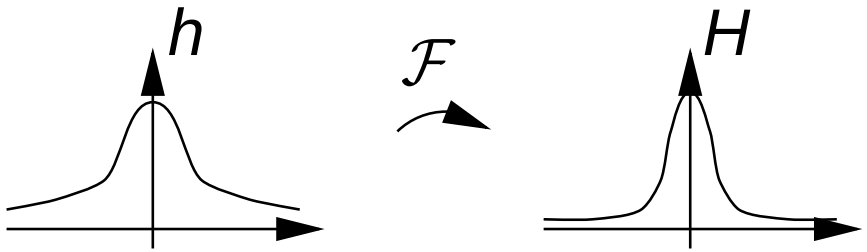
$$g(x) = h * f = \int h(x-y)f(y)dy$$

$$\mathcal{F}g = G, \mathcal{F}h = H, \mathcal{F}f = F$$

$$G(u,v) = H(u,v)F(u,v) \quad .$$

Definition

$H = \mathcal{F}(h)$ is called the **frequency function** of h. ■



Different filters

Filter for image enhancement

signal plane	frequency plane
smoothing	low pass
sharpening	high pass

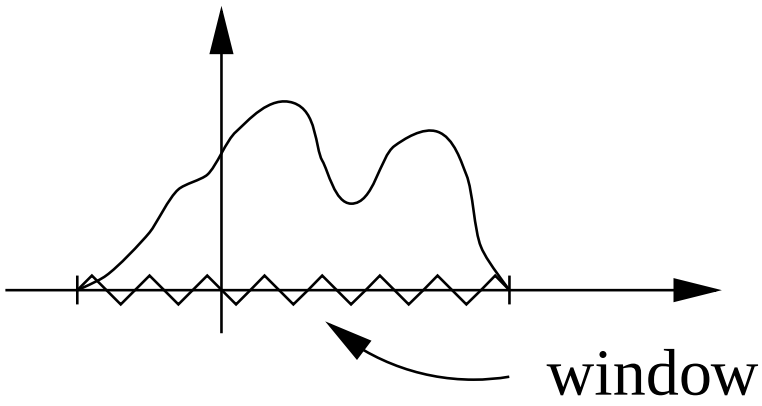
For discrete functions: $DFT\,(h * f)(u,v) = H(u,v)F(u,v)$.

Window operator

Let the output, g be give by the convolution

$$g(x) = S(f)(x) = \int h(x-y)f(y)dy \quad ,$$

where f represents the input and h the impulse response
If $g(x)$ only depends on f:s values in a surrounding (=a small window) of x then S is called a **window operator**.
The window is given by $\{ x \mid h(x) \neq 0 \}$.



Mean value operator

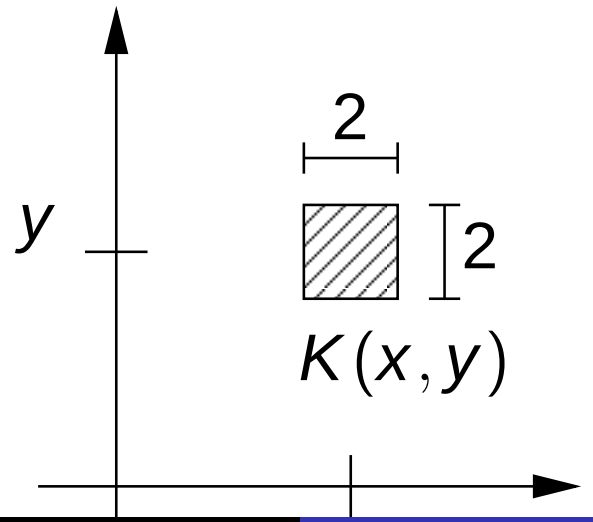
Assume that $f(x,y)$ represents a continuous image. Let

$$h(x,y) = \text{rect}(x)\,\text{rect}(y) \quad .$$

Then

$$S(f) = h * f = \int_{K(x,y)} f(s,t)dsdt \quad ,$$

where the region of integration $K(x,y)$ is a unit square with centre at (x,y).



Mean value filter

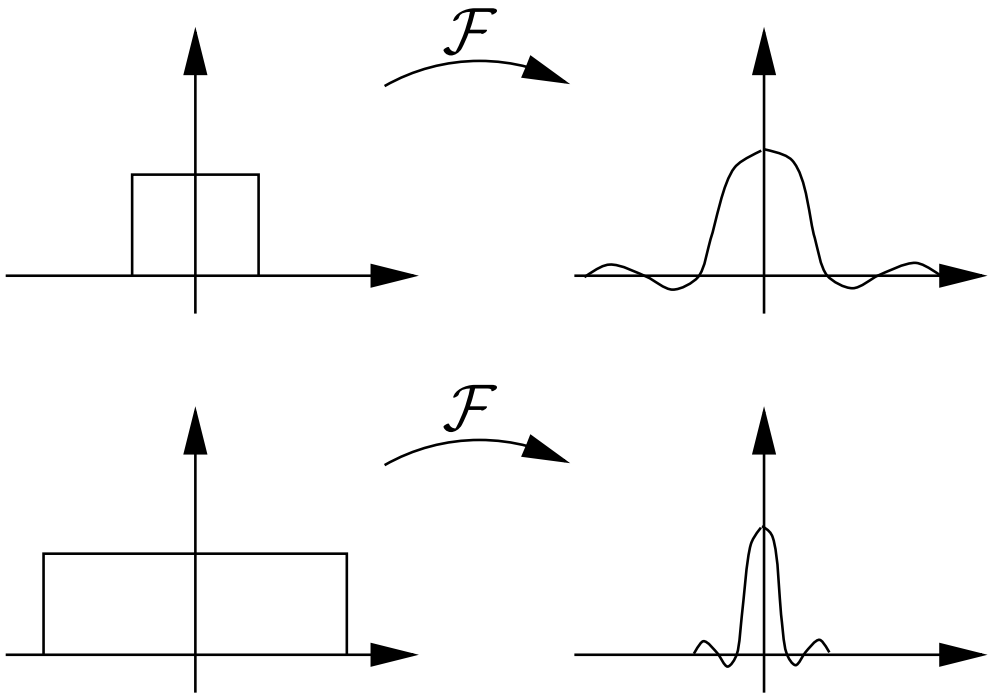
S is called a **mean value operator**.
The fourier transform gives

$$H(u) = 4\,\mathrm{sinc}(2\pi u)\,\mathrm{sinc}(2\pi v) \quad .$$

The scaling rule (page 148 in Forsythe-Ponce)

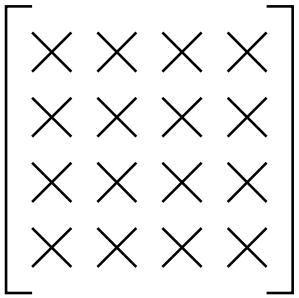
$$f(\lambda x) \rightarrow \frac{1}{\lambda} F(\frac{u}{\lambda}) \quad .$$

Illustration



Filters and discrete images

For discrete images, convolve with a **kernel**, represented by a matrix:

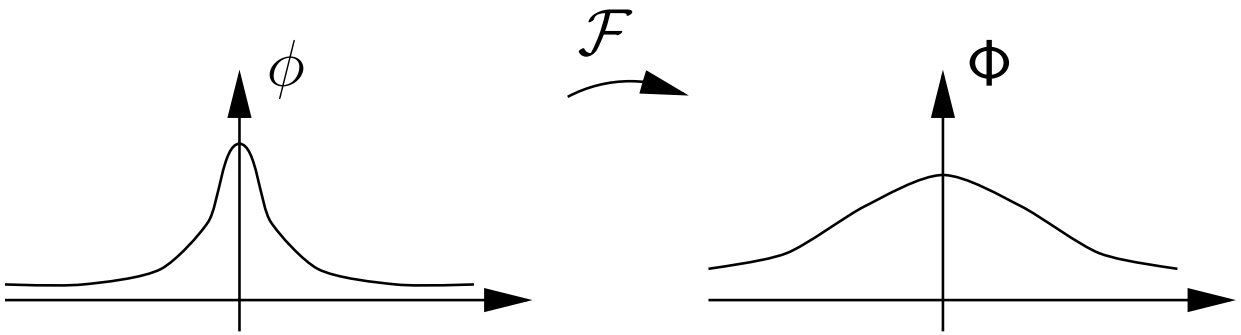


Example: Gaussian Filters

Example

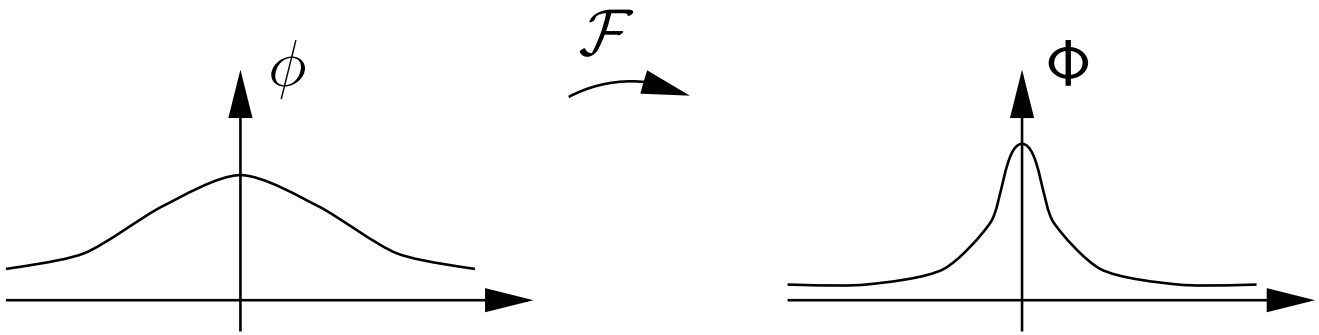
Notice that

$$\phi(x) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-x^2/(2\sigma^2)} \quad \rightarrow \quad \Phi(u) = e^{-2(\sigma\pi u)^2}.$$



Example: Effect of different σ

Larger σ gives



More smoothing $\Rightarrow$ more low-pass type.

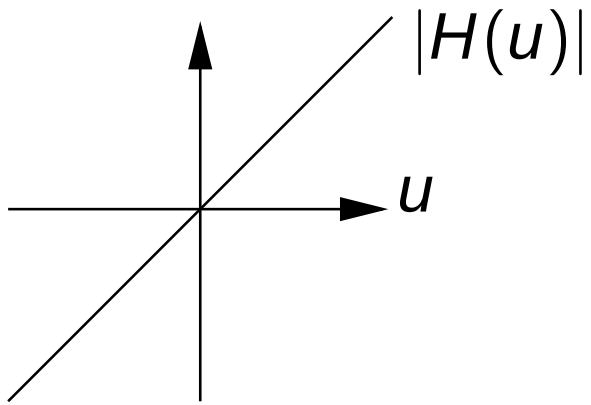
Differentiation filter

Example

Differentiation

$$\frac{\partial f}{\partial x} \quad \rightarrow \quad 2\pi i u F(u)$$

$$H(u) = 2\pi i u$$



High-pass properties of differentiation

High-pass filter

$$h = \frac{\partial \delta}{\partial x} \quad \text{since}$$

$$f = \delta * f \quad \Rightarrow \quad \frac{\partial f}{\partial x} = \frac{\partial \delta}{\partial x} * f$$

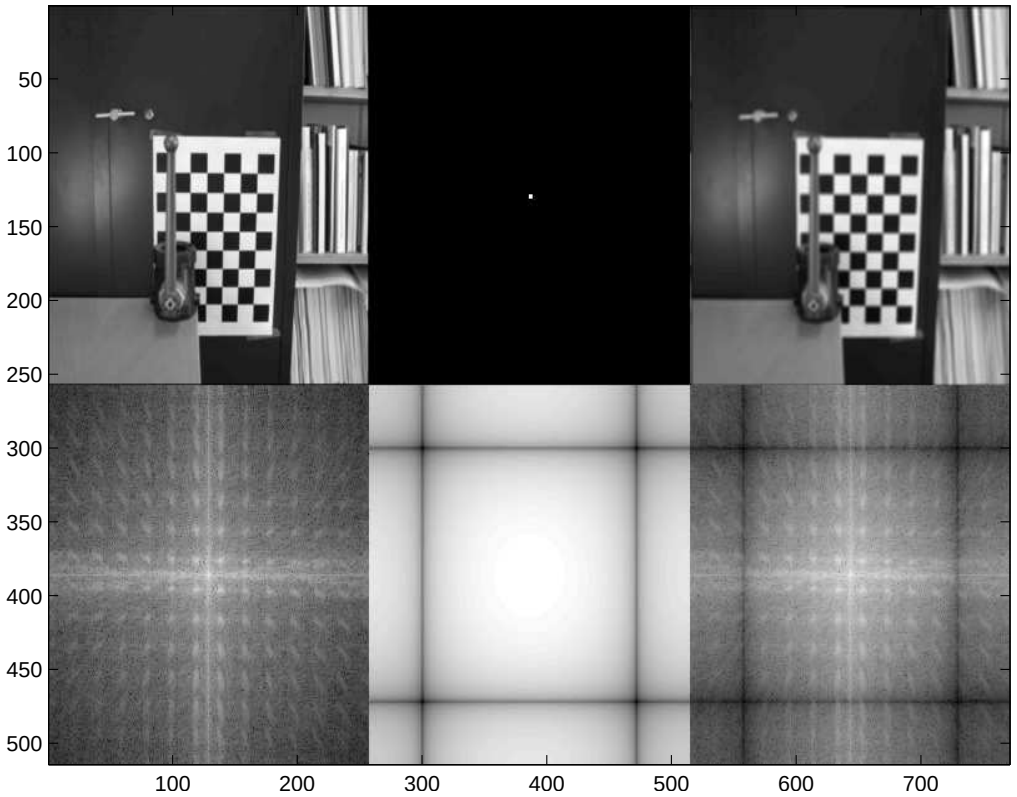
Sensitive to noise.
Combine with smoothing:

$$f \rightarrow \phi * f \rightarrow \frac{\partial}{\partial x} \phi * f$$

$$\frac{\partial}{\partial x} \phi = -\frac{x}{\sqrt{2\sigma^6 \pi}} e^{-x^2/(2\sigma^2)}$$

Filter example: small mean value filter

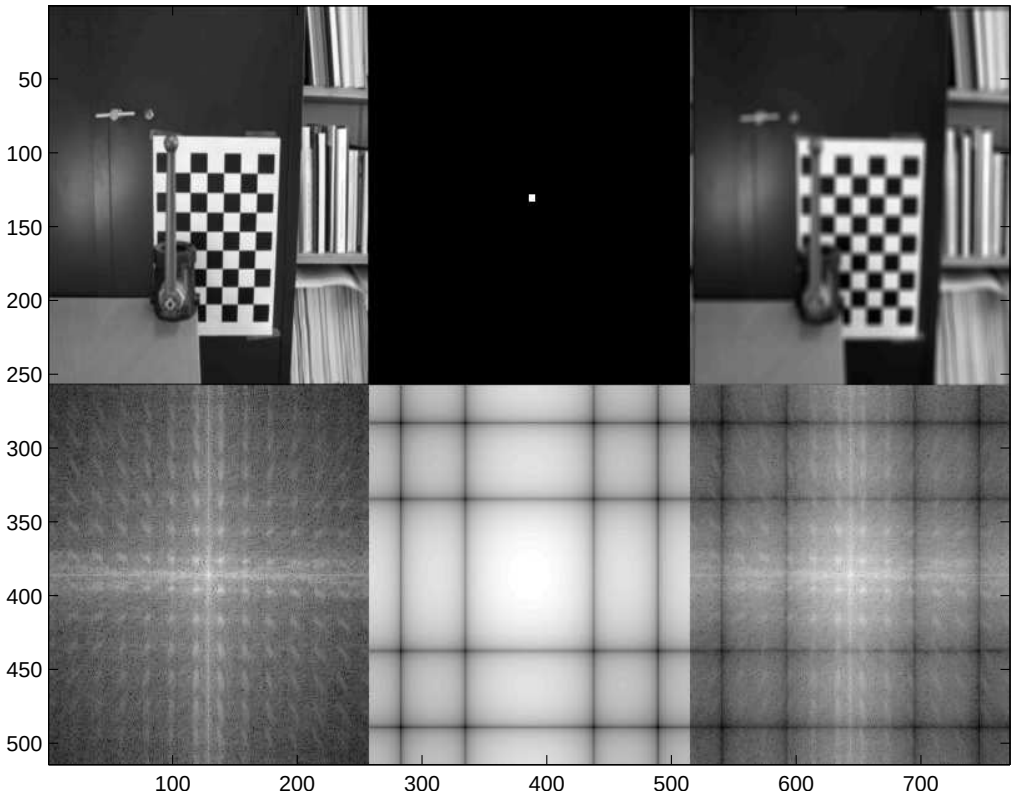
signal space: image, filter, result



frequency space: image, filter result

Filter example: medium mean value filter

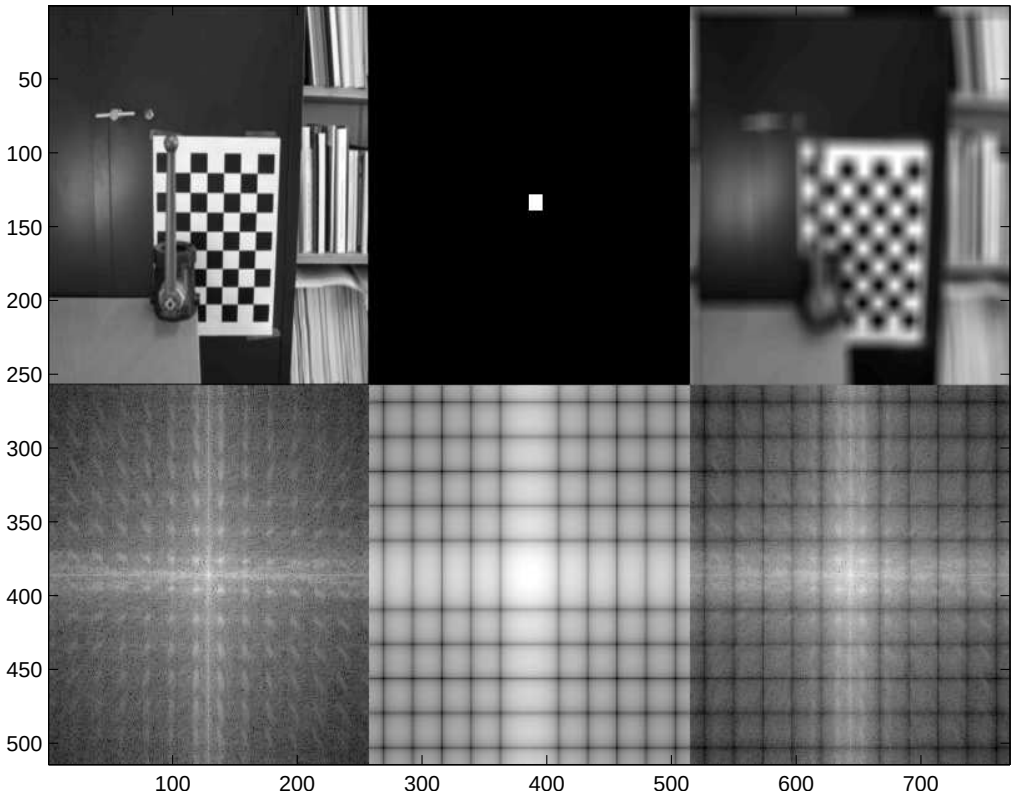
signal space: image, filter, result



frequency space: image, filter result

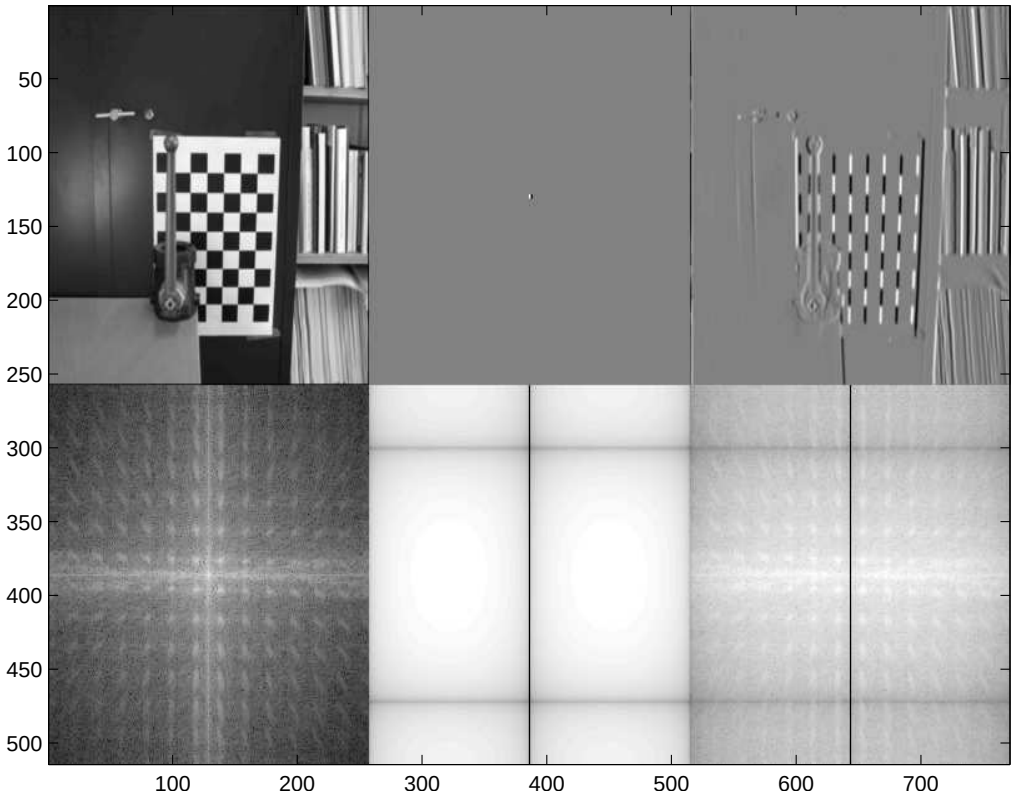
Filter example: larger mean value filter

signal space: image, filter, result

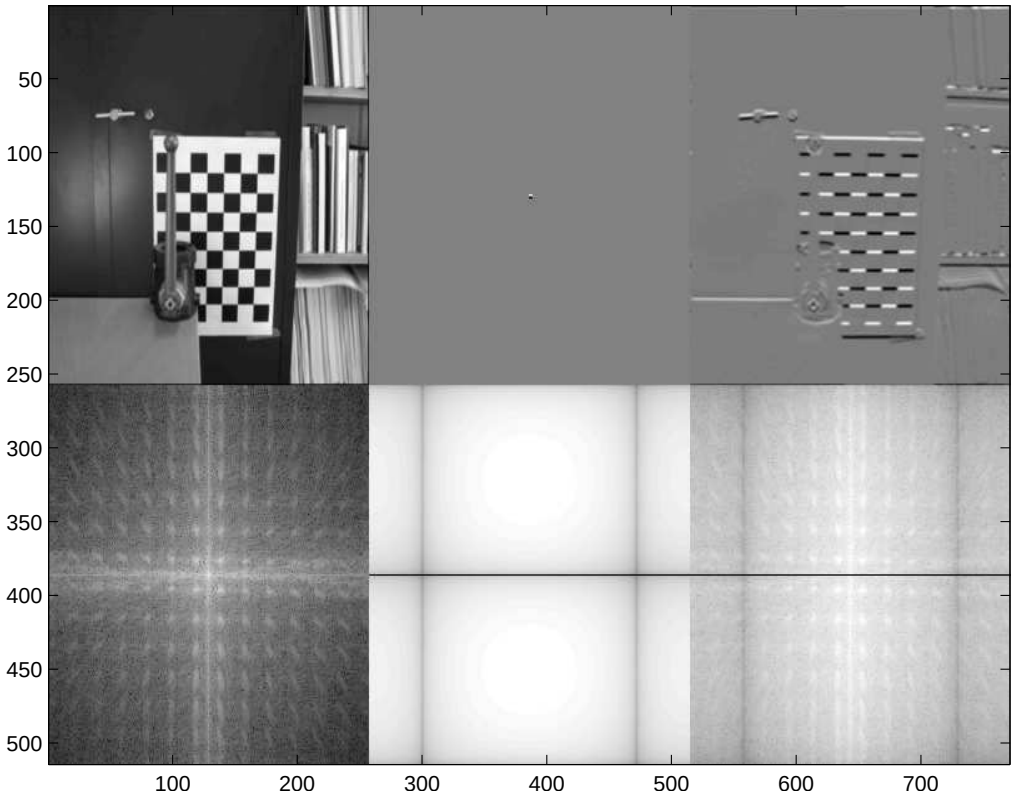


frequency space: image, filter result

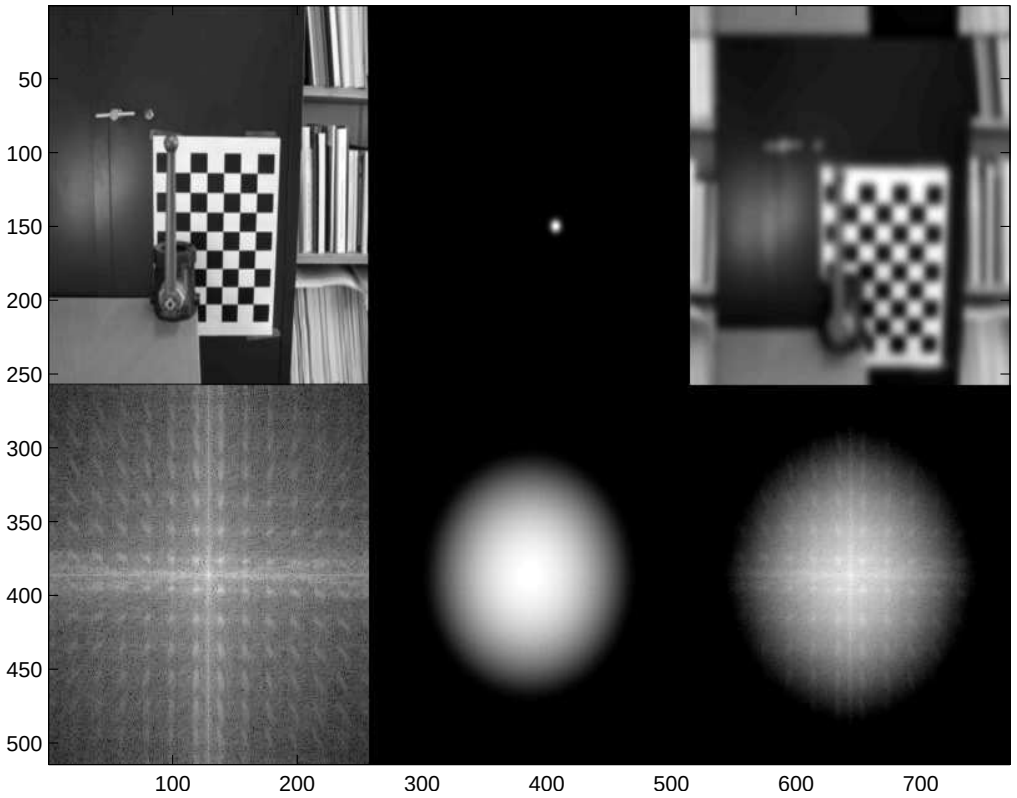
Filter example: differentiation in the y-direction



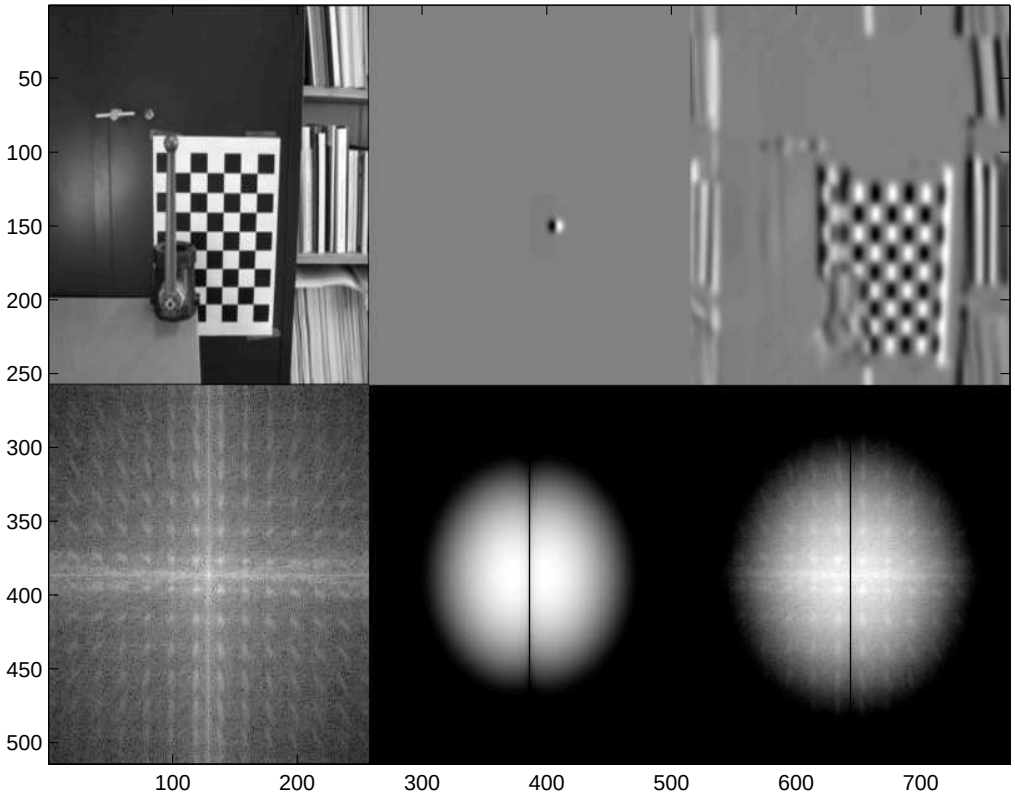
Filter example: differentiation in the x-direction



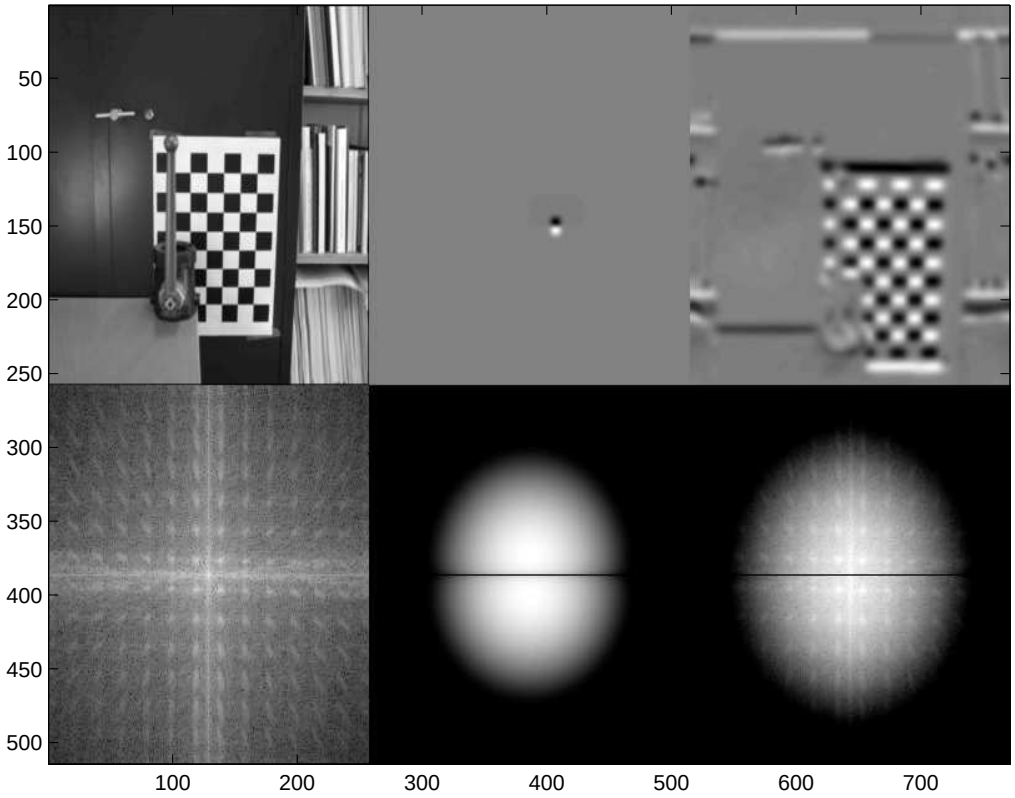
Filter example: Gaussian filter



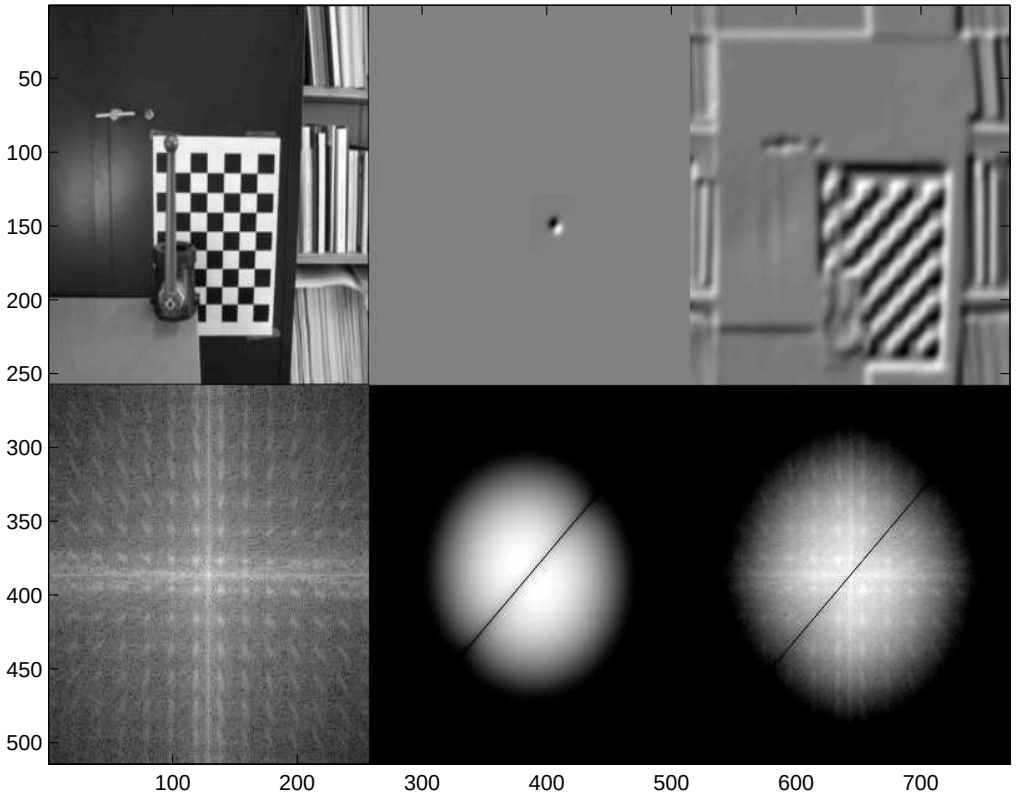
Filter example: Differentiation and Gaussian in x-direction



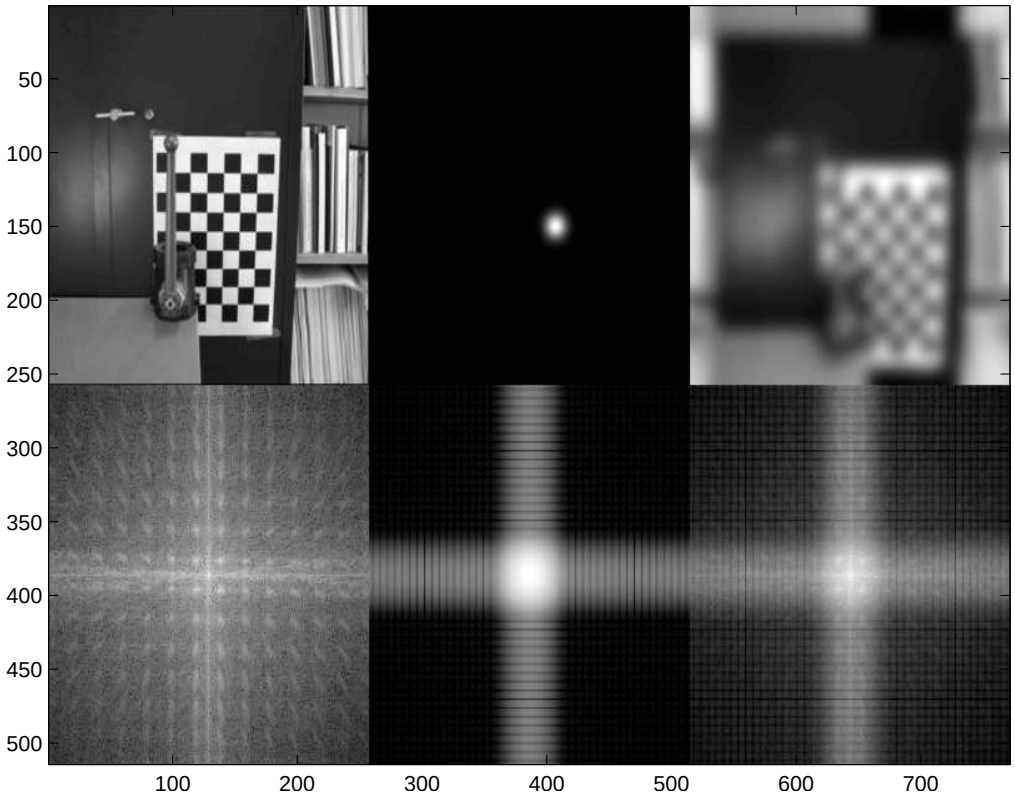
Filter example: Differentiation and Gaussian in y-direction



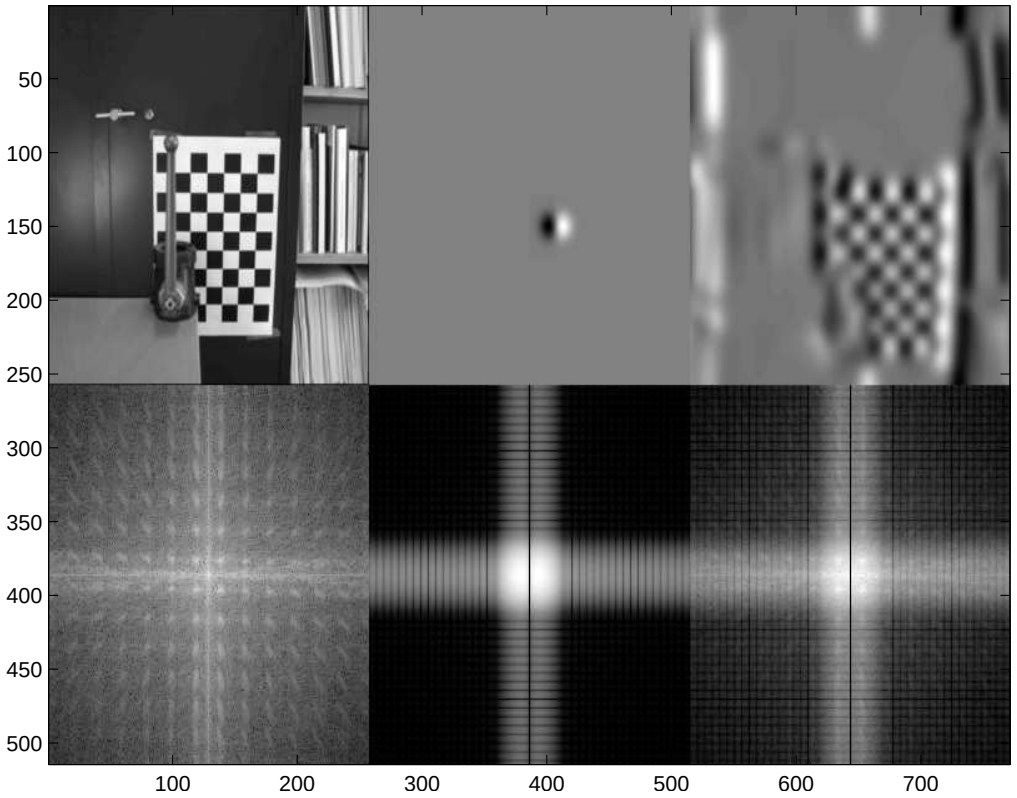
Filter example: Differentiation and Gaussian in a general direction



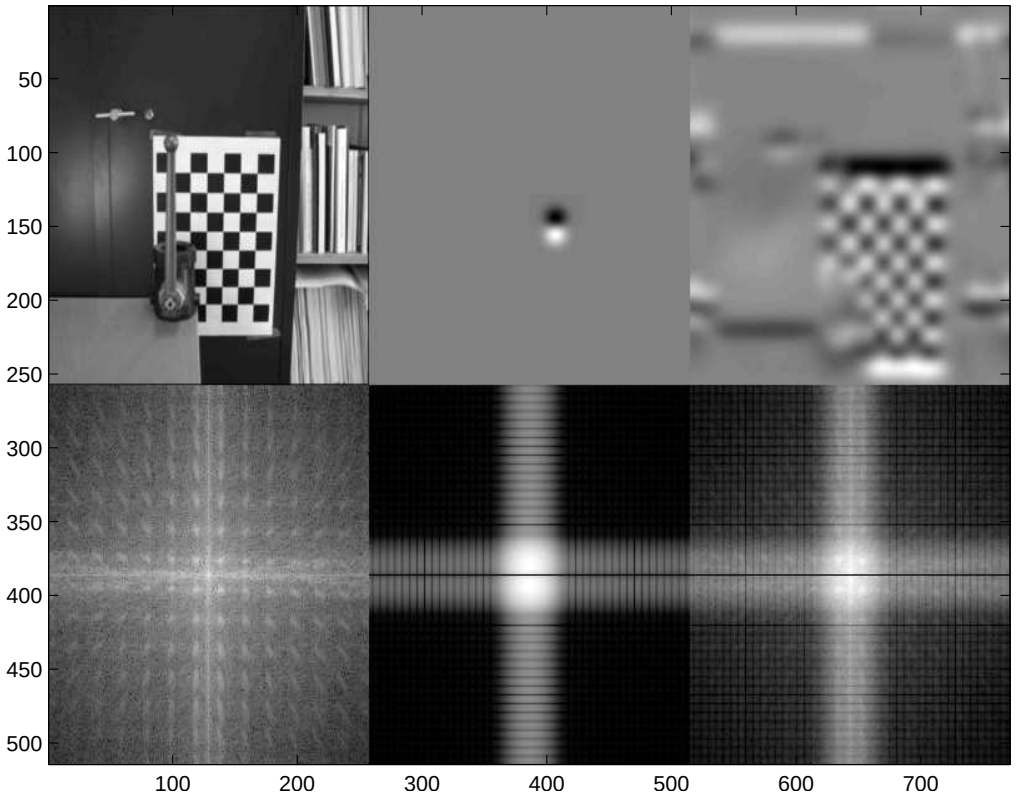
Filter example: Increasing σ: Filter?



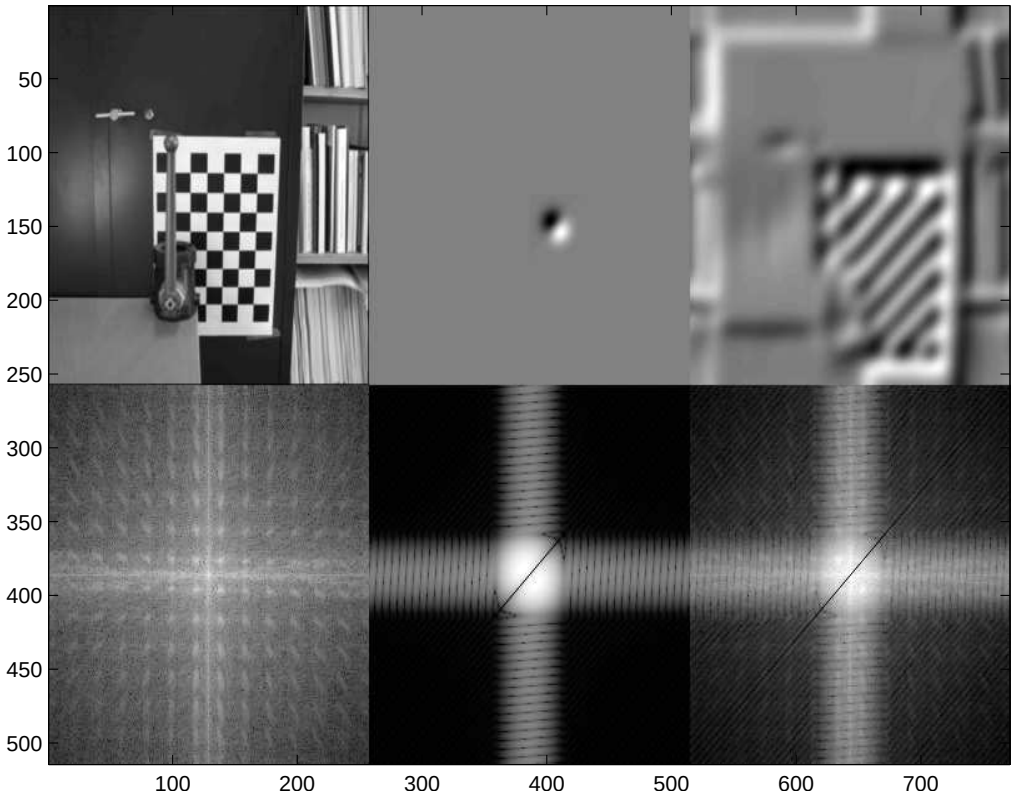
Filter example: Increasing σ: Der in y-dir.



Filter example: Increasing σ: Der in y-dir.



Filter example: Increasing σ: Der in [11]-dir.



Finding Patterns

Observation:
Filter kernels look like the effects they are intended to detect.

Read chapter 7.6: about detecting and tracking hands using convolution.

Finding point correspondences

Correlation can be used to find corresponding points in two similar images.

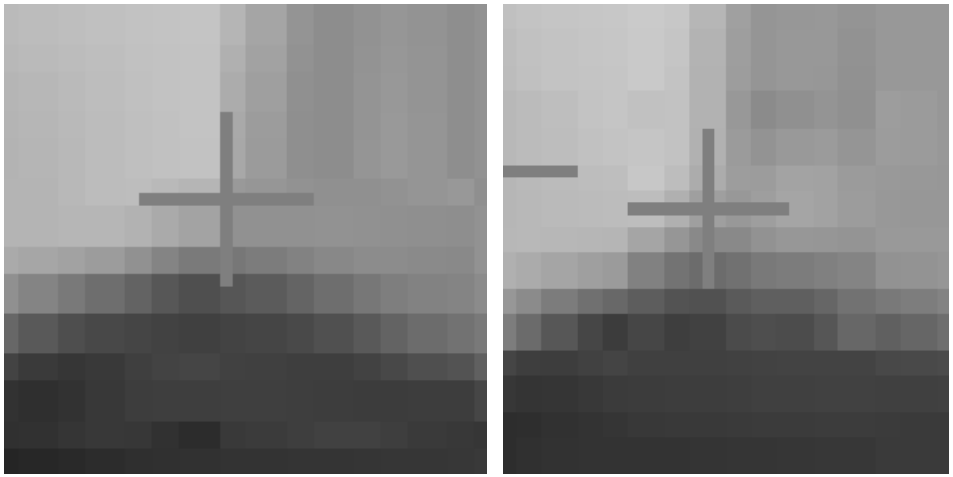
Correlation
Let $\Omega_A \subset \mathbb{R}^2$ be a neighborhood of a feature point in an image with intensities I_A. Similarly let Ω_B and I_B be the neighborhood and the intensities around a feature point in another image.

Point correspondences from correlation

The correlation between the neighborhoods Ω_A and Ω_B (assumed to be the same size) is defined as

$$C_{I_A,I_B} = \int_{\Omega_A} (I_A((x,y)) - I_B(T(x,y))^2 dxdy \quad ,$$

where $T(x,y)$ is a transformation that maps Ω_A onto Ω_B.



A Note

$$\int (I_A - I_B)^2 = \int I_A^2 - 2\int I_A I_B + \int I_B^2 \approx \text{const.} - 2\int I_A I_B \ ,$$

i.e. the correlation can be approximated with a convolution, if you only use integer translations. This is because $\int I_A^2$ is constant when A is fixed and $\int I_B^2$ is approximately constant when B varies.

Conclusions: Convolution (using FFT) can be used to find the correlation at every position at once.

Again: Kernels look like the effect they are intended to detect.

In Matlab

```
patch = image1(rpos1(2)-s:rpos1(2)+s,rpos1(1)-s:rpos1(1)+s);
patch = flipud(fliplr(patch));
AA = sum(sum(patch.*patch));
BBt = image2.*image2;
BB = conv2(BBt,ones(size(patch)));
disp(['Convolving image']);
AB = conv2(image2,patch);
disp(['Calculating correlation']);
res = AA - 2*AB + BB;
disp(['Finding minimum']);
f = min(min(res));
```

Recommended reading

- Forsyth & Ponce: **7. Linear filters**. Lectures 2-3.
- Szeliski: **3. Image processing**, sections 3.2-3.4. Lectures 2-3.

Review - Lecture 3

- Examples of fourier transforms
- Convolution
- Edge effects
- Understanding convolutions
- Convolutions to find patterns