

Image Analysis - Lecture 3

Kalle Åström

6 September 2016

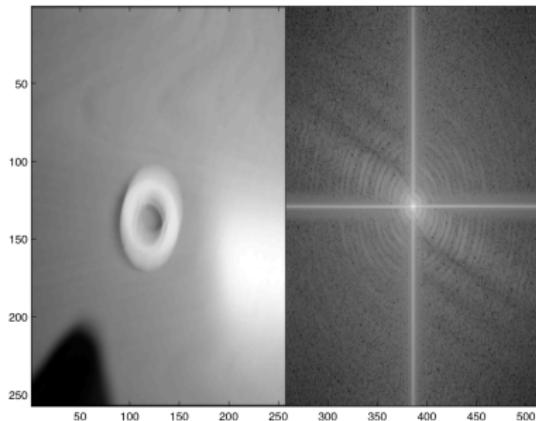
Contents

- ▶ Understanding the Fourier transform
 - ▶ Effects due to periodic continuation
 - ▶ Effects due to edges/structure in the images
- ▶ Filters and convolution
 - ▶ The convolution theorem, circulant matrices
 - ▶ Low pass-, high pass-, median-, mean value-, differentiation-filters
 - ▶ Scale-space
 - ▶ Finding patterns
- ▶ Master thesis proposal of the day

Understanding the Fourier transform

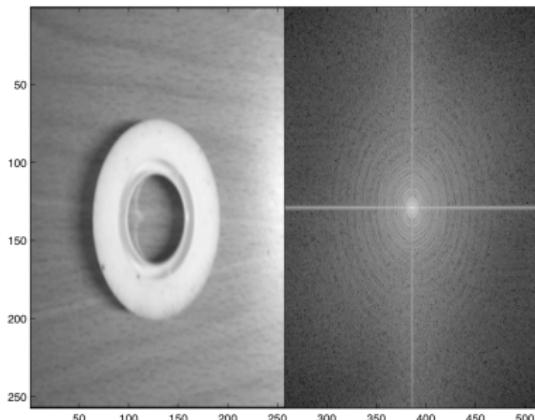
- ▶ After fftshift **low frequencies** are located in the centre of the FFT image
- ▶ **High frequencies** are located at the borders of the FFT image
- ▶ The fact that images are considered as periodic functions, the borders of the images can influence the FFT
- ▶ High frequencies in one direction (e.g.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

Image b and $\log(\text{abs}(\text{fftshift}(\text{fft2}(b))))$



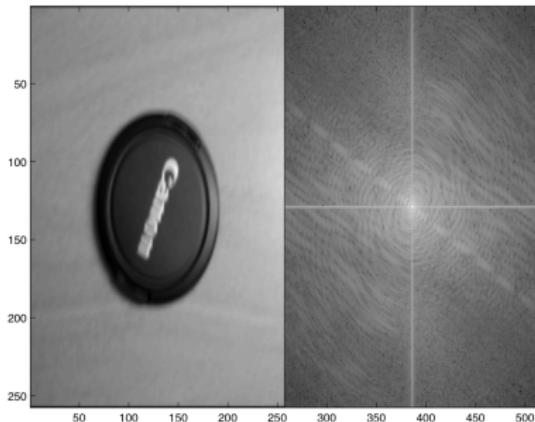
Observe the boundary effects!

Image b and $\log(\text{abs}(\text{fftshift}(\text{fft2}(b))))$



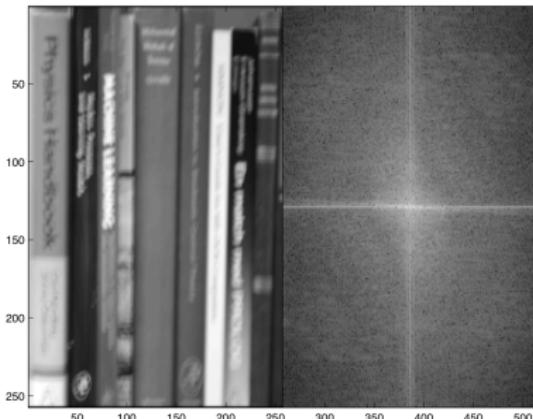
Smaller boundary effects due to more uniform background.

Image b and $\log(\text{abs}(\text{fftshift}(\text{fft2}(b)))$



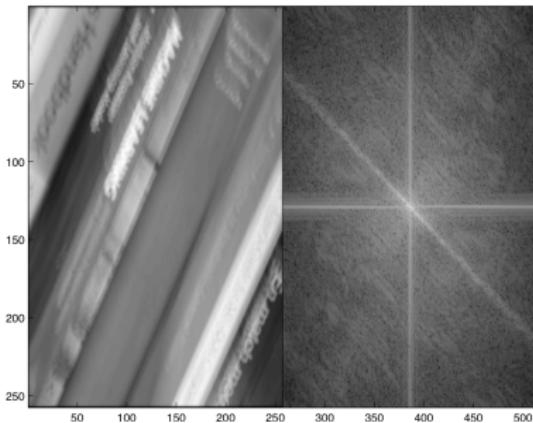
Sharp edges in the image gives high frequencies in the FFT image.

Image b and $\log(\text{abs}(\text{fftshift}(\text{fft2}(b)))$



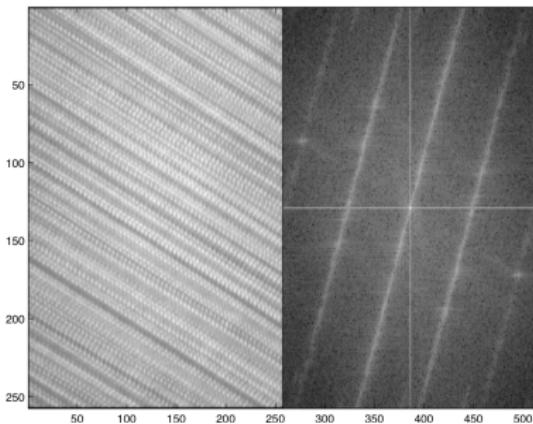
Sharp edges in the image gives high frequencies in the FFT image.

Image b and $\log(\text{abs}(\text{fftshift}(\text{fft2}(b)))$



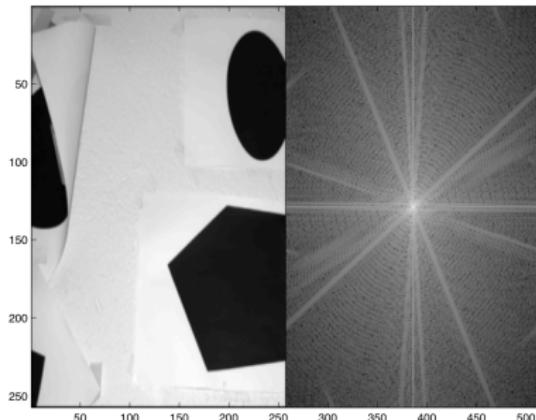
Observe the directions of the edges and the high frequency components.

Image b and $\log(\text{abs}(\text{fftshift}(\text{fft2}(b)))$



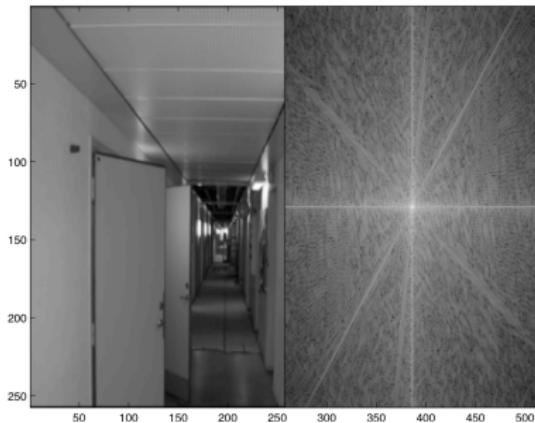
Several edges can give rise to several high frequency components.

Image b and $\log(\text{abs}(\text{fftshift}(\text{fft2}(b)))$



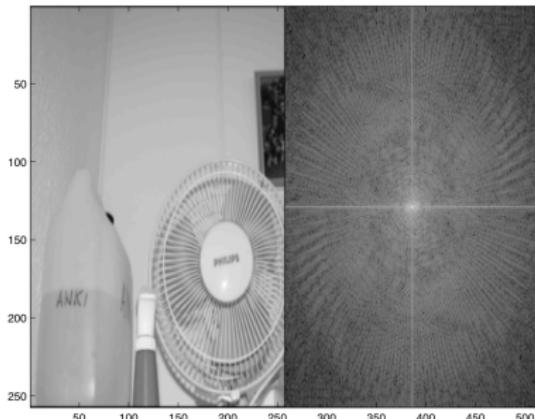
Identify the different edge directions and the corresponding high frequency components.

Image b and $\log(\text{abs}(\text{fftshift}(\text{fft2}(b))))$



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

Image b and $\log(\text{abs}(\text{fftshift}(\text{fft2}(b)))$



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} .

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

Observe that

$$f * g = g * f .$$



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} .

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



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 convolute the image

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

with the **smoothing filter**

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

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

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

(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

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

Here we have also made a periodic function of h :

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

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.

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

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

$$\begin{aligned} g(i, j) &= \sum_{u,v} f(u, v) h(i - u, j - v) = \\ &= \sum_{u,v} f(u, v) \check{h}(i + u, i + v) = \sum_{u,v} f(i + u, j + v) \check{h}(u, v) . \quad (1) \end{aligned}$$

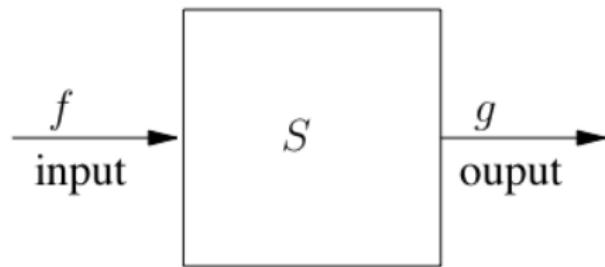
Two-dimensional periodic convolution

Periodic continuation gives:

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

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

Convolution as image transform



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

Linear translational invariant systems

Any linear and translation invariant system can be represented by

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

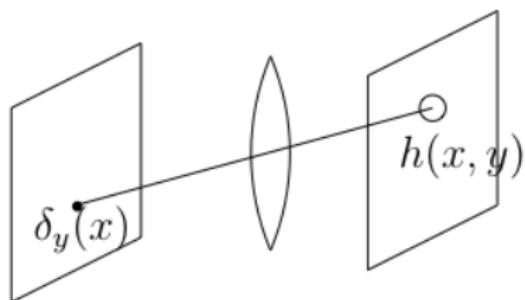
i.e. a **convolution**.

The proof is left as an exercise.

Point-spread function

Example

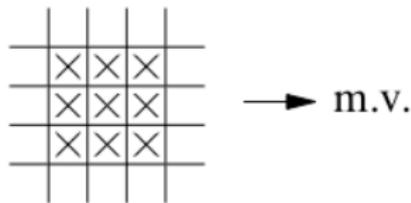
In an optical system $h(x, y) = S(\delta_y(x))$ is called a point-spread function.



The mean filter

Example

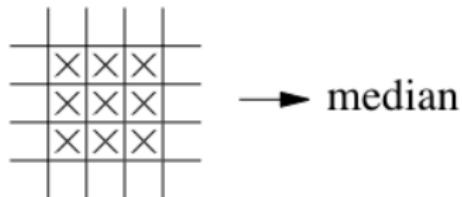
$g(x, y) = \text{mean of values } f \text{ in a } 3 \times 3\text{-neighborhood}$ is called a **mean filter**.



The median filter

Example

$g(x, y) = \text{median of } f \text{ in a } 3 \times 3\text{-neighborhood is called a}$
median filter (*not linear filter*).



The convolution theorem

First in one dimension

$$f = \begin{bmatrix} f(0) \\ \vdots \\ f(N-1) \end{bmatrix} \quad f_p = \begin{bmatrix} \vdots \\ f(N-1) \\ f(0) \\ \vdots \\ f(N-1) \\ f(0) \\ \vdots \end{bmatrix}$$

The period = $N \Rightarrow f(x + N) = f(x)$ for every x

$$g(x) = h * f(x) = \sum_{m=0}^{N-1} h(x-m)f(m), \quad x \in \mathbb{N}$$

Convolution as a linear mapping

The convolution of h and f can be written

$$g(x) = h * f(x) = \sum_{m=0}^{N-1} h(x-m)f(m), \quad x \in \mathbb{N}$$

$f \mapsto g$ is a linear mapping $\Rightarrow g = C_h f$.
What does C_h look like?

Calculating C_h

$$g(x) = h * f(x) = \sum_{m=0}^{N-1} h(x-m)f(m), \quad x \in \mathbb{N}$$

gives

$$g(N) = h(N)f(0) + h(N-1)f(1) + \dots + h(1)f(N-1)$$

$$g(N+1) = h(N+1)f(0) + h(N)f(1) + \dots + h(2)f(N-1)$$

...

$$\begin{aligned} g(2N-1) &= h(2N-1)f(0) + h(2N-2)f(1) + \dots + \\ &\quad + h(N)f(N-1) \end{aligned}$$

Using the periodicity

The periodicity gives

$$g(0) = h(0)f(0) + h(N-1)f(1) + \dots + h(1)f(N-1)$$

$$g(1) = h(1)f(0) + h(0)f(1) + \dots + h(2)f(N-1)$$

...

$$\begin{aligned} g(N-1) &= h(N-1)f(0) + h(N-2)f(1) + \dots + \\ &\quad + h(0)f(N-1) \end{aligned}$$

The Circulant matrix

In matrix form:

$$\begin{bmatrix} g(0) \\ g(1) \\ \vdots \\ g(N-1) \end{bmatrix} = C_h \begin{bmatrix} f(0) \\ f(1) \\ \vdots \\ f(N-1) \end{bmatrix}$$

with

$$C_h = \begin{bmatrix} h(0) & h(N-1) & \dots & h(2) & h(1) \\ h(1) & h(0) & \dots & h(3) & h(2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h(N-1) & h(N-2) & \dots & h(1) & h(0) \end{bmatrix}$$

Definition

The matrix C_h is called the **circulant matrix** for h .



A Lemma

We have $g = h * f \Rightarrow g = C_h f$. Introduce

$$\overline{\mathcal{F}_N} = [\phi_0 \quad \phi_1 \quad \dots \quad \phi_{N-1}] \quad \mathcal{F}_N = \text{the Fourier matrix}$$

Lemma

$$C_h \phi_i = H(i) \phi_i ,$$

where $H = \mathcal{F}_N h$.

In other words: All circulant matrices have the same eigenvectors, i.e. the columns ϕ_i of the conjugated Fourier Matrix. The eigenvalue corresponding to the eigenvector ϕ_i is the i :th Fourier coefficient of h .

Proof of Lemma

Proof: Start with $C_h\phi_i = h * \phi_i = \phi_i * h$. This implies that

$$\begin{aligned}(C_h\phi_i)(x) &= \sum_k \omega^{-(x-k)i} h(k) = \\ &= \omega^{-xi} \sum_k \omega^{ki} h(k) = H(i)\omega^{-xi},\end{aligned}$$

where H_i denotes the Fourier transform of h at frequency i . ■

A Consequence

The lemma gives

$$C_h \overline{\mathcal{F}_N} = \overline{\mathcal{F}_N} \begin{bmatrix} H(0) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & H(N-1) \end{bmatrix} = \overline{\mathcal{F}_N} \Lambda_H$$

Multiply with \mathcal{F}_N from the left:

$$\mathcal{F}_N C_h \overline{\mathcal{F}_N} = \mathcal{F}_N \overline{\mathcal{F}_N} \Lambda_H = N I \Lambda_H = N \Lambda_H$$

Multiply then with \mathcal{F}_N from the right:

$$\mathcal{F}_N C_h \overline{\mathcal{F}_N} \mathcal{F}_N = N \Lambda_H \mathcal{F}_N$$

Another Lemma

Remember

$$\mathcal{F}_N C_h \overline{\mathcal{F}_N} \mathcal{F}_N = N \Lambda_H \mathcal{F}_N$$

and that $\overline{\mathcal{F}_N} \mathcal{F}_N = NI$, giving

$$\mathcal{F}_N C_h \overline{\mathcal{F}_N} \mathcal{F}_N = \mathcal{F}_N C_h NI = N \mathcal{F}_N C_h$$

It follows that

Lemma

$$\mathcal{F}_N C_h = \Lambda_H \mathcal{F}_N$$

The convolution theorem

Theorem

$$\mathcal{F}(h * f) = \begin{bmatrix} H(0)F(0) \\ H(1)F(1) \\ \vdots \\ H(N-1)F(N-1) \end{bmatrix}$$

Proof of the convolution theorem

Proof:

$$\begin{aligned}\mathcal{F}(h * f) &= \mathcal{F}(C_h f) = \Lambda_H \mathcal{F}f = \\&= \begin{bmatrix} H(0) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & H(N-1) \end{bmatrix} \begin{bmatrix} F(0) \\ F(1) \\ \vdots \\ F(N-1) \end{bmatrix} = \\&= \begin{bmatrix} H(0)F(0) \\ H(1)F(1) \\ \vdots \\ H(N-1)F(N-1) \end{bmatrix}\end{aligned}$$

■

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.



Computation of 2D convolution

Row stack f and $h \Rightarrow \tilde{f}$ and \tilde{h} .

$g = h * f$ implies that \tilde{g} is a linear function of \tilde{f} , i.e.

$$\tilde{f} = C_{\tilde{h}} \tilde{g} .$$

The Block Circulant Matrix

Similar arguments as in the one dimensional case gives

$$C_{\tilde{h}} = \begin{bmatrix} C_{h(0,\cdot)} & C_{h(M-1,\cdot)} & \cdots & C_{h(1,\cdot)} \\ C_{h(1,\cdot)} & C_{h(0,\cdot)} & \cdots & C_{h(2,\cdot)} \\ \vdots & \vdots & \ddots & \vdots \\ C_{h(M-1,\cdot)} & C_{h(M-2,\cdot)} & \cdots & C_{h(0,\cdot)} \end{bmatrix},$$

where $C_{h(j,\cdot)}$ denotes the circulant matrix of the j :th row in h .

Definition

The matrix $C_{\tilde{h}}$ above is called the **block circulant matrix** of h .



The convolution theorem in 2D

Theorem

$$G(k, j) = H(k, j)F(k, j) ,$$

where G , H and F denote the Fourier Transform of $g = h * f$, h and f .

For details see notes.

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

$$MNmn ,$$

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

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

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

$$G(u, v) = H(u, v)F(u, v)$$

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

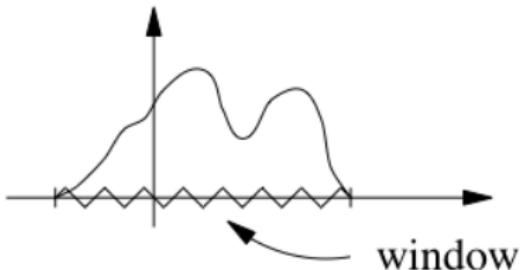
Let the output, g be given by the convolution

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

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

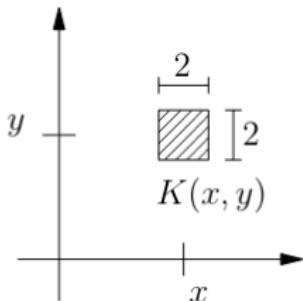
Assume that $f(x, y)$ represents a continuous image. Let

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

Then

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

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**.

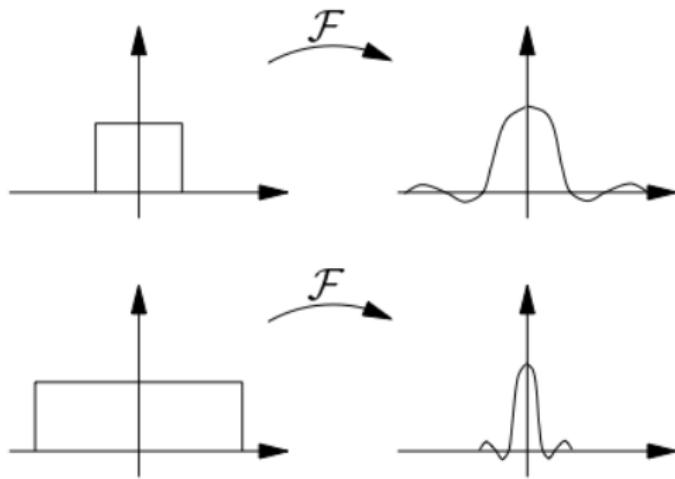
The fourier transform gives

$$H(u) = 4 \operatorname{sinc}(2\pi u) \operatorname{sinc}(2\pi v) .$$

The scaling rule (page 148 in Forsythe-Ponce)

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

Illustration



Filters and discrete images

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

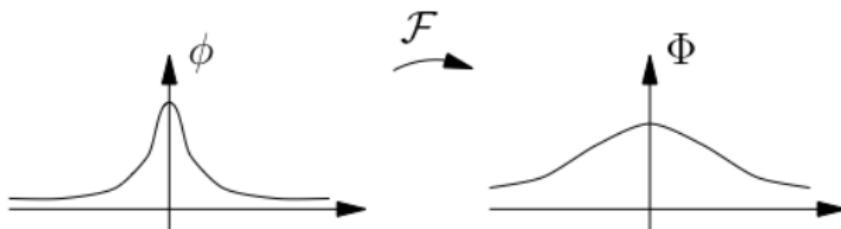
$$\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}$$

Example: Gaussian Filters

Example

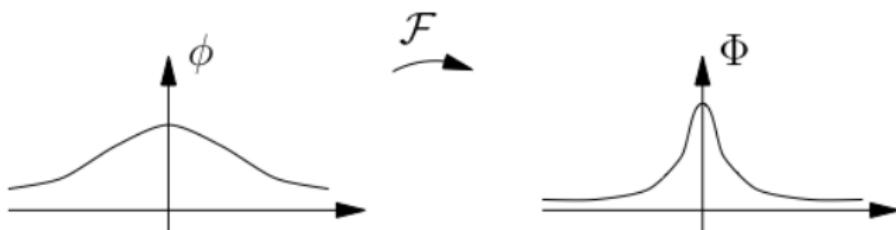
Notice that

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



Example: Effect of different σ

Larger σ gives



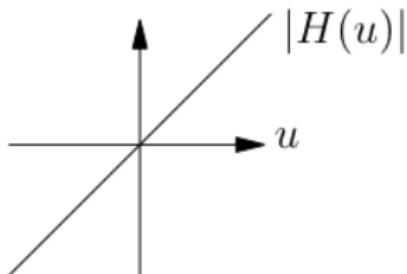
Differentiation filter

Example

Differentiation

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

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



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.

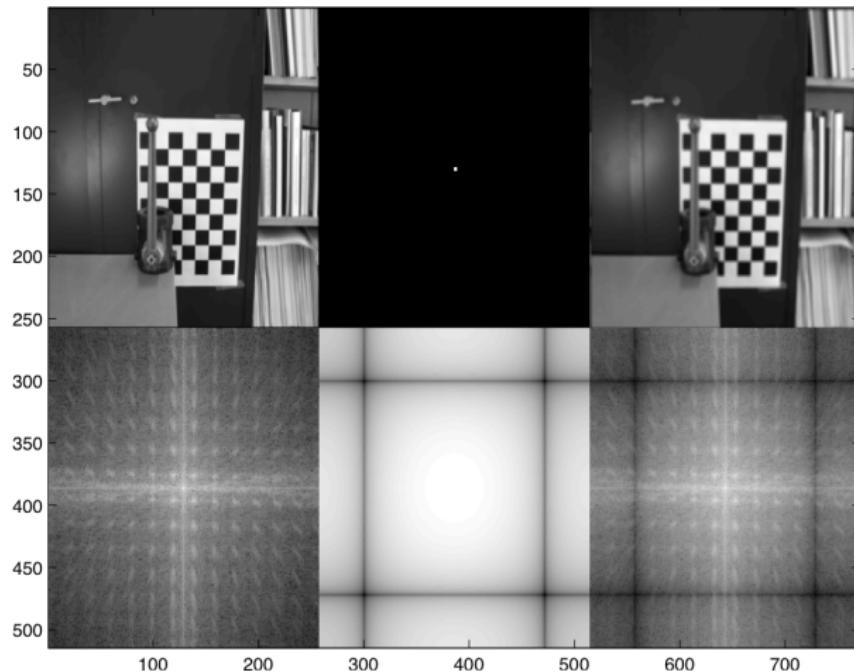
Combine with smoothing:

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

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

Filter example: small mean value filter

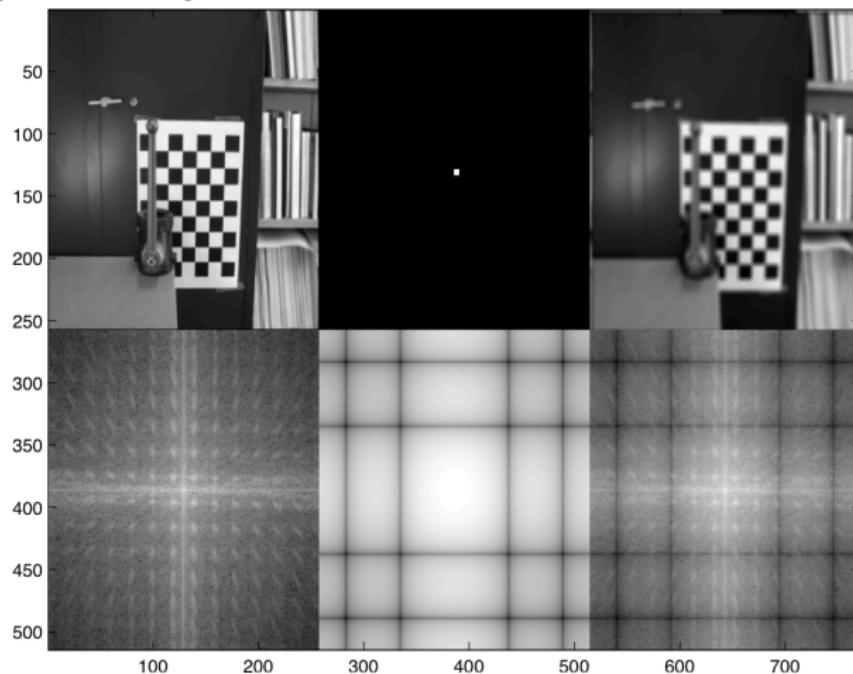
signal space: image, filter, result



frequency space: image, filter result

Filter example: medium mean value filter

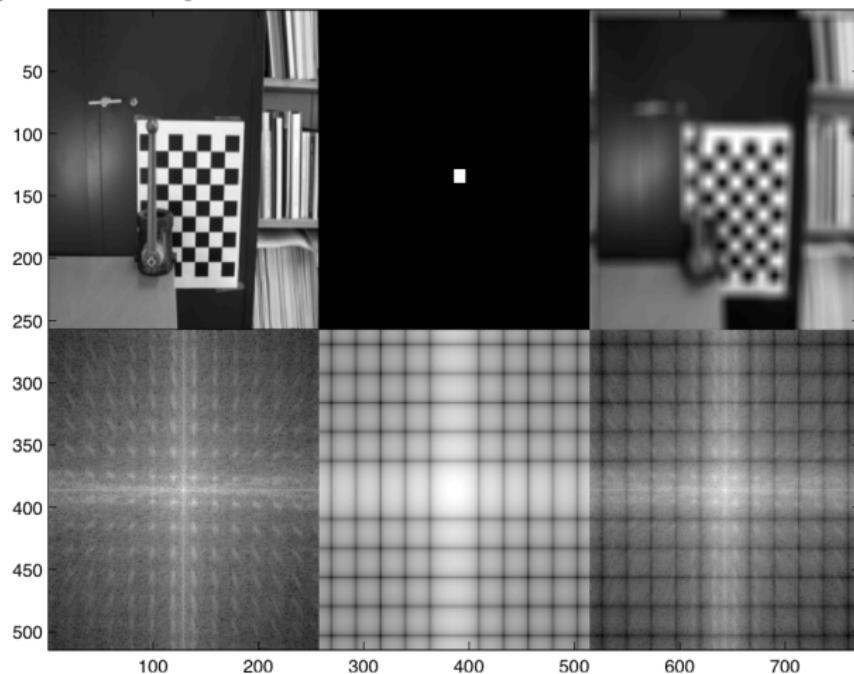
signal space: image, filter, result



frequency space: image, filter result

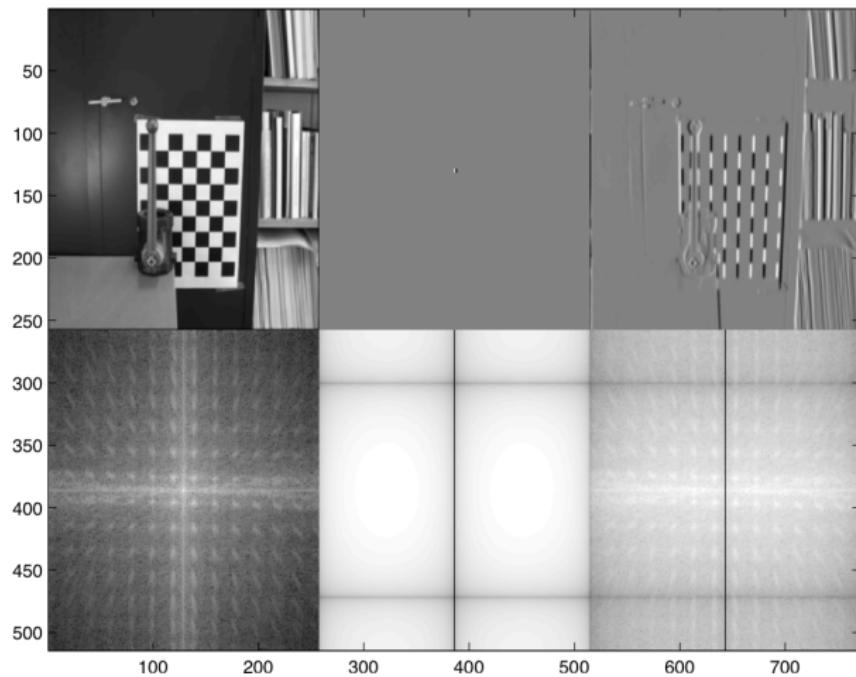
Filter example: larger mean value filter

signal space: image, filter, result

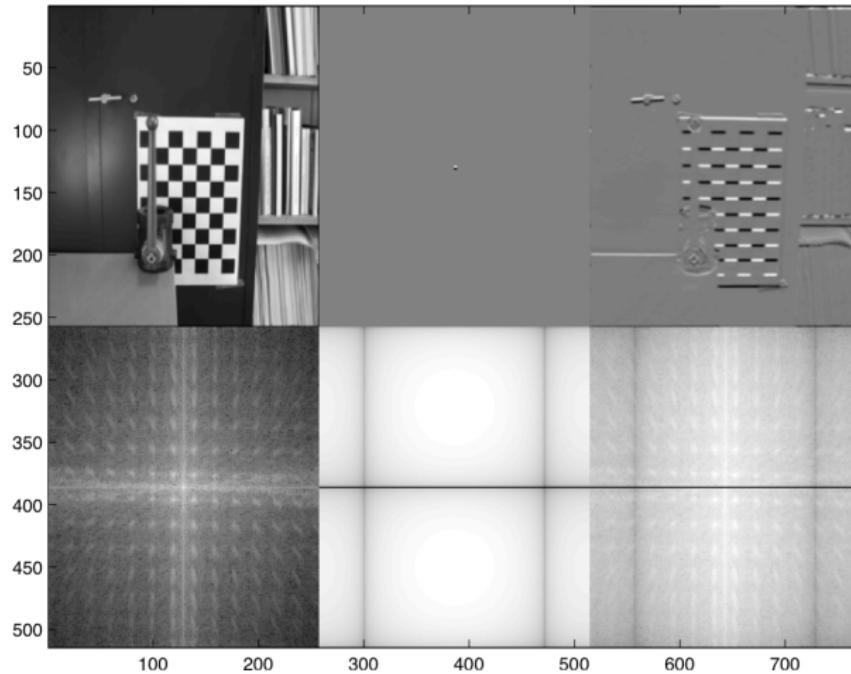


frequency space: image, filter result

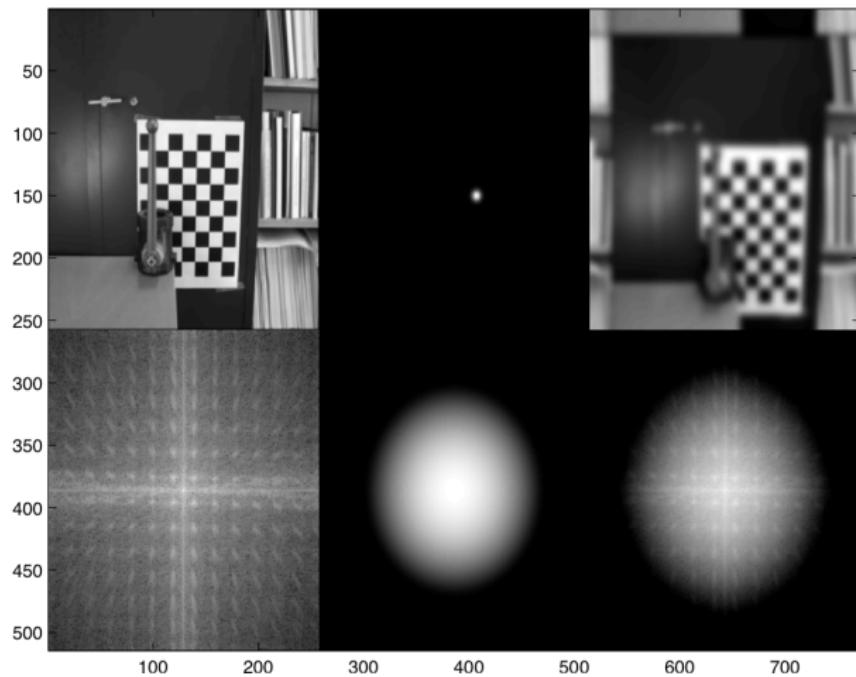
Filter example: differentiation in the y -direction



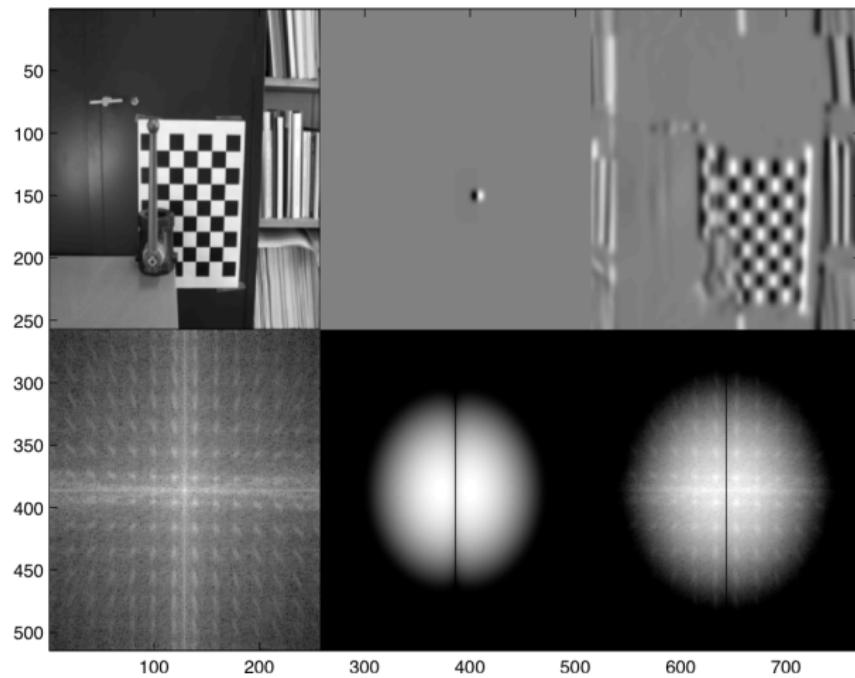
Filter example: differentiation in the x -direction



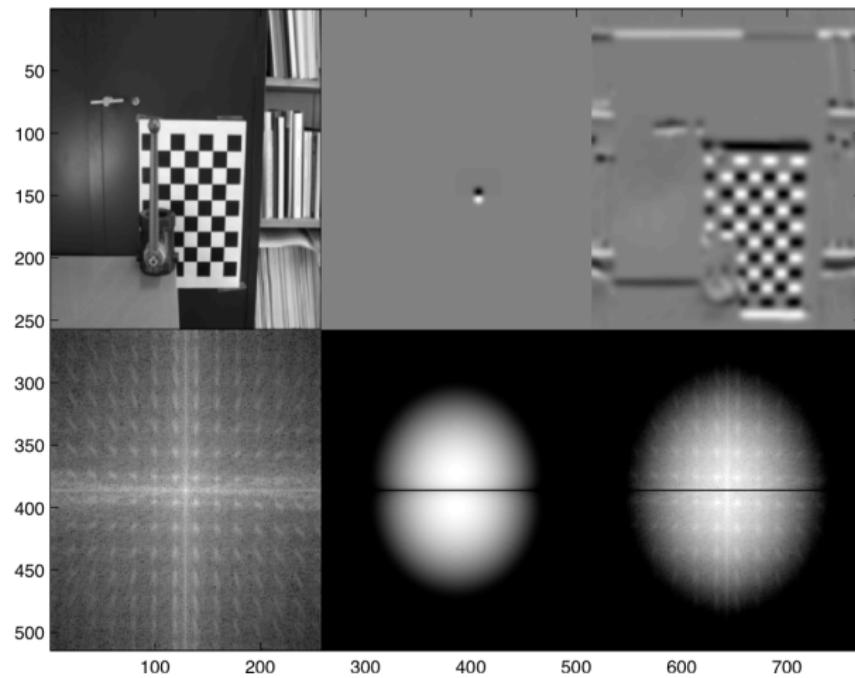
Filter example: Gaussian filter



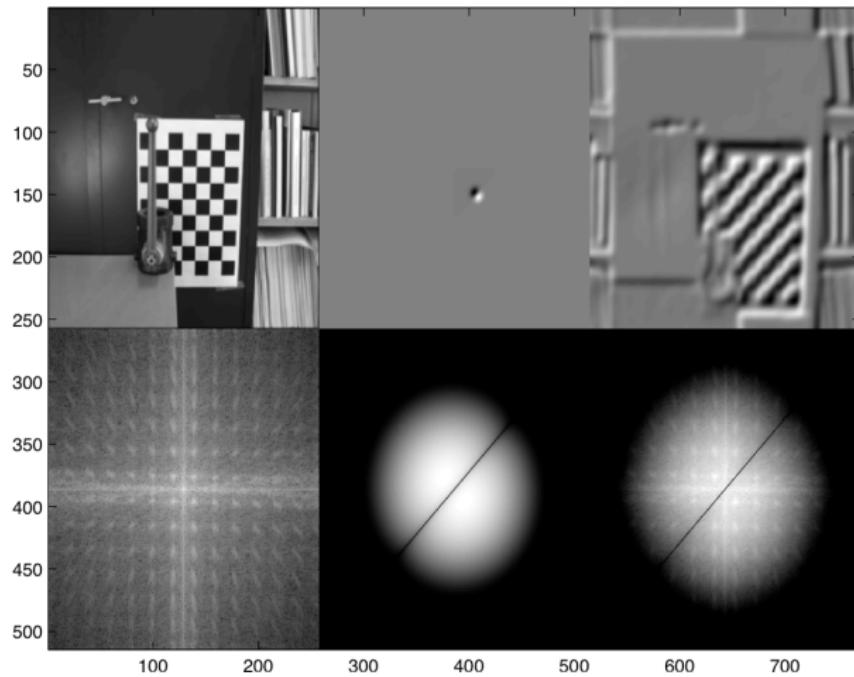
Filter example: Differentiation and Gaussian in x-direction



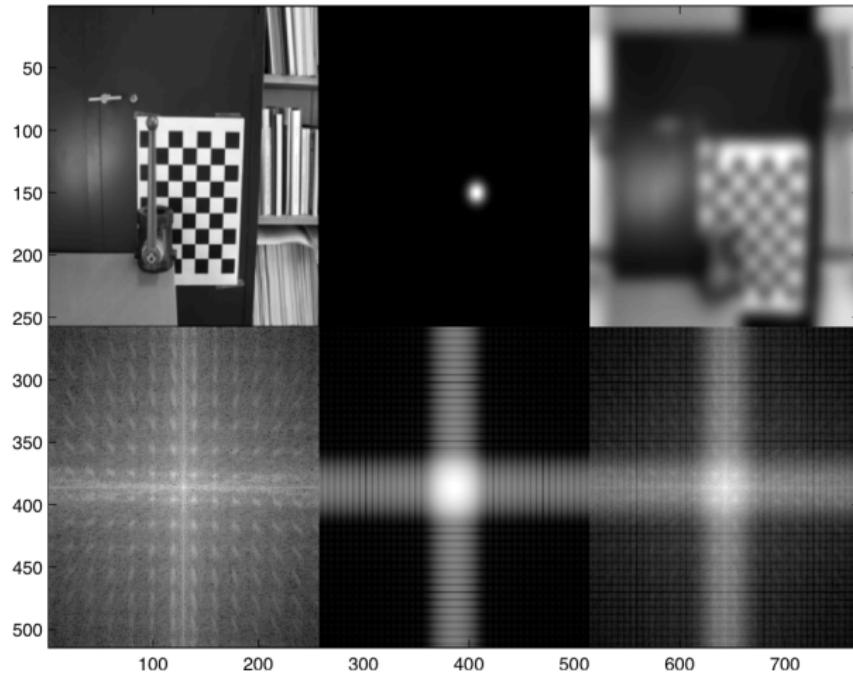
Filter example: Differentiation and Gaussian in y-direction



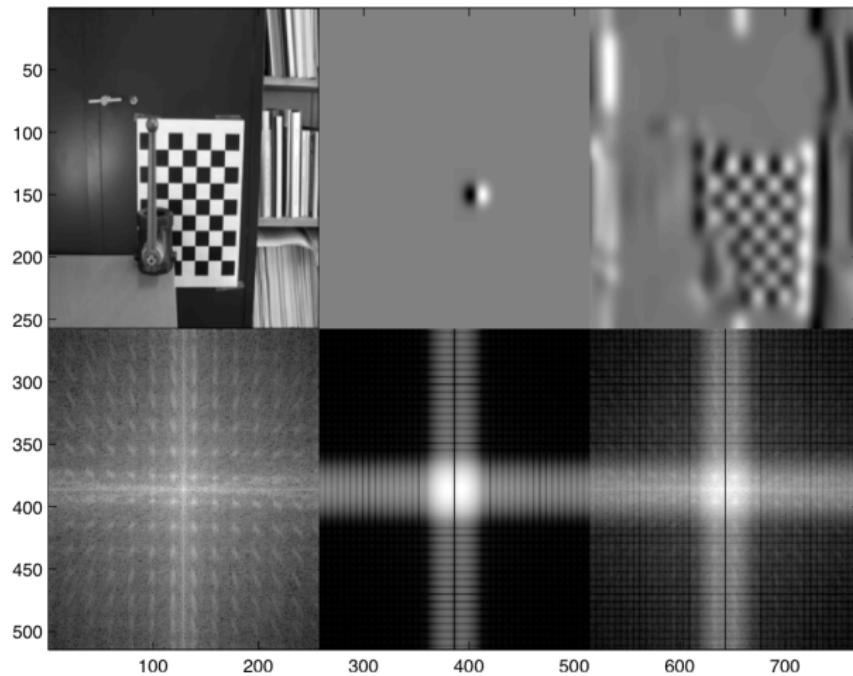
Filter example: Differentiation and Gaussian in a general direction



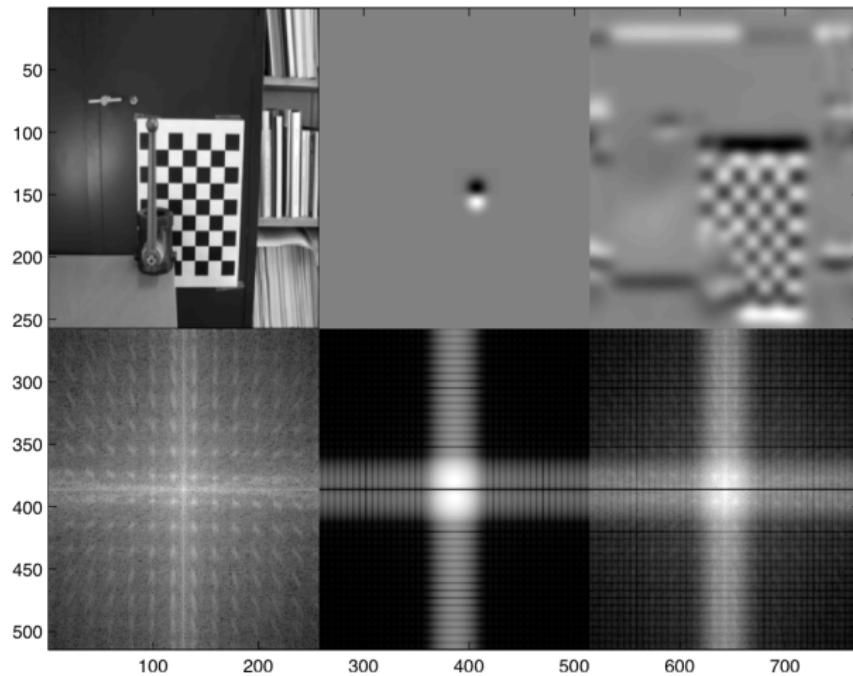
Filter example: Increasing σ : Filter?



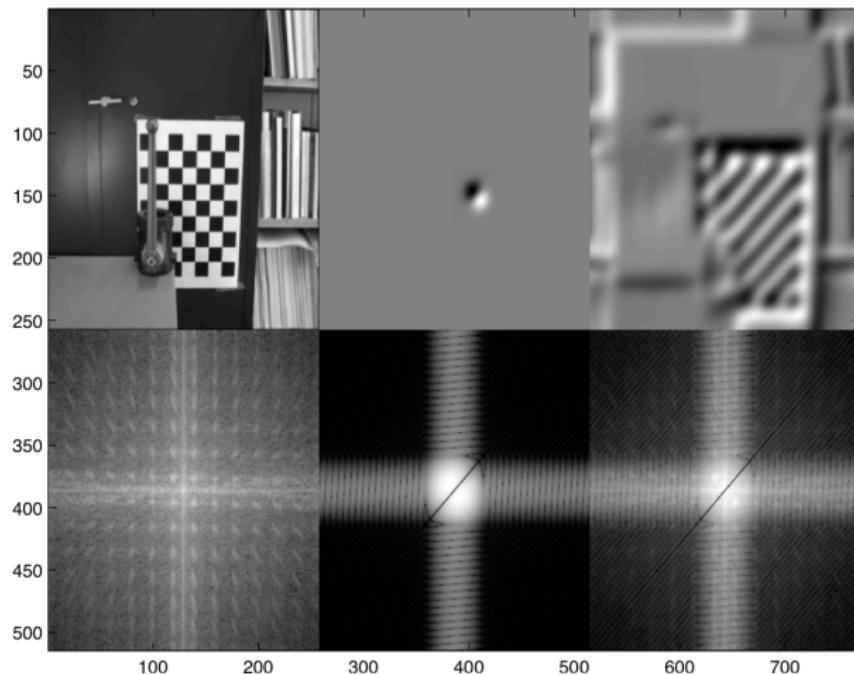
Filter example: Increasing σ : Der in y-dir.



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

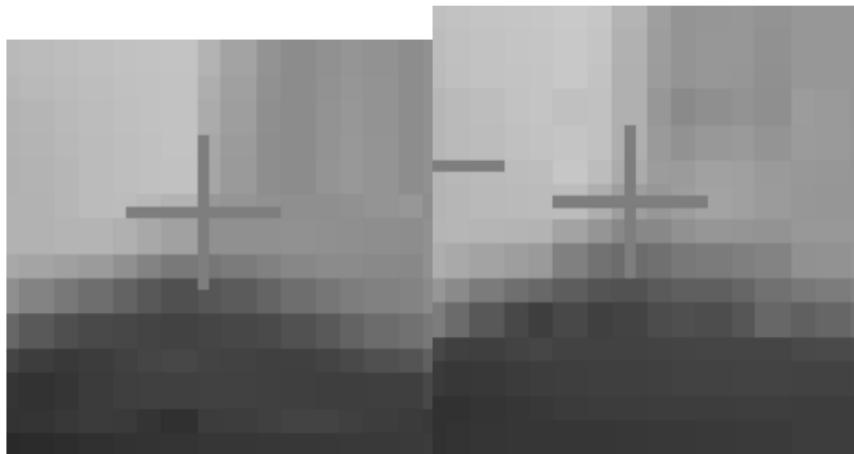
Let $\Omega_A \subset \mathbb{R}^2$ be a neighborhood of a feature point in an image with intensities I_A . Similarly let Ω_B and I_B be the neighborhood and the intensities around a feature point in another image.

Point correspondences from correlation

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

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

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



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

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

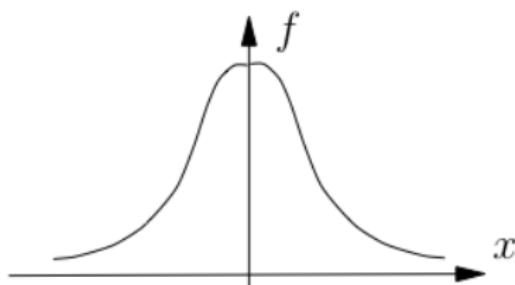
```
patch = image1(rpos1(2)-s:rpos1(2)+s,rpos1(1)-s:rpos1(1)+s);
patch = flipud(fliplr(patch));
AA = sum(sum(patch.*patch));
BBt = image2.*image2;
BB = conv2(BBt,ones(size(patch)));
disp(['Convolving image']);
AB = conv2(image2,patch);
disp(['Calculating correlation']);
res = AA - 2*AB + BB;
disp(['Finding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

Gaussian 'interpolation'

Interpolate with

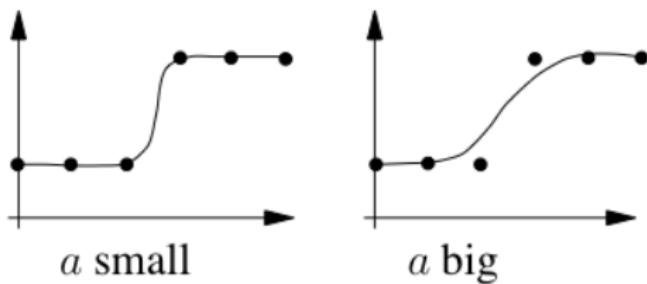
$$f(x) = \sum_k e^{-(x-k)^2/a^2} f(k),$$

where a determines the 'scale'



The Gaussian pyramid

Gives a "pyramid" of the same image in different scales



Scale space pyramid

Alternatively: Interpolate first with sinc and then with Gauss,
 G_a :

$$f(x) = (f * \text{sinc}) * G_a = f * (\text{sinc} * G_a).$$

Advantage: $\text{sinc} * G_a$ has small lobes.

For large a (larger than 1):

$$\text{sinc} * G_a \approx G_a$$

Discrete versions of continuous filters

$f(x, y)$ =image (x, y discrete variables)

$F(u, v)$ =DFT(f) (u, v discrete variables)

$h(x, y)$ =discrete filter (x, y discrete variables)

$H(u, v)$ =DFT(h) (u, v discrete variables)

Convolution:

$$h * f = \sum_{k,l} h(x - k, y - l) f(k, l)$$

Two methods:

1. **In the signal plane:** discretize h

$$g = h * f$$

2. **In the frequency plane:** discretize H

$$g = IDFT(HF)$$

Example:

Interpolation can be used to calculate 'real' differentiation in images. This forms the basis for "sub-pixel" feature extraction.

Idea:

1. Interpolate the image (with your favorite method).
2. Differentiate.
3. Sample the differentiated signal.

Example: Differentiation

$$f(x) = \sum_i f(i)h(x - i)$$

$$\frac{df}{dx}(x) = \sum_i f(i) \frac{dh}{dx}(x - i)$$

$$\frac{df}{dx}(j) = \sum_i f(i) \frac{dh}{dx}(j - i) = f * h$$

with

$$g(i) = \frac{dh}{dx}(i)$$

i.e. convolute with g a sampled version of the differentiated interpolation function.

Discrete operators

More generally, every operator that is linear and translation invariant, but defined for continuous functions can be mapped to a convolution in the discrete case.

Idea: Operator A (continuous)

1. Interpolate the image (with your favorite method).
2. Use the operator (defined for continuous signals).
3. Sample the resulting signal.

=

Operator A (discrete)

1. Convolute with the sampled version of the interpolated version of the operator.

Reducing noise sensitivity

Sometimes one adds a bit of smoothing as well.

Idea: Operator B (continuous)

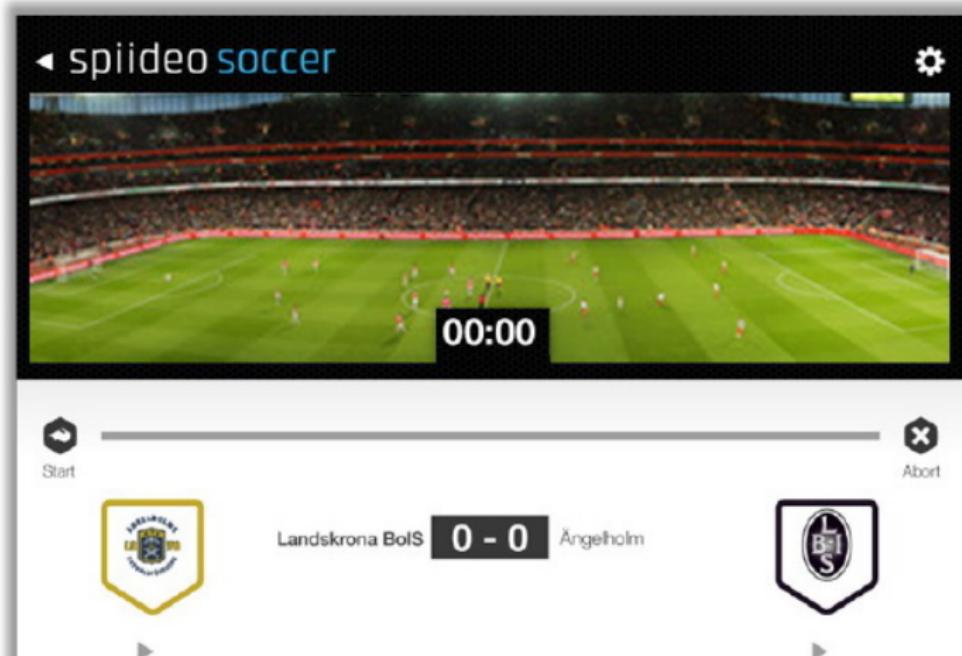
1. Interpolate the image (with your favorite method).
2. Smooth the signal.
3. Use the operator (defined for continuous signals).
4. Sample the resulting signal.

=

Operator B (discrete)

1. Convolute with the sampled version of the interpolated version of the smoothed operator.

Today's masters thesis suggestion: Automatic recording and event management in team sports



Today's masters thesis suggestion: Automatic recording and event management in team sports

In team sports more and more effort is put into recording and analyzing practice and matches. In current solutions a normal video camera, with a person live recording, is used. Post recording the video is tagged with interesting events to make search replay efficient and interesting. This can be done easier and more efficient using modern fix network cameras. Multiple cameras is typically needed to cover the entire field and to make that easily available to everyday users a simple automated calibration of the setup is needed.

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
- ▶ Convolution theorem $f * g \rightarrow F(u, v)G(u, v)$. Note that this is NOT matrix multiplication!!!
- ▶ Understanding convolutions
- ▶ Convolutions to find patterns
- ▶ Sampling + continuous operator + interpolation = discrete convolution
- ▶ Scale space