Image Analysis

3. Fourier Transform

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute for Business Economics and Information Systems
& Institute for Computer Science
University of Hildesheim
http://www.ismll.uni-hildesheim.de

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
1/65

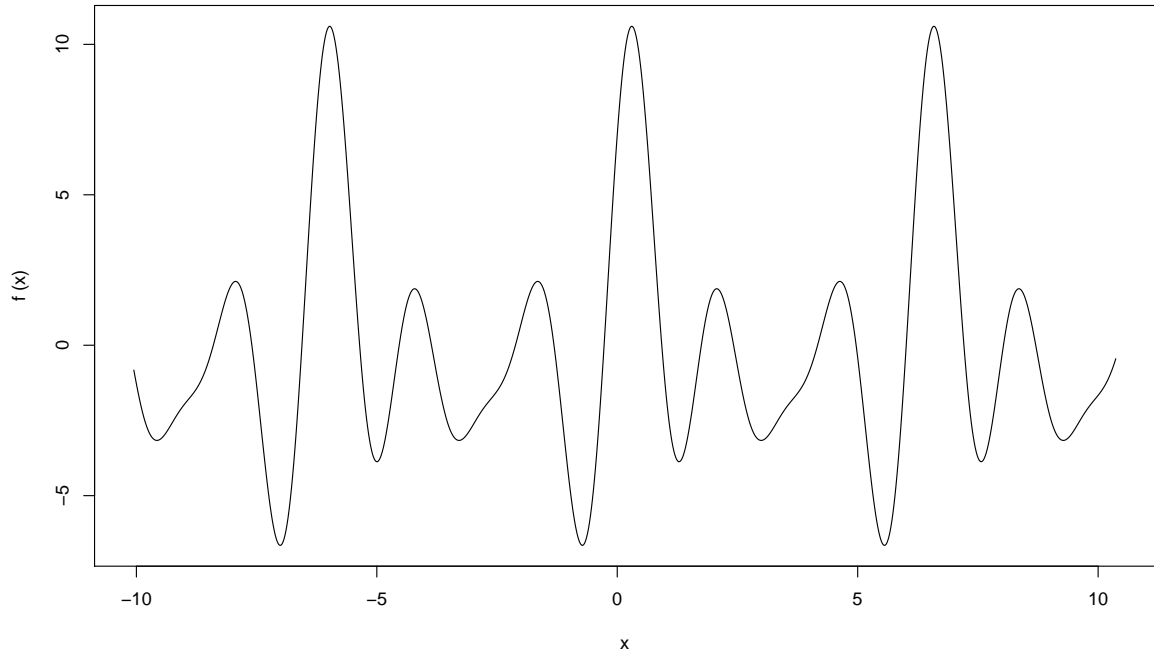## 1. Fourier Series Representation

2. The Fourier Transform
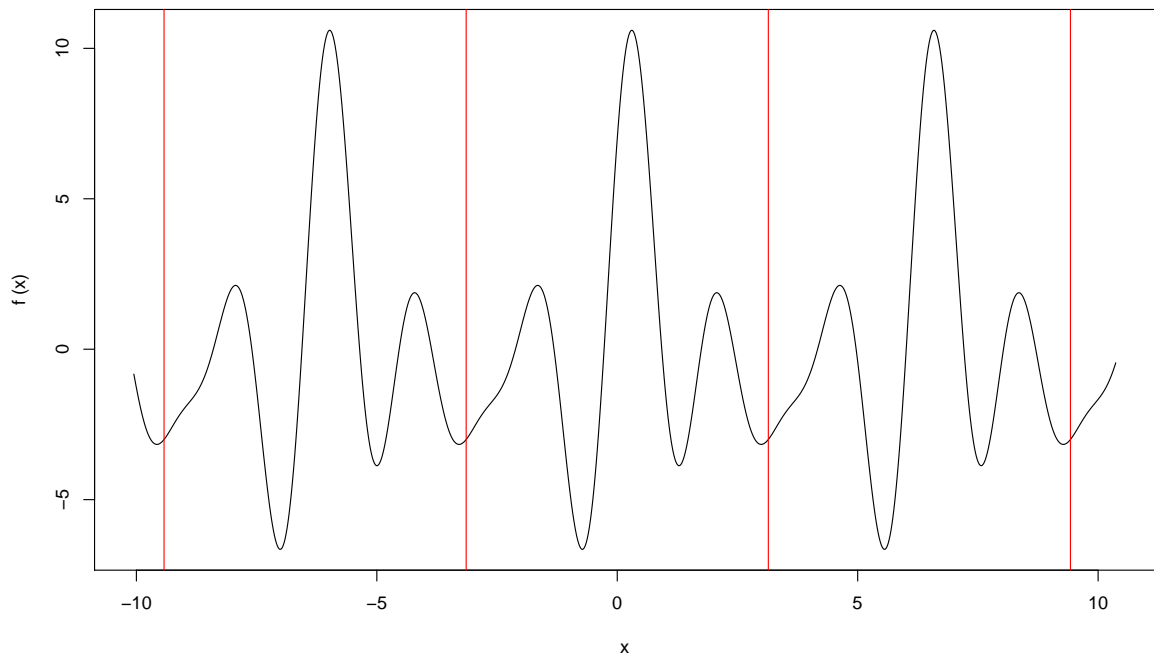
3. Discrete Signals

4. Discrete Fourier Transform

5. Two-dimensional Fourier Transforms

6. Applications

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
1/65

Periodic Functions

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
1/65

Periodic Functions

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
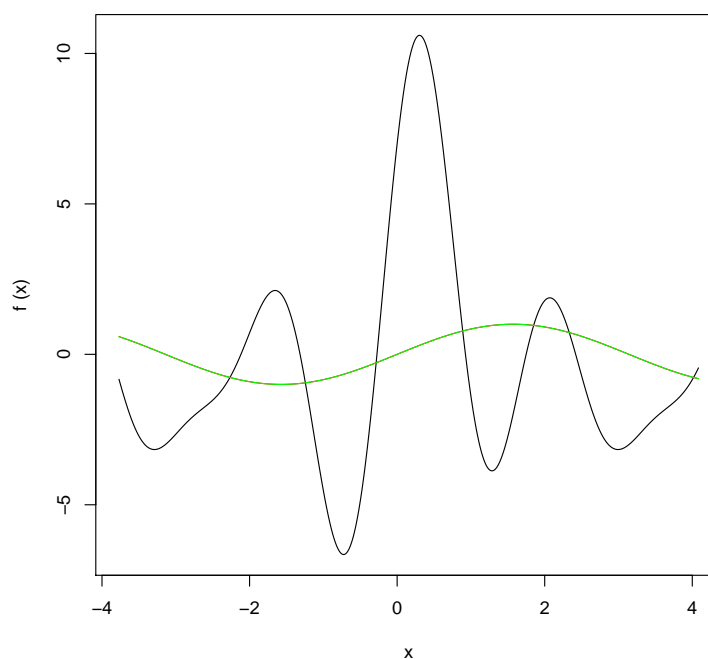Course on Image Analysis, winter term 2011/12
1/65

Periodic Functions

A function $f : \mathbb{R} \to \mathbb{C}$ is called T-**periodic** if

$$f(x + T) = f(x) \quad \forall x \in \mathbb{R}$$

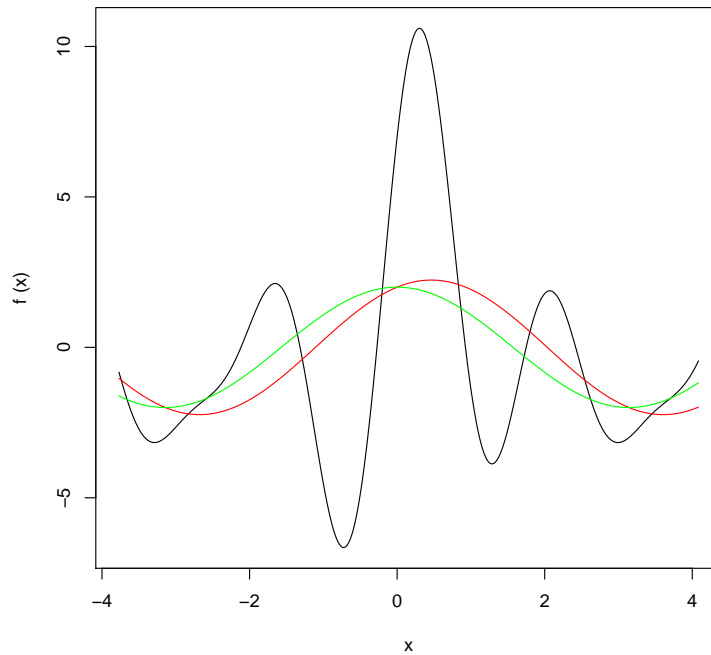Example: the functions $\sin$ and $\cos$ are 2π-periodic.

Example: the function $\sin(\omega x)$ is $2\pi/\omega$-periodic.

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
2/65

Periodic Functions / Approximation

$$\hat{f}(x) = 1 \sin(x)$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
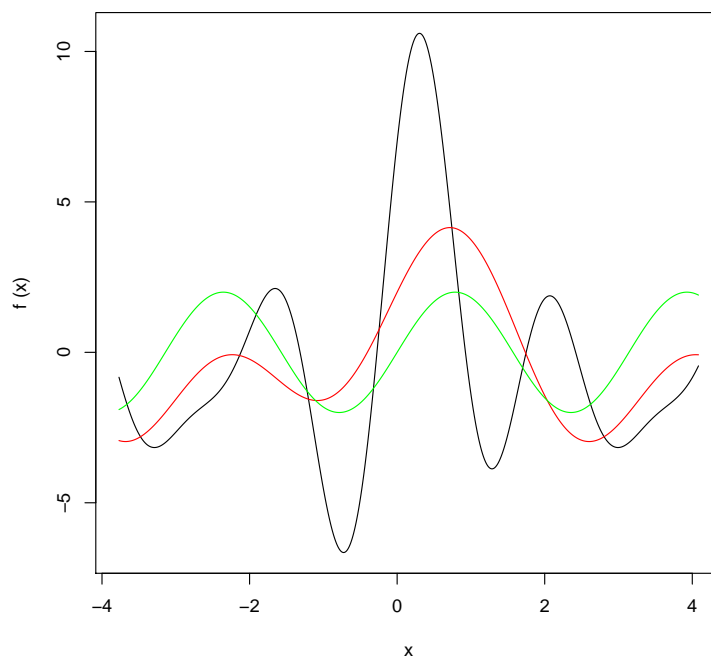2/65

Periodic Functions / Approximation

$$\hat{f}(x) = 1\sin(x) + 2\cos x$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
2/65

Periodic Functions / Approximation

$$\hat{f}(x) = 1\sin(x) + 2\cos x + 2\sin 2x$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
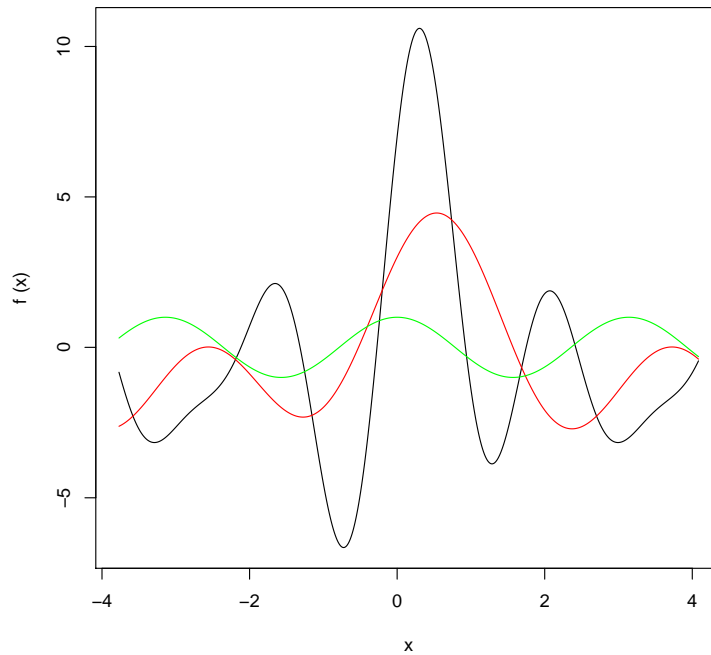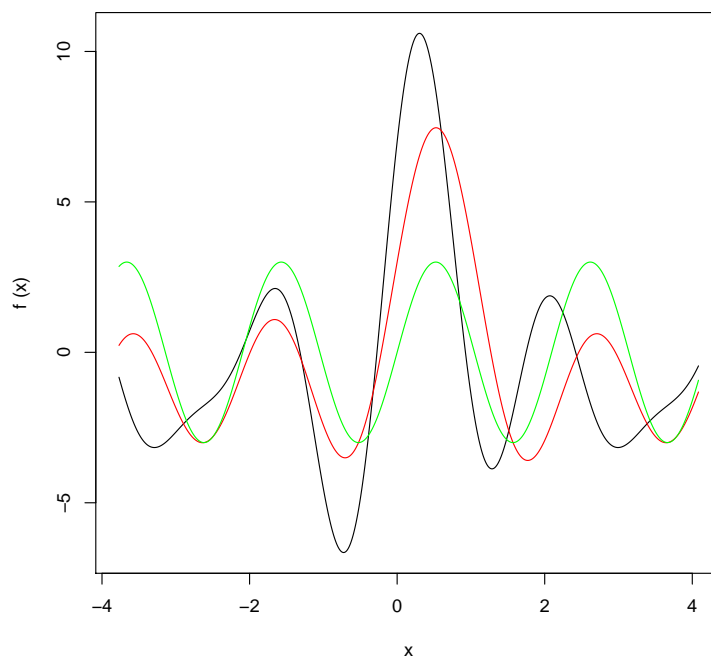2/65

Periodic Functions / Approximation

$$\hat{f}(x) = 1\sin(x) + 2\cos x + 2\sin 2x + \textcolor{green}{1\cos 2x}$$

Periodic Functions / Approximation

$$\hat{f}(x) = 1\sin(x) + 2\cos x + 2\sin 2x + 1\cos 2x + \textcolor{green}{3\sin 3x}$$

Periodic Functions / Approximation

$$\hat{f}(x) = 1\sin(x) + 2\cos x + 2\sin 2x + 1\cos 2x + 3\sin 3x + 3\cos 3x$$

Periodic Functions / Approximation

$$\hat{f}(x) = 1\sin(x) + 2\cos x + 2\sin 2x + 1\cos 2x + 3\sin 3x + 3\cos 3x + 2\sin 4x$$

Periodic Functions / Approximation

$$\hat{f}(x) = 1\sin(x) + 2\cos x + 2\sin 2x + 1\cos 2x + 3\sin 3x + 3\cos 3x + 2\sin 4x + 1\cos 4x$$

Fourier Series Representation

Theorem (Fourier Series Representation). Any continuous, differentiable and T-periodic function $f : \mathbb{R} \to \mathbb{C}$ can be written as

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos k\omega x + b_k \sin k\omega x, \quad \omega := \frac{2\pi}{T}$$

with coefficients $a_k, b_k \in \mathbb{C}$, called a **Fourier Series of** f.

How to compute Fourier coefficients a_k, b_k for a given function f ?

Jean Baptiste Joseph Fourier (1768–1830),
French Mathematician and Physicist

Note. Actually, the class of functions that can be represented as Fourier series is much larger (see, e.g., [Kö90, p. 314]).

Trigonometric Addition Formulas

Lemma (Trigonometric Addition Formulas). For all $x, y \in \mathbb{R}$:

$$\cos(x + y) = \cos x \cos y - \sin x \sin y$$
$$\sin(x + y) = \sin x \cos y + \sin y \cos x$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
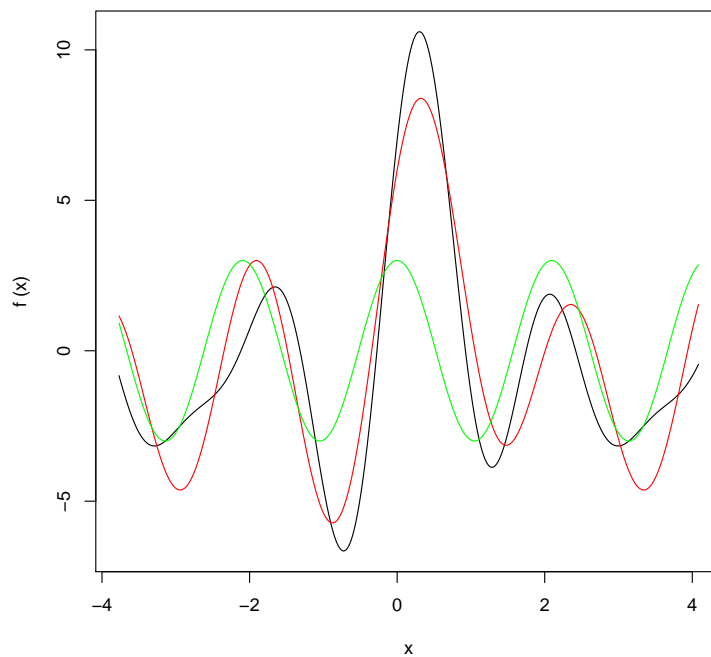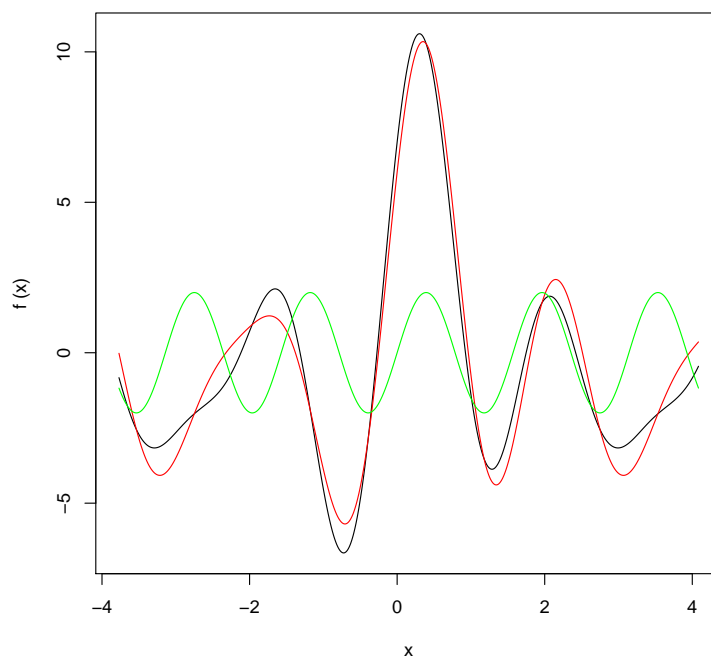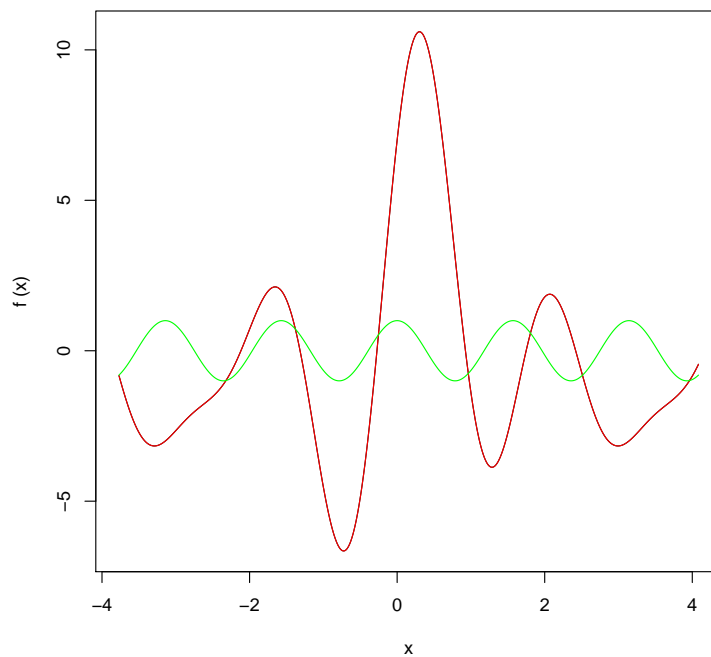Course on Image Analysis, winter term 2011/12
4/65

Some Trigonometric Integrals

$$\int \cos ax \cos bx \, dx = \begin{cases} \dfrac{1}{2}\left(\dfrac{\sin(a+b)x}{a+b} + \dfrac{\sin(a-b)x}{a-b}\right), & \text{if } a \neq b \\[2ex] \dfrac{\sin(2ax) + 2ax}{4a}, & \text{else} \end{cases}$$

$$\int \sin ax \sin bx \, dx = \begin{cases} -\dfrac{1}{2}\left(\dfrac{\sin(a+b)x}{a+b} - \dfrac{\sin(a-b)x}{a-b}\right), & \text{if } a \neq b \\[2ex] -\dfrac{\sin(2ax) - 2ax}{4a}, & \text{else} \end{cases}$$

$$\int \sin ax \cos bx \, dx = \begin{cases} -\dfrac{1}{2}\left(\dfrac{\cos(a+b)x}{a+b} + \dfrac{\cos(a-b)x}{a-b}\right), & \text{if } a \neq b \\[2ex] \dfrac{\sin^2(ax)}{2a}, & \text{else} \end{cases}$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
5/65

Some Trigonometric Integrals

The six formulas easily can be proven by differentiation, e.g.,

$$\int \cos ax \cos bx \, dx \stackrel{?}{=} \frac{1}{2} \left(\frac{\sin(a+b)x}{a+b} + \frac{\sin(a-b)x}{a-b} \right)$$

for $a \neq b$. Derivation $\frac{d}{dx}$ yields:

$$
\begin{aligned}
\cos ax \cos bx &\stackrel{?}{=} \frac{1}{2} \left(\frac{(a+b)\cos(a+b)x}{a+b} + \frac{(a-b)\cos(a-b)x}{a-b} \right) \\
&= \frac{1}{2} \left(\cos(a+b)x + \cos(a-b)x \right) \\
&= \frac{1}{2} \left(\cos ax \cos bx - \sin ax \sin bx + \cos ax \cos(-bx) - \sin ax \sin(-bx) \right) \\
&= \frac{1}{2} \left(\cos ax \cos bx - \sin ax \sin bx + \cos ax \cos bx + \sin ax \sin bx \right) \\
&= \cos ax \cos bx
\end{aligned}
$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
6/65

Trigonometric Orthogonality Relations

Let $\omega \in \mathbb{R}^+$. The functions

$$\{\sin k\omega x \mid k \in \mathbb{N}, k > 0\} \cup \{\cos k\omega x \mid k \in \mathbb{N}, k > 0\}$$

are pairwise othogonal with respect to

$$\langle f, g \rangle := \int_{-\pi/\omega}^{+\pi/\omega} f(x)g(x)dx$$

i.e., for any two distinct such functions f, g

$$\langle f, g \rangle = \int_{-\pi/\omega}^{+\pi/\omega} f(x)g(x)dx = 0$$

but

$$\langle f, f \rangle = \frac{\pi}{\omega} \neq 0$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
7/65

Trigonometric Orthogonality Relations

Proof: let $f = \cos k\omega x$ and $g = \cos l\omega x$ with $k \neq l$, then:

$$\langle f, g \rangle = \int_{-\pi/\omega}^{+\pi/\omega} \cos k\omega x \cos l\omega x \, dx = \left[\frac{1}{2} \left(\frac{\sin(k+l)\omega x}{(k+l)\omega} + \frac{\sin(k-l)\omega x}{(k-l)\omega} \right) \right]_{-\pi/\omega}^{+\pi/\omega}$$

$$= \frac{1}{2} \left(\frac{\sin(k+l)\pi}{(k+l)\omega} + \frac{\sin(k-l)\pi}{(k-l)\omega} - \frac{\sin -(k+l)\pi}{(k+l)\omega} - \frac{\sin -(k-l)\pi}{(k-l)\omega} \right)$$

$$= \frac{\sin(k+l)\pi}{(k+l)\omega} + \frac{\sin(k-l)\pi}{(k-l)\omega} = 0$$

but

$$\langle f, f \rangle = \left[\frac{\sin(2k\omega x) + 2k\omega x}{4k\omega} \right]_{-\pi/\omega}^{+\pi/\omega} = \frac{\sin(2k\pi) + 2k\pi}{4k\omega} - \frac{\sin(-2k\pi) + (-2k\pi)}{4k\omega} = \frac{\pi}{\omega}$$

Fourier Series Representation

Theorem (Fourier Series Representation). Any continuous, differentiable and T-periodic function $f : \mathbb{R} \to \mathbb{C}$ can be written as

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos k\omega x + b_k \sin k\omega x, \quad \omega := \frac{2\pi}{T}$$

with coefficients $a_k, b_k \in \mathbb{C}$, called a **Fourier Series of** f.

How to compute Fourier coefficients a_k, b_k for a given function f?

$$a_k = \frac{1}{\pi/\omega} \int_{-\pi/\omega}^{+\pi/\omega} f(x) \cos k\omega x \, dx$$

$$b_k = \frac{1}{\pi/\omega} \int_{-\pi/\omega}^{+\pi/\omega} f(x) \sin k\omega x \, dx$$

Note. Actually, the class of functions that can be represented as Fourier series is much larger (see, e.g., [Kö90, p. 314]).
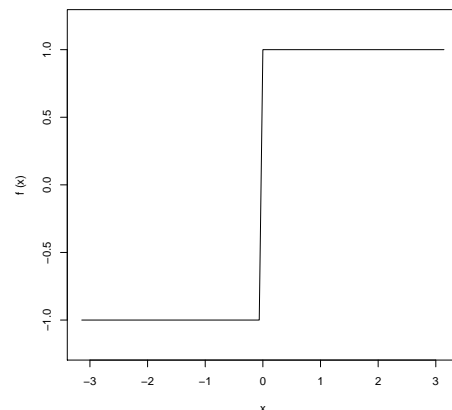
Fourier Series Representation

Proof.

$$a_k \stackrel{?}{=} \frac{1}{\pi/\omega} \int_{-\pi/\omega}^{+\pi/\omega} f(x) \cos k\omega x \, dx$$

$$= \frac{1}{\pi/\omega} \int_{-\pi/\omega}^{+\pi/\omega} \left(\frac{a_0}{2} + \sum_{l=1}^{\infty} a_l \cos l\omega x + b_l \sin l\omega x \right) \cos k\omega x \, dx$$

$$= \frac{1}{\pi/\omega} \left(\int_{-\pi/\omega}^{+\pi/\omega} \frac{a_0}{2} \cos k\omega x \, dx + \sum_{l=1}^{\infty} \int_{-\pi/\omega}^{+\pi/\omega} a_l \cos l\omega x \cos k\omega x \, dx \right.$$

$$\left. + \int_{-\pi/\omega}^{+\pi/\omega} b_l \sin l\omega x \cos k\omega x \, dx \right)$$

$$= \frac{1}{\pi/\omega} \frac{\pi}{\omega} a_k$$

$$= a_k$$

Fourier Representation / Rectangular function

Let f be the 2π-periodic rectangular function

$$f(x) = \begin{cases} -1, & \text{if } x \in (-\pi, 0) \\ 0, & \text{if } x \in \{-\pi, 0, \pi\} \\ +1, & \text{if } x \in (0, \pi) \end{cases}$$

The Fourier representation of f is

$$f(x) = \frac{4}{\pi} \sum_{k \in \mathbb{N} \text{ odd}} \frac{\sin kx}{k}$$

Proof:

$$b_k = \frac{2}{\pi} \int_{-\pi}^{+\pi} f(x) \sin kx \, dx = \frac{2}{\pi} 2 \int_0^{+\pi} \sin kx \, dx = \frac{4}{\pi} \left[-\frac{1}{k} \cos kx \right]_0^{\pi} = \begin{cases} -\frac{4}{\pi k}(0-1) = \frac{4}{\pi k}, & \text{if } k \text{ odd} \\ -\frac{4}{\pi k}(1-1) = 0, & \text{if } k \text{ even} \end{cases}$$

$$a_k = \frac{2}{\pi} \int_{-\pi}^{+\pi} f(x) \cos kx \, dx = 0$$

⤳ in general, Fourier representations are infinite as in this example!

Fourier Representation / Rectangular function
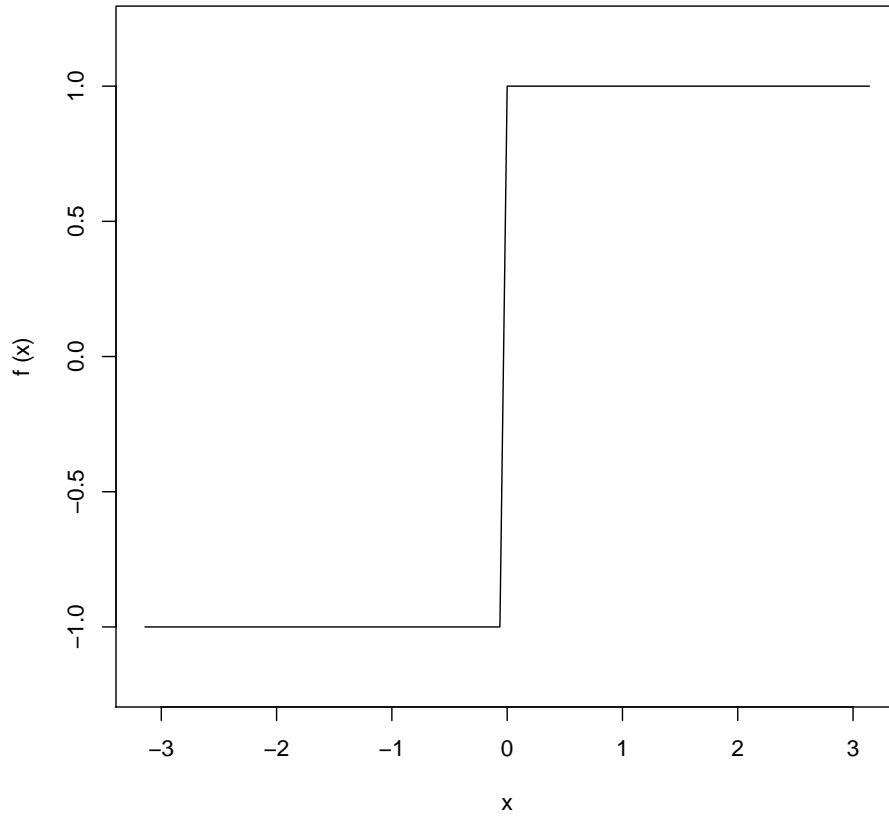
Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
12/65

Fourier Representation / Rectangular function / $k = 1$
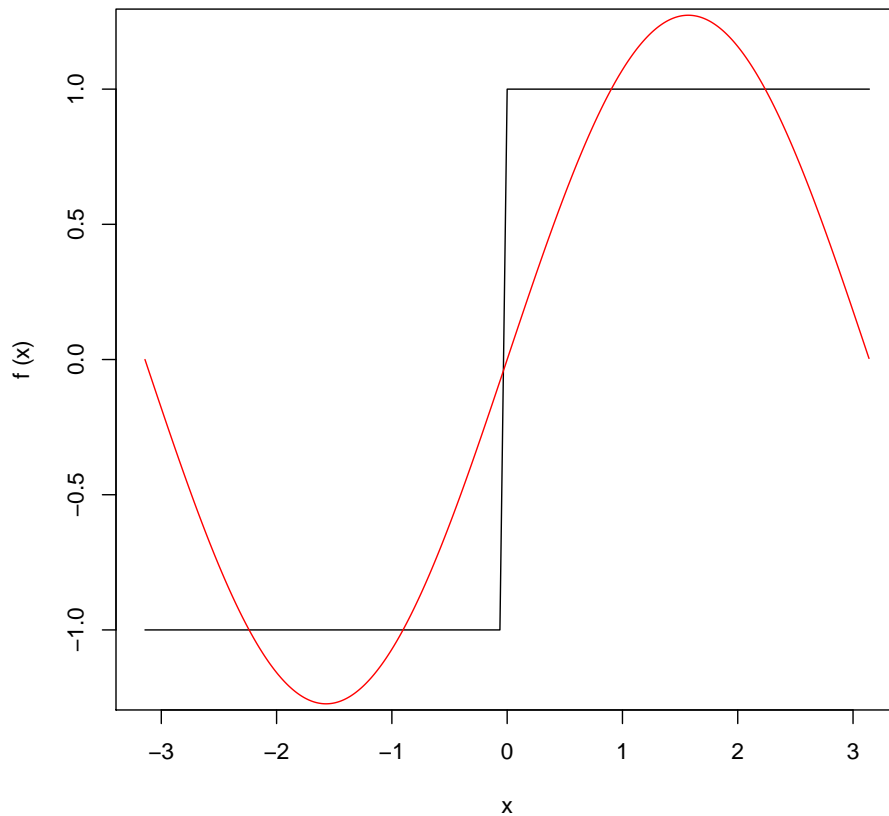
Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
12/65

Fourier Representation / Rectangular function / $k = 5$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
12/65

Fourier Representation / Rectangular function / $k = 21$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
12/65

Eulers Formula

$$\cos x := \sum_{n \in \mathbb{N}} (-1)^n \cdot \frac{x^{2n}}{(2n)!} \qquad = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$

$$\sin x := \sum_{n \in \mathbb{N}} (-1)^n \cdot \frac{x^{2n+1}}{(2n+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

$$\exp x := \sum_{n \in \mathbb{N}} \frac{x^n}{n!} \qquad\qquad = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$$

Lemma (**Eulers formula**). For $x \in \mathbb{C}$:

$$e^{ix} = \cos(x) + i \cdot \sin(x), \quad \text{with } i := \sqrt{-1} \text{ the imaginary unit}$$

Proof:

$$e^{ix} = \sum_{n \in \mathbb{N}} \frac{(ix)^n}{n!}$$

$$= \sum_{n \in \mathbb{N}} \frac{(ix)^{2n}}{(2n)!} + \sum_{n \in \mathbb{N}} \frac{(ix)^{2n+1}}{(2n+1)!}$$

$$= \sum_{n \in \mathbb{N}} (-1)^n \frac{x^{2n}}{(2n)!} + i \cdot \sum_{n \in \mathbb{N}} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$
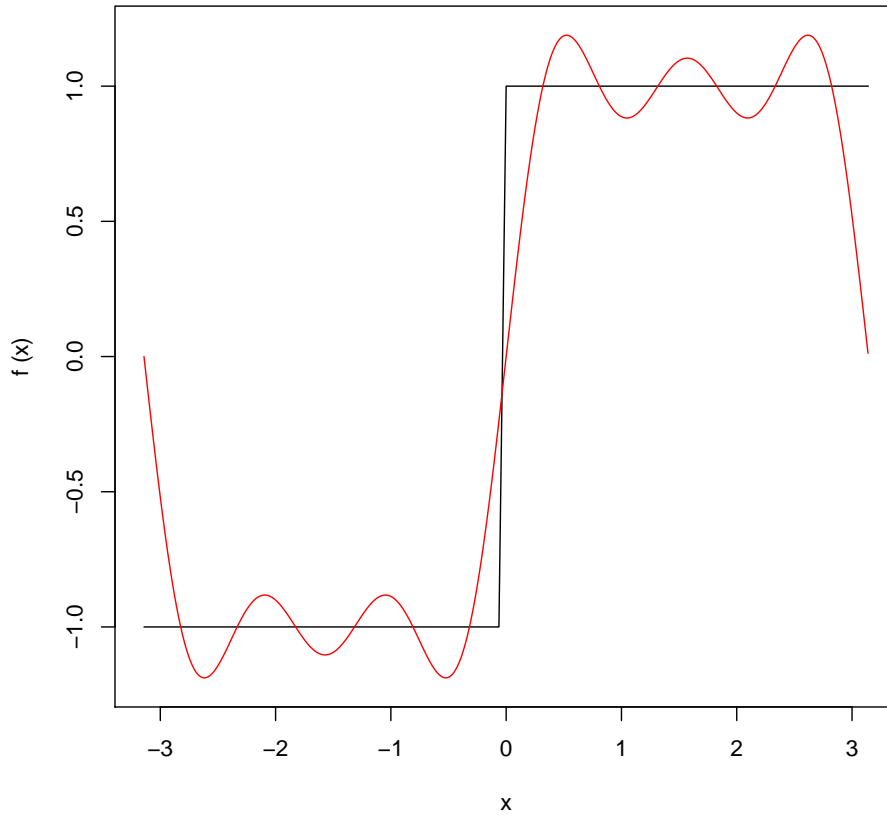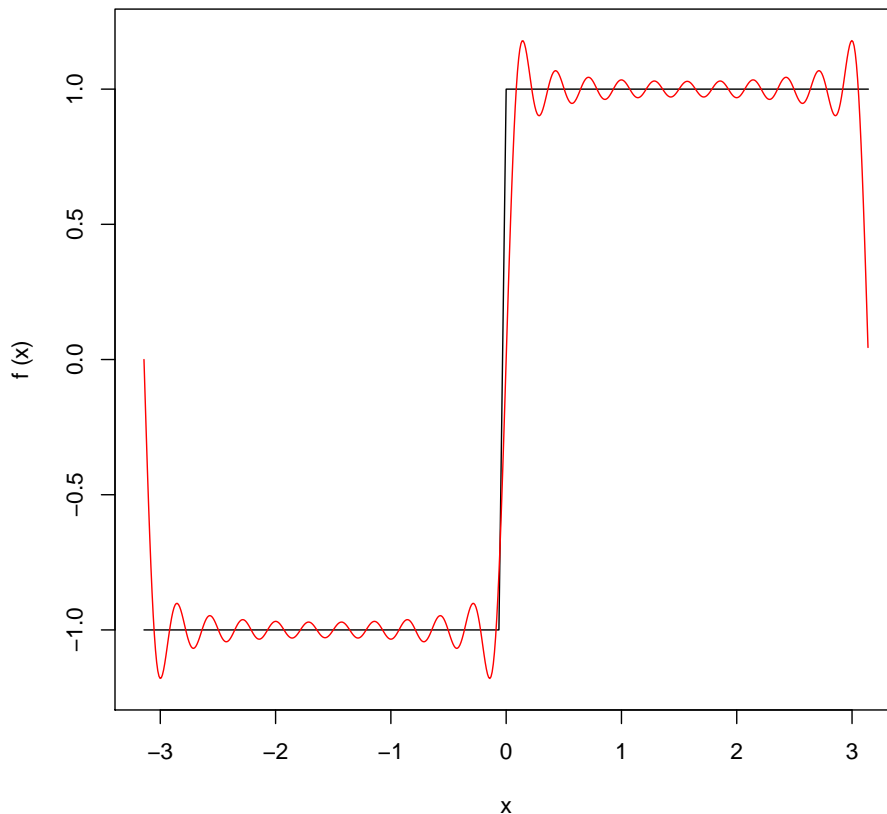
$$= \cos(x) + i \sin(x)$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
13/65

Complex Fourier Series Representation

Theorem (Complex Fourier Series Representation). Any continuous, differentiable and T-periodic function $f : \mathbb{R} \to \mathbb{C}$ can be written as

$$f = \sum_{k \in \mathbb{Z}} c_k \, e^{ik\omega x}, \quad \omega := \frac{2\pi}{T}$$

with coefficients $c_k \in \mathbb{C}$, called a **Fourier Series of** f.

The coefficients of the complex Fourier series can be computed via

$$c_k = \frac{1}{2\pi/\omega} \int_{-\pi/\omega}^{+\pi/\omega} f(x) \, e^{-ik\omega x} dx$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
14/65

Complex Fourier Series Representation

Proof.

$$f = \sum_{k \in \mathbb{Z}} c_k \, e^{ik\omega x}$$

$$= \sum_{k \in \mathbb{Z}} c_k \left(\cos k\omega x + i \sin k\omega x \right)$$

$$= c_0 + \sum_{k=1}^{\infty} (c_k + c_{-k}) \, \cos k\omega x + (c_k - c_{-k}) i \sin k\omega x$$

$$= \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos k\omega x + b_k \sin k\omega x$$

with

$$a_0 = 2c_0, \quad a_k = c_k + c_{-k}, \quad b_k = i(c_k - c_{-k})$$

and vice versa via:

$$c_0 = \frac{a_0}{2}, \quad c_k = \frac{1}{2}(a_k - ib_k), \quad c_{-k} = \frac{1}{2}(a_k + ib_k)$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
15/65

Image Analysis

1. Fourier Series Representation

2. The Fourier Transform

3. Discrete Signals

4. Discrete Fourier Transform

5. Two-dimensional Fourier Transforms

6. Applications

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
16/65

Fourier Transform

Let $f : \mathbb{R} \to \mathbb{C}$ be a function (that satisfies some regularity conditions). Then

$$
\begin{aligned}
F : \quad \mathbb{R} &\to \mathbb{C} \\
\omega &\mapsto \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(x) e^{-i\omega x} dx
\end{aligned}
$$

exists for each ω and is a continuous function called **Fourier Transform of f** (aka **Fourier spectrum of f**).

One can show that (if F also satisfies some regularity conditions):

$$
f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} F(\omega) e^{i\omega x} d\omega
$$

This is called **Inverse Fourier Transform**.

We will write $\mathcal{F}(f) := F$ for the Fourier transform of a function f and $\mathcal{F}^{-1}(F)$ for the inverse Fourier transform of a function F.

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
16/65

Fourier Transforms / Examples / Gaussian

$$
f(x) = \frac{1}{\sigma} \cdot e^{-\frac{x^2}{2\sigma^2}}
$$

$$
F(x) = e^{-\frac{\sigma^2 x^2}{2}}
$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
17/65
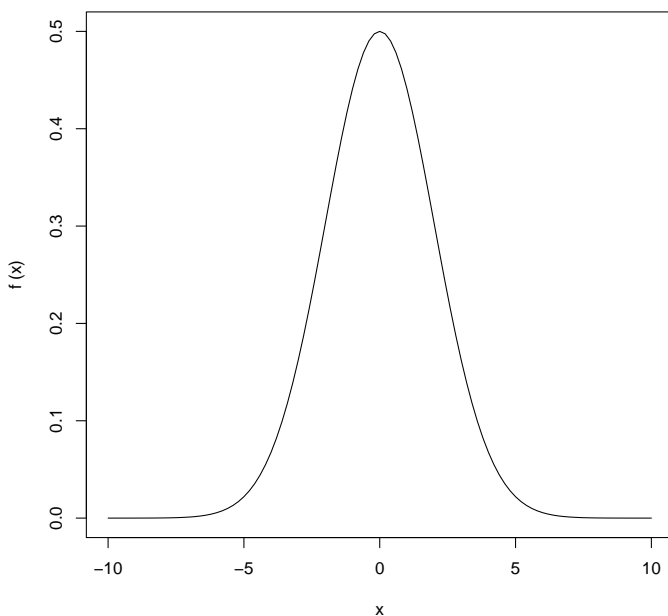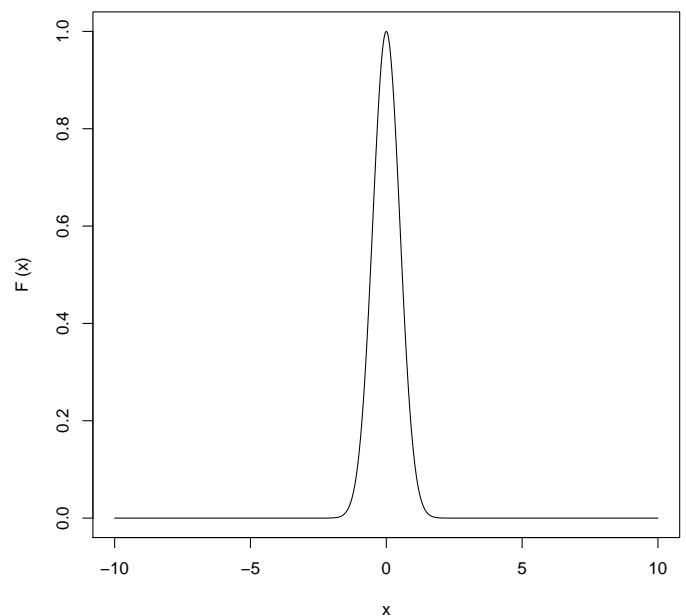
Fourier Transforms / Examples / Uniform
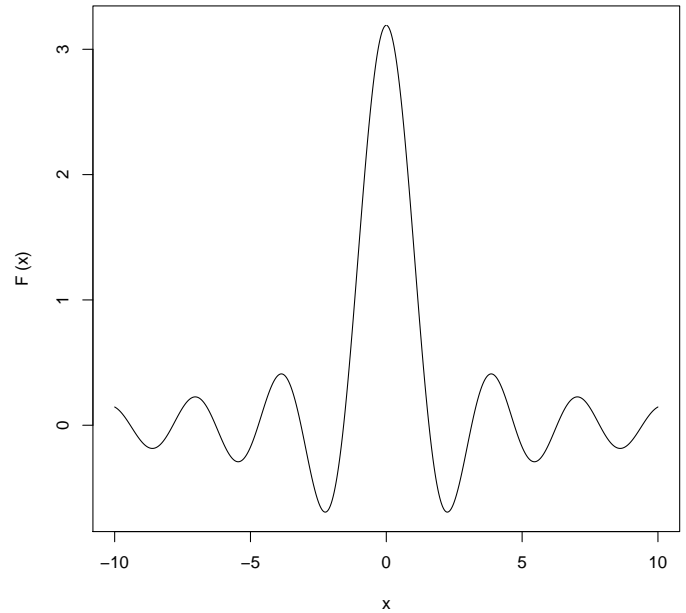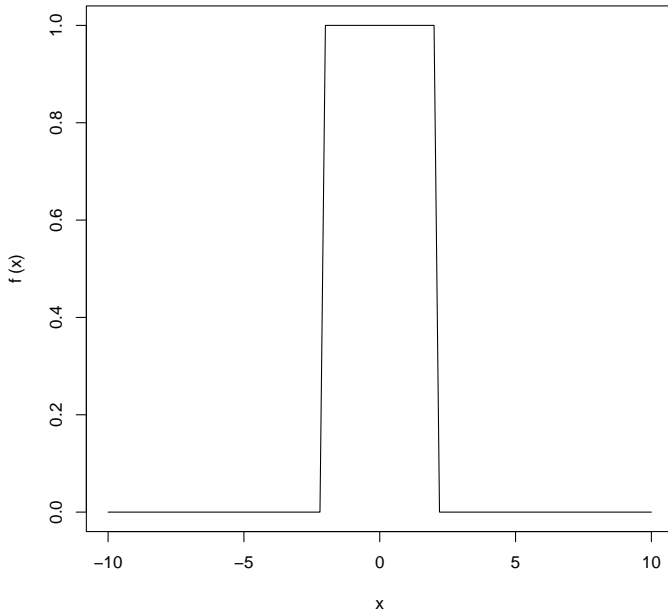
$$f(x) = \delta(x \in [-b, b])$$

$$F(x) = \frac{2b \sin(bx)}{\sqrt{2\pi} x}$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
18/65

Fourier Transforms / Examples / Cosine

$$f(x) = \cos(\omega x)$$

$$F(x) = \sqrt{\frac{\pi}{2}}(\delta(x - \omega) + \delta(x + \omega))$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
19/65

Fourier Transforms / Examples / Sine

$$f(x) = \sin(\omega x)$$

$$F(x) = i \cdot \sqrt{\frac{\pi}{2}}(\delta(x - \omega) - \delta(x + \omega))$$

Properties

function h	Fourier transform H	property name
f	F	
F	$f(-x)$	inverse
$af + bg$	$aF + bG$	linearity
$f * g$	$F(x)G(x)$	convolution
$f(x)g(x)$	$\frac{1}{2\pi}F * G$	multiplication
$f(x - a)$	$e^{-iax}F(x)$	translation
$e^{iax}f(x)$	$F(x - a)$	modulation
$f(x/a)$	$|a|F(ax)$	scaling
f^*	$F^*(-x)$	complex conjugate
$f(x) \in \mathbb{R}$	$F(-x) = F^*(x)$	hermitian symmetry

1. Fourier Series Representation

2. The Fourier Transform

3. Discrete Signals

4. Discrete Fourier Transform

5. Two-dimensional Fourier Transforms

6. Applications

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
22/65

Dirac Comb

The symbol

$$\Delta_T(x) := \sum_{n \in \mathbb{Z}} \delta(x - Tn)$$

is called **Dirac comb** (aka **impulse train**, **sampling function**, **Shah function**) with sampling interval T.
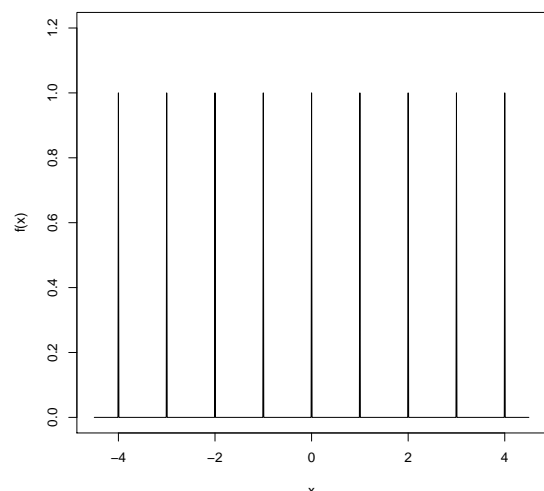
Lemma. The Fourier series of the Dirac comb Δ_T is

$$\Delta_T(x) = \frac{1}{T} \sum_{k \in \mathbb{Z}} e^{-i \frac{2\pi k}{T} x}$$

and its Fourier transform

$$\mathcal{F}(\Delta_T)(x) = \frac{1}{T} \cdot \Delta_{2\pi/T}(x) = \frac{1}{T} \sum_{n \in \mathbb{Z}} \delta(x - \frac{2\pi}{T} n)$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
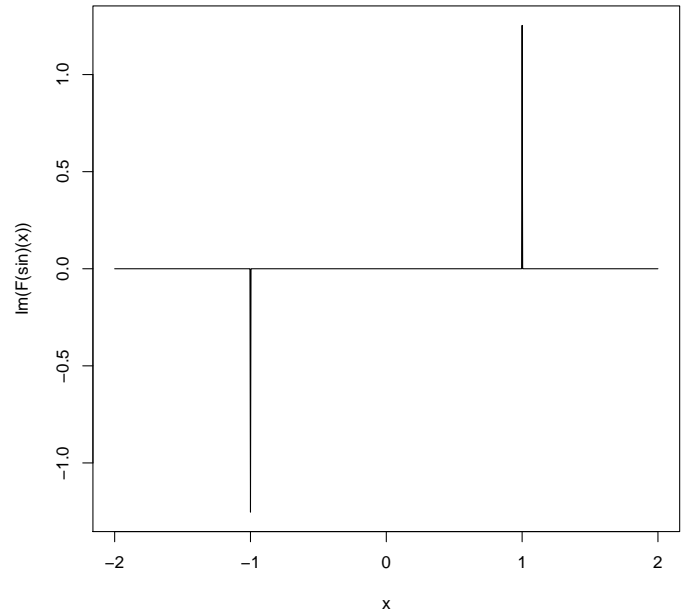Course on Image Analysis, winter term 2011/12
22/65

Dirac Comb

Proof: Obviously $\triangle_T$ is periodic with period T. Therefore

$$f(x) = \sum_{k \in \mathbb{Z}} c_k \, e^{-i\frac{2\pi k}{T}x}$$

with

$$
\begin{aligned}
c_k &= \frac{1}{T} \int_x^{x+T} \triangle_T(y) \, e^{-i\frac{2\pi k}{T}y} dy \\
&= \frac{1}{T} \int_{-T/2}^{+T/2} \triangle_T(y) \, e^{-i\frac{2\pi k}{T}y} dy \\
&= \frac{1}{T} \int_{-T/2}^{+T/2} \delta(y) \, e^{-i\frac{2\pi k}{T}y} dy \\
&= \frac{1}{T} \, e^{-i\frac{2\pi k}{T}0} \\
&= \frac{1}{T}
\end{aligned}
$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
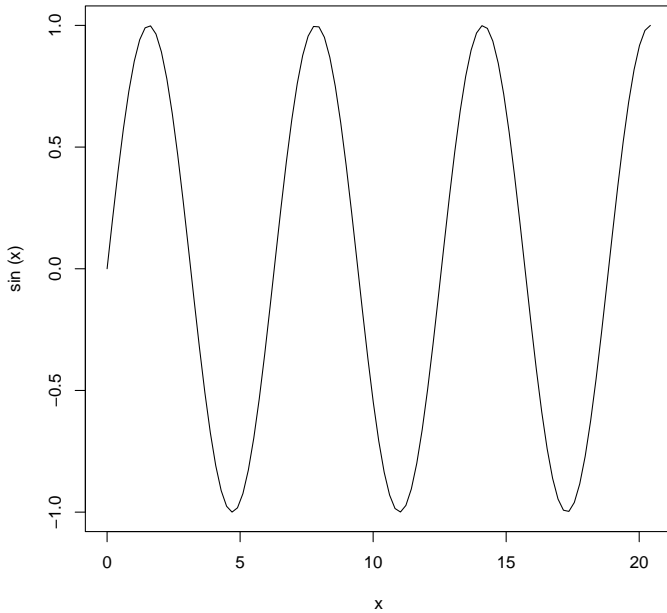Course on Image Analysis, winter term 2011/12
23/65

Sampling

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
24/65

Sampling

Sampling a function f at equidistant points $T \cdot \mathbb{Z}$ can be understood as

$$f^{\mathsf{sampled}}(x) = f(x) \cdot \Delta_T(x)$$

Fourier Transform of a Sampled Function

Let $f : \mathbb{R} \to \mathbb{C}$ be a function and
f^{sampled} be a sample of f with sampling period T.

Then its Fourier transform $\mathcal{F}(f^{\mathsf{sampled}})$ is $2\pi/T$-periodic and
aggregates the Fourier transform $\mathcal{F}(f)$ over a $2\pi/T$-periodic grid:

$$\mathcal{F}(f^{\mathsf{sampled}})(x) = \sum_{n \in \mathbb{Z}} \mathcal{F}(f)(x + n\frac{2\pi}{T})$$

If $\mathcal{F}(f)$ vanishes for $|x| > \pi/T$, then the Fourier transform $\mathcal{F}(f)$ is
replicated in a period of the Fourier transform $\mathcal{F}(f^{\mathsf{sampled}})$.

Otherwise replicas overlap and the Fourier transform becomes
corrupted. This effect is called **aliasing**.

Fourier Transform of a Sampled Function

The maximal occurring frequency

$$\omega_{\max} := \max\{|x| \,|\, x \in \mathbb{R}, \mathcal{F}(f)(x) \neq 0\}$$

of the Fourier transform is called its **bandwidth**.

The frequency

$$\omega_s := \frac{2\pi}{T}$$

of the sampling function is called its **sampling frequency**.

Then the sampling frequency must be at least twice the bandwith:

$$\omega_s > 2\omega_{\max}$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
26/65

Fourier Transform of a Sampled Function

(cf. [BB08, p. 330])

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
27/65
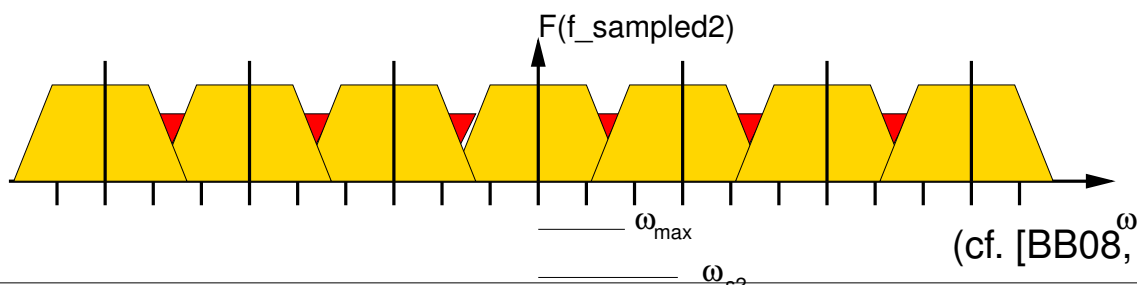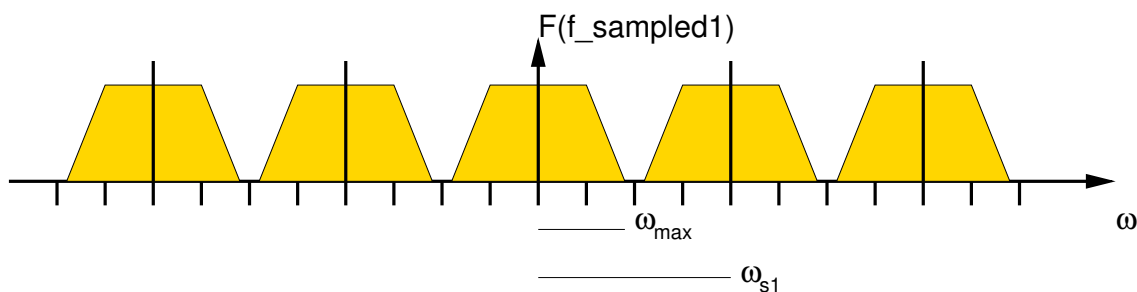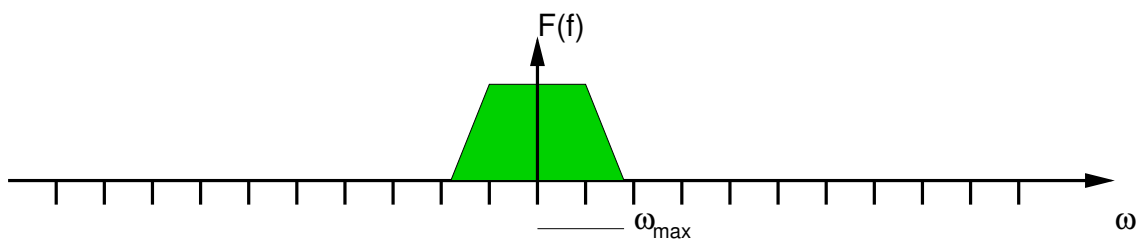
Fourier Transform of a Sampled Function

Proof.

$$
\begin{aligned}
\mathcal{F}(f^{\mathsf{sampled}})(x) =& \mathcal{F}(f \cdot \Delta_T)(x) \\
=& \mathcal{F}(f) * \mathcal{F}(\Delta_T)(x) \\
=& \mathcal{F}(f) * \frac{1}{T} \Delta_{2\pi/T}(x) \\
=& \mathcal{F}(f) * \frac{1}{T} \sum_{n \in \mathbb{Z}} \delta(x + n\frac{2\pi}{T}) \\
=& \frac{1}{T} \sum_{n \in \mathbb{Z}} \mathcal{F}(f) * \delta(x + n\frac{2\pi}{T}) \\
=& \frac{1}{T} \sum_{n \in \mathbb{Z}} \mathcal{F}(f)(x + n\frac{2\pi}{T})
\end{aligned}
$$

If $|x| < \pi/T$ and $\mathcal{F}(f)(y)$ vanishes for $|y| > \pi/T$, then

$$
\mathcal{F}(f^{\mathsf{sampled}})(x) = \frac{1}{T} \mathcal{F}(f)(x)
$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
28/65

The Fourier Transform of a Discrete Function

Let f be a discrete function:

$$
f(x) = \sum_{n \in \mathbb{Z}} y_n \, \delta(x - n), \quad y_n \in \mathbb{R}
$$

Then its Fourier transform is:

$$
\begin{aligned}
\mathcal{F}(f)(\omega) =& \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx \\
=& \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \sum_{n \in \mathbb{Z}} y_n \, \delta(x - n) \, e^{-i\omega x} dx \\
=& \frac{1}{\sqrt{2\pi}} \sum_{n \in \mathbb{Z}} \int_{-\infty}^{\infty} y_n \, e^{-i\omega x} \, \delta(x - n) dx \\
=& \frac{1}{\sqrt{2\pi}} \sum_{n \in \mathbb{Z}} y_n \, e^{-i\omega n} \\
=& \frac{1}{\sqrt{2\pi}} \sum_{n \in \mathbb{Z}} f(n) \, e^{-i\omega n}
\end{aligned}
$$

i.e., a periodic function — or equivalently: a function defined on an interval.

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
29/65

Definition

Let $f : \{0, 1, \ldots, N-1\} \to \mathbb{C}$ be a finite discrete function, then

$$F : \{0, 1, \ldots, N-1\} \to \mathbb{C}$$

$$\omega \mapsto \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x)(\cos(2\pi\frac{\omega x}{N}) - i\sin(2\pi\frac{\omega x}{N}))$$

$$= \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x) e^{-i2\pi\frac{\omega x}{N}}$$

is called **discrete Fourier transform of** f, denoted $\text{DFT}(f)$.
Then

$$f(x) = \frac{1}{\sqrt{N}} \sum_{\omega=0}^{N-1} F(\omega)(\cos(2\pi\frac{\omega x}{N}) + i\sin(2\pi\frac{\omega x}{N}))$$

$$= \frac{1}{\sqrt{N}} \sum_{\omega=0}^{N-1} F(\omega) e^{i2\pi\frac{\omega x}{N}}$$

This is called **inverse discrete Fourier transform of** f, denoted $\text{DFT}^{-1}(F)$..

Example

x	$f(x)$		ω	$F(\omega)$
0	$1 + 0i$		0	$14.2302 + 0.0000i$
1	$3 + 0i$	**DFT**	1	$-5.6745 - 2.9198i$
2	$5 + 0i$	$\longrightarrow$	2	$0.0000 + 0.0000i$
3	$7 + 0i$		3	$-0.0176 - 0.6893i$
4	$9 + 0i$		4	$0.0000 + 0.0000i$
5	$8 + 0i$		5	$0.3162 + 0.0000i$
6	$6 + 0i$		6	$0.0000 + 0.0000i$
7	$4 + 0i$	**DFT**$^{-1}$	7	$-0.0176 + 0.6893i$
8	$2 + 0i$	$\longleftarrow$	8	$0.0000 + 0.0000i$
9	$0 + 0i$		9	$-5.6745 + 2.9198i$

(cf. [BB08, p. 333])

$$\mathsf{DFT}(f)(0) = \frac{1}{\sqrt{10}} \sum_{x=0}^{9} f(x)(\cos(2\pi \frac{0x}{10}) - i\sin(2\pi \frac{0x}{10})) = \frac{1}{\sqrt{10}} \sum_{x=0}^{9} f(x) = \frac{45}{\sqrt{10}} = 14.2303$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
31/65

Example

x	$f(x)$		ω	$F(\omega)$
0	$1 + 0i$		0	$14.2302 + 0.0000i$
1	$3 + 0i$	**DFT**	1	$-5.6745 - 2.9198i$
2	$5 + 0i$	$\longrightarrow$	2	$0.0000 + 0.0000i$
3	$7 + 0i$		3	$-0.0176 - 0.6893i$
4	$9 + 0i$		4	$0.0000 + 0.0000i$
5	$8 + 0i$		5	$0.3162 + 0.0000i$
6	$6 + 0i$		6	$0.0000 + 0.0000i$
7	$4 + 0i$	**DFT**$^{-1}$	7	$-0.0176 + 0.6893i$
8	$2 + 0i$	$\longleftarrow$	8	$0.0000 + 0.0000i$
9	$0 + 0i$		9	$-5.6745 + 2.9198i$

(cf. [BB08, p. 333])

$$\mathsf{DFT}(f)(1) = \frac{1}{\sqrt{10}} \sum_{x=0}^{9} f(x)(\cos(2\pi \frac{1x}{10}) - i\sin(2\pi \frac{1x}{10}))$$

$$= \frac{1}{\sqrt{10}} \sum_{x=0}^{9} f(x)\cos(\pi\frac{x}{5}) - i\frac{1}{\sqrt{10}} \sum_{x=0}^{9} f(x)\sin(\pi\frac{x}{5}) = -5.6745 - 2.9198i$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
31/65

Discrete Fourier Transform / Algorithm (naive)

For $x \in \mathbb{C}$, denote $\Re(x)$ its **real part** and $\Im(x)$ its **imaginary part**, i.e.,

$$x = \Re(x) + i\Im(x), \quad \Re(x), \Im(x) \in \mathbb{R}$$

(for $\Re$ one often also uses Re, for $\Im$ also Im).

To compute the discrete Fourier transform, one computes $\mathsf{dft}(f)(\omega)$ for $\omega = 0, \ldots, N-1$ via:

$$
\begin{aligned}
\mathsf{DFT}(f)(\omega) =& \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x)(\cos(2\pi\frac{\omega x}{N}) - i\sin(2\pi\frac{\omega x}{N})) \\
=& \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} (\Re(f(x)) + i\Im(f(x)))(\cos(2\pi\frac{\omega x}{N}) - i\sin(2\pi\frac{\omega x}{N})) \\
=& \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \Re(f(x))\cos(2\pi\frac{\omega x}{N}) + \Im(f(x))\sin(2\pi\frac{\omega x}{N}) \\
& + i\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} -\Re(f(x))\sin(2\pi\frac{\omega x}{N}) + \Im(f(x))\cos(2\pi\frac{\omega x}{N})
\end{aligned}
$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
32/65

Discrete Fourier Transform / Algorithm (naive)

1 $\mathsf{dft\text{-}naive}(\text{sequence } f = (f(x)_0, f(x)_1)_{x=0,\ldots,N-1}) :$
2 $N := \text{length}(f)$
3 $F := (F(x)_0, F(x)_1)_{x=0,\ldots,N-1} = (0,0)_{x=0,\ldots,N-1}$
4 **for** $\omega := 0, \ldots, N-1$ **do**
5 $\quad c := (c_0, c_1) := (0, 0)$
6 $\quad$ **for** $x := 0, \ldots, N-1$ **do**
7 $\quad\quad c_0 := c_0 + f(x)_0 \cdot \cos(2\pi\omega x/N) + f(x)_1 \cdot \sin(2\pi\omega x/N)$
8 $\quad\quad c_1 := c_1 - f(x)_0 \cdot \sin(2\pi\omega x/N) + f(x)_1 \cdot \cos(2\pi\omega x/N)$
9 $\quad$ **od**
10 $\quad c_0 := c_0/\sqrt{N}$
11 $\quad c_1 := c_1/\sqrt{N}$
12 $\quad F(\omega) := c$
13 **od**
14 **return** F

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
33/65

Discrete Fourier Transform / Algorithm (naive)

When computing the values of the discrete Fourier transform for different arguments ω, the cosine and sine functions repeatedly are calle with the same arguments:

$\mathbf{dft}(f)(1)$:

$$\cos(2\pi \cdot 0/10)$$
$$\cos(2\pi \cdot 1/10)$$
$$\cos(2\pi \cdot 2/10)$$
$$\cos(2\pi \cdot 3/10)$$
$$\cos(2\pi \cdot 4/10)$$
$$\cos(2\pi \cdot 5/10)$$
$$\cos(2\pi \cdot 6/10)$$
$$\cos(2\pi \cdot 7/10)$$
$$\cos(2\pi \cdot 8/10)$$
$$\cos(2\pi \cdot 9/10)$$

$\mathbf{dft}(f)(2)$:

$$\cos(2\pi \cdot 0/10)$$
$$\cos(2\pi \cdot 2/10)$$
$$\cos(2\pi \cdot 4/10)$$
$$\cos(2\pi \cdot 6/10)$$
$$\cos(2\pi \cdot 8/10)$$
$$\cos(2\pi \cdot 10/10) = \cos(2\pi \cdot 0/10)$$
$$\cos(2\pi \cdot 12/10) = \cos(2\pi \cdot 2/10)$$
$$\cos(2\pi \cdot 14/10) = \cos(2\pi \cdot 4/10)$$
$$\cos(2\pi \cdot 16/10) = \cos(2\pi \cdot 6/10)$$
$$\cos(2\pi \cdot 18/10) = \cos(2\pi \cdot 8/10)$$

Caching the expensive sine and cosine computations accelerates the algorithm!

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
34/65

Discrete Fourier Transform / Algorithm (naive, cached)

1 dft-naive-cached(sequence $f = (f(x)_0, f(x)_1)_{x=0,\ldots,N-1})$:
2 $N := \text{length}(f)$
3 **for** $\omega := 0, \ldots, N-1$ **do**
4 $\quad C(\omega) := \cos(2\pi\omega/N)$
5 $\quad S(\omega) := \sin(2\pi\omega/N)$
6 **od**
7 $F := (F(x)_0, F(x)_1)_{x=0,\ldots,N-1} = (0,0)_{x=0,\ldots,N-1}$
8 **for** $\omega := 0, \ldots, N-1$ **do**
9 $\quad c := (c_0, c_1) := (0,0)$
10 $\quad$ **for** $x := 0, \ldots, N-1$ **do**
11 $\quad\quad c_0 := c_0 + f(x)_0 \cdot C(\omega x \bmod N) + f(x)_1 \cdot S(\omega x \bmod N)$
12 $\quad\quad c_1 := c_1 - f(x)_0 \cdot S(\omega x \bmod N) + f(x)_1 \cdot C(\omega x \bmod N)$
13 $\quad$ **od**
14 $\quad c_0 := c_0/\sqrt{N}$
15 $\quad c_1 := c_1/\sqrt{N}$
16 $\quad F(\omega) := c$
17 **od**
18 **return** F

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
35/65

Fast Fourier Transform (Gauss ca. 1805; Cooley/Tukey 1965)

The naive algorithm for DFT still has complexity $O(N^2)$.

The **Fast Fourier Transform** algorithm is based on a decomposition of the DFT for sequences f of even length N:

$$\mathsf{DFT}(f)(\omega) = \mathsf{DFT}(f^{\mathsf{even}})(\omega \bmod N/2) + e^{-i2\pi\omega/N}\mathsf{DFT}(f^{\mathsf{odd}})(\omega \bmod N/2)$$

for $\omega = 0, \ldots, N-1$
and where

$$f^{\mathsf{even}}(x) := f(2x), \quad x = 0, \ldots, N/2$$
$$f^{\mathsf{odd}}(x) := f(2x+1)$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
36/65

Fast Fourier Transform / Proof

Proof.

$$\begin{aligned}
\mathsf{DFT}(f)(\omega) &= \sum_{x=0}^{N-1} f(x)e^{-i2\pi\frac{\omega x}{N}} \\
&= \sum_{x=0 \text{ even}}^{N-1} f(x)e^{-i2\pi\frac{\omega x}{N}} + \sum_{x=0 \text{ odd}}^{N-1} f(x)e^{-i2\pi\frac{\omega x}{N}} \\
&= \sum_{x=0}^{N/2-1} f(2x)e^{-i2\pi\frac{\omega 2x}{N}} + \sum_{x=0}^{N/2-1} f(2x+1)e^{-i2\pi\frac{\omega(2x+1)}{N}} \\
&= \sum_{x=0}^{N/2-1} f^{\mathsf{even}}(x)e^{-i2\pi\frac{\omega x}{N/2}} + e^{-i2\pi\omega/N}\sum_{x=0}^{N/2-1} f^{\mathsf{odd}}(x)e^{-i2\pi\frac{\omega x}{N/2}} \\
&= \mathsf{DFT}(f^{\mathsf{even}})(\omega) + e^{-i2\pi\omega/N}\mathsf{DFT}(f^{\mathsf{odd}})(\omega)
\end{aligned}$$

or more exaclty, as $\mathsf{DFT}(f^{\mathsf{even}})(\omega)$ is only defined for $\omega < N/2$:

$$= \mathsf{DFT}(f^{\mathsf{even}})(\omega \bmod N/2) + e^{-i2\pi\omega/N}\mathsf{DFT}(f^{\mathsf{odd}})(\omega \bmod N/2)$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
37/65

Fast Fourier Transform / Real version

$$\text{DFT}(f)(\omega) = \text{DFT}(f^{\text{even}})(\omega \bmod N/2) + e^{-i2\pi\omega/N}\text{DFT}(f^{\text{odd}})(\omega \bmod N/2)$$
$$= \text{DFT}(f^{\text{even}})(\omega \bmod N/2)$$
$$+ (\cos 2\pi\omega/N - i\sin 2\pi\omega/N)\text{DFT}(f^{\text{odd}})(\omega \bmod N/2)$$

and thus

$$\Re(\text{DFT}(f)(\omega)) = \Re(\text{DFT}(f^{\text{even}})(\omega \bmod N/2))$$
$$+ \cos 2\pi\omega/N \cdot \Re(\text{DFT}(f^{\text{odd}})(\omega \bmod N/2))$$
$$+ \sin 2\pi\omega/N \cdot \Im(\text{DFT}(f^{\text{odd}})(\omega \bmod N/2))$$

$$\Im(\text{DFT}(f)(\omega)) = \Im(\text{DFT}(f^{\text{even}})(\omega \bmod N/2))$$
$$+ \cos 2\pi\omega/N \cdot \Im(\text{DFT}(f^{\text{odd}})(\omega \bmod N/2))$$
$$- \sin 2\pi\omega/N \cdot \Re(\text{DFT}(f^{\text{odd}})(\omega \bmod N/2))$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
38/65

Fast Fourier Transform / Algorithm

1 fft(sequence $f = (f(x)_0, f(x)_1)_{x=0,\dots,N-1}$) :
2 $N := \text{length}(f)$
3 **if** N is even
4      $F := (F(x)_0, F(x)_1)_{x=0,\dots,N-1} = (0,0)_{x=0,\dots,N-1}$
5      $A := \text{fft}((f(x))_{x=0,2,4,\dots,N-2})$
6      $B := \text{fft}((f(x))_{x=1,3,5,\dots,N-1})$
7   **for** $\omega := 0,\dots,N-1$ **do**
8          $a := A(\omega \bmod N/2)$
9          $b := B(\omega \bmod N/2)$
10          $F(\omega)_0 := a_0 + \cos 2\pi\omega/N \cdot b_0 + \sin 2\pi\omega/N \cdot b_1$
11          $F(\omega)_1 := a_1 + \cos 2\pi\omega/N \cdot b_1 - \sin 2\pi\omega/N \cdot b_0$
12   **od**
13   **return**
14 **else**
15      $F := \text{dft-naive-cached}(f)$
16 **fi**
17 **return** F

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
39/65

Fast Fourier Transform / Outlook

- The computation of $\cos 2\pi\omega/N$ and $\sin 2\pi\omega/N$ also can be done recursively using the addition formulas.

- In this way, FFT best is applied to sequences of length 2^n (called **radix-2 case**).

- The FFT decomposition works with any factorization $N = N_1 \cdot N_2$ in a similar way, and thus also for sequences of length other than 2^n.

- FFT has complexity $O(N \log N)$ (if N is a power of 2).

- An early experiment from 1969 reports a runtime of 13 1/2 hours for computing the DFT of a sequence of length 2048 by the naive method and 2.4 seconds using FFT.

- In practice, FFT is implemented in a linearized version avoiding explicit recursions (see [CLRS03, p. 839]).

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
40/65

Types of Fourier Transforms

type of function f	sup f	sup $\mathcal{F}(f)$	type of Fourier decomp.
general (integrable) function	$\mathbb{R}$	$\mathbb{R}$	(general) Fourier decomposition
periodic function, function on interval	interval $I \subseteq \mathbb{R}$	$\mathbb{Z}$	Fourier series
general (integrable) discrete function (sum of Diracs)	$\mathbb{Z}$	interval $I \subseteq \mathbb{R}$	discrete-time Fourier transform
periodic discrete function, finite discrete function (finite sum of Diracs)	finite $I \subseteq \mathbb{Z}$	I	discrete Fourier transform

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
41/65

1. Fourier Series Representation

2. The Fourier Transform

3. Discrete Signals

4. Discrete Fourier Transform

5. Two-dimensional Fourier Transforms

6. Applications

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
42/65

General Fourier Transform in 2D

For two-dimensional functions $f : \mathbb{R} \times \mathbb{R} \to \mathbb{C}$ Fourier Transforms, Fourier Series and Discrete Fourier Transforms can be defined analogously.

Let $f : \mathbb{R} \times \mathbb{R} \to \mathbb{C}$ be a function (that satisfies some regularity conditions). Then

$$F : \mathbb{R} \times \mathbb{R} \to \mathbb{C}$$
$$(\omega_1, \omega_2) \mapsto \frac{1}{2\pi} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x,y) \, e^{-i\omega_1 x} \, e^{-i\omega_2 y} \, dy \, dx$$

exists for each (ω_1, ω_2) and is a continuous function called **Fourier Transform of** f (aka **Fourier spectrum of** f).

One can show that (if F also satisfies some regularity conditions):

$$f(x,y) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} F(\omega_1, \omega_2) e^{i\omega_1 x} e^{i\omega_2 y} d\omega_1 d\omega_2$$

This is called **Inverse Fourier Transform**.

We will write $\mathcal{F}(f) := F$ for the Fourier transform of a function f and $\mathcal{F}^{-1}(F)$ for the inverse Fourier transform of a function F.

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
42/65

Bases in 2D

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
43/65

Fourier Series in 2D

If $f : \mathbb{R} \times \mathbb{R} \to \mathbb{C}$ is (T_1, T_2)-periodic, i.e.,

$$f(x, y) = f(x + T_1, y + T_2) \quad \forall x, y \in \mathbb{R}$$

then f can already be reconstructed from $\mathbb{Z}$-many Fourier coefficients:

$$f(x, y) = \frac{1}{2\pi} \sum_{n \in \mathbb{Z}} \sum_{m \in \mathbb{Z}} F(n, m) \, e^{i\frac{2\pi}{T_1}nx} \, e^{i\frac{2\pi}{T_2}my}$$

with

$$F(n, m) := \frac{1}{2\pi} \int_{-T_1/2}^{+T_1/2} \int_{-T_2/2}^{+T_2/2} f(x, y) \, e^{-in\omega_1 x} \, e^{-im\omega_2 y} \, dy \, dx$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
44/65

Discrete Fourier Transform in 2D

If f is discrete, i.e.,

$$f(x,y) = \sum_{n \in \mathbb{Z}} \sum_{m \in \mathbb{Z}} y_{n,m} \delta(x - T_1 n, y - T_2 m), \quad y_{n,m} \in \mathbb{R}$$

then its Fourier transform is periodic:

$$f(x,y) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} F(\omega_1, \omega_2) \, e^{i\omega_1 x} \, e^{i\omega_2 y} \, d\omega_1 \, d\omega_2$$

with

$$F(\omega_1, \omega_2) := \frac{1}{2\pi} \sum_{n \in \mathbb{Z}} \sum_{m \in \mathbb{Z}} f(n,m) \, e^{-i\omega_1 n} \, e^{-i\omega_2 m}$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
45/65

Discrete Fourier Transform in 2D

And finally, if f is discrete and finite, i.e.,

$$f(x,y) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} y_{n,m} \, \delta(x - n, y - m), \quad y_{n,m} \in \mathbb{R}$$

then its Fourier transform is periodic and made from finitely many components:

$$f(x,y) = \frac{1}{\sqrt{NM}} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} F(\omega_1, \omega_2) \, e^{i\omega_1 x} \, e^{i\omega_2 y}$$

with

$$F(\omega_1, \omega_2) := \frac{1}{\sqrt{NM}} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f(n,m) \, e^{-i\omega_1 n} \, e^{-i\omega_2 m}$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
46/65

Discrete Fourier Transform in 2D / Algorithm (naive, cached)

1   dft-2d-naive-cached(array $f = (f(x)_0, f(x)_1)_{x=0,\dots,N-1,y=0,\dots,M-1})$:

2   **for** $\omega := 0, \dots, N-1$ **do**

3     $C_1(\omega) := \cos(2\pi\omega/N)$

4     $S_1(\omega) := \sin(2\pi\omega/N)$

5   **od**

6   **for** $\omega := 0, \dots, M-1$ **do**

7     $C_2(\omega) := \cos(2\pi\omega/M)$

8     $S_2(\omega) := \sin(2\pi\omega/M)$

9   **od**

10   $F := (F(x)_0, F(x)_1)_{x=0,\dots,N-1,y=0,\dots,M-1} = (0,0)_{x=0,\dots,N-1,y=0,\dots,M-1}$

11   **for** $\omega_1 := 0, \dots, N-1$ **do**

12   **for** $\omega_2 := 0, \dots, M-1$ **do**

13       $c := (c_0, c_1) := (0,0)$

14   **for** $x := 0, \dots, N-1$ **do**

15   **for** $y := 0, \dots, M-1$ **do**

16          $C := C_1(\omega_1 x \bmod N) \cdot C_2(\omega_2 y \bmod M) - S_1(\omega_1 x \bmod N) \cdot S_2(\omega_2 y \bmod M)$

17          $S := S_1(\omega_1 x \bmod N) \cdot C_2(\omega_2 y \bmod M) + C_1(\omega_1 x \bmod N) \cdot S_2(\omega_2 y \bmod M)$

18          $c_0 := c_0 + f(x)_0 \cdot C - f(x)_1 \cdot S$

19          $c_1 := c_1 - f(x)_0 \cdot S + f(x)_1 \cdot C$

20   **od**

21   **od**

22       $c_0 := c_0/\sqrt{NM}$

23       $c_1 := c_1/\sqrt{NM}$

24       $F(\omega) := c$

25   **od**

26   **od**

27   **return** F

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
47/65

Discrete Fourier Transform in 2D / Separability

$$
\begin{aligned}
F(\omega_1, \omega_2) :=& \frac{1}{\sqrt{NM}} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f(n,m)\, e^{-i\omega_1 n}\, e^{-i\omega_2 m} \\
=& \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \left(\frac{1}{\sqrt{M}} \sum_{m=0}^{M-1} f(n,m)\, e^{-i\omega_2 m} \right) e^{-i\omega_1 n} \\
=& \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \mathsf{DFT}((f(n,m))_{m=1,\dots,M})(\omega_2)\, e^{-i\omega_1 n}
\end{aligned}
$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
48/65

Discrete Fourier Transform in 2D / FFT

1   fft-2d(array $f = (f(x)_0, f(x)_1)_{x=0,\dots,N-1,y=0,\dots,M-1})$:
2    $G := (G(x)_0, G(x)_1)_{x=0,\dots,N-1,y=0,\dots,M-1} = (0,0)_{x=0,\dots,N-1,y=0,\dots,M-1}$
3   **for** $\omega_1 := 0, \dots, N-1$ **do**
4      $G(\omega_1, .) := \text{fft}(f(\omega_1, y)_{y=0,\dots,M-1})$
5   **od**
6    $F := (F(x)_0, F(x)_1)_{x=0,\dots,N-1,y=0,\dots,M-1} = (0,0)_{x=0,\dots,N-1,y=0,\dots,M-1}$
7   **for** $\omega_2 := 0, \dots, M-1$ **do**
8      $F(., \omega_2) := \text{fft}(G(x, \omega_2)_{x=0,\dots,N-1})$
9   **od**
10   **return** F

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
49/65

Power Spectrum

The fourier spectrum of

– a discrete $N \times M$ gray-scale image f,
  i.e., with one channel,

is

– a discrete $N \times M$ image F with complex intensity values,
  i.e., two channels.

For visualization one usually shows the **power spectrum** defined
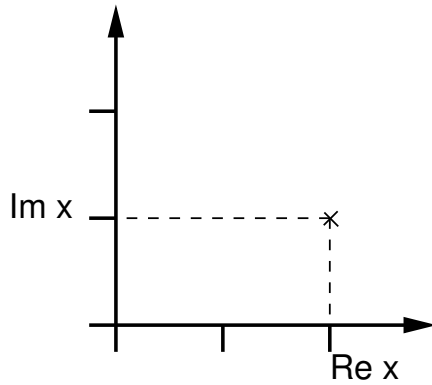as:
$$\mathcal{F}^{\text{power}}(f)(x) := \sqrt{(\Re\mathcal{F}(f)(x))^2 + (\Im\mathcal{F}(f)(x))^2}$$

The power spectrum measures the absolute value of the complex
amplitude.
The complementary information θ called **phase** is not shown.

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
50/65

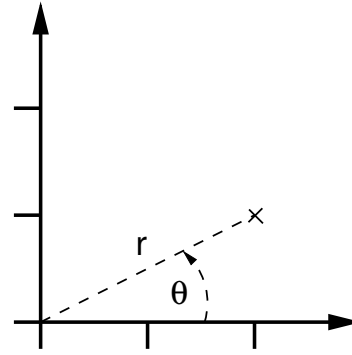## What does Power Spectrum mean? — Complex Coordinates

Cartesian coordinates:

Polar coordinates:

$$x = (\Re x, \Im x) \in \mathbb{R} \times \mathbb{R}$$

$$\Re x = r \cos \theta$$
$$\Im x = r \sin \theta$$

$$x = (r, \theta) \in \mathbb{R}_0^+ \times [0, 2\pi)$$

$$r = \sqrt{(\Re x)^2 + (\Im x)^2}$$

$$\theta = \begin{cases} \arctan(\Re x / \Im x), & \text{if } x > 0, y > 0 \\ \arctan(\Re x / \Im x) + \pi, & \text{if } y < 0 \\ \arctan(\Re x / \Im x) + 2\pi, & \text{if } x > 0, y < 0 \end{cases}$$

$$x = \Re x + i \Im x = r \cos \theta + i r \sin \theta = r e^{i\theta}$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
51/65

Example

image f:

power spectrum $\mathcal{F}^{\text{power}}(f)$:

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
52/65

Displaying Fourier Power Spectra

For displaying spectra, some further conventions are used:

- As the scale of many power spectra is dominated by a few large values, one usually plots

$$\log \mathcal{F}^{\mathsf{power}}(f)(x) \quad \text{or} \quad (\mathcal{F}^{\mathsf{power}}(f)(x))^{\frac{1}{2}}$$

  instead of the raw power spectrum values $\mathcal{F}^{\mathsf{power}}(f)(x)$.

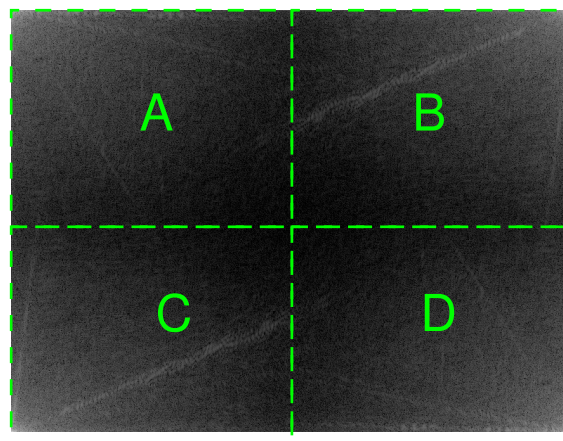- Usually, the **centered spectrum** is shown, i.e., the intensities for

$$x \in \{-N/2, -N/2 + 1, \ldots, -1, 0, 1, \ldots, N/2 - 1, N/2\}$$
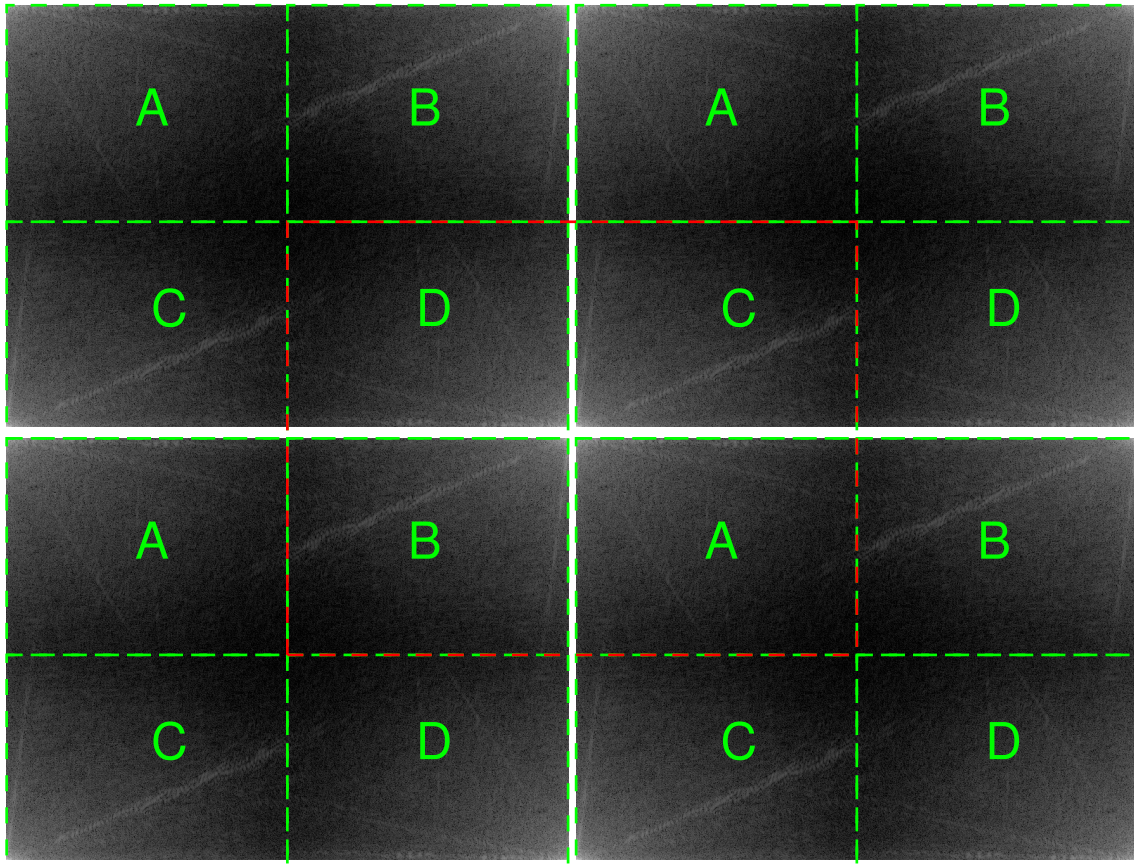
  and

$$y \in \{-M/2, -M/2 + 1, \ldots, -1, 0, 1, \ldots, M/2 - 1, M/2\}$$

  instead of the intensities for $0, 1, \ldots, N-1$ and $0, 1, \ldots, M-1$.

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
53/65
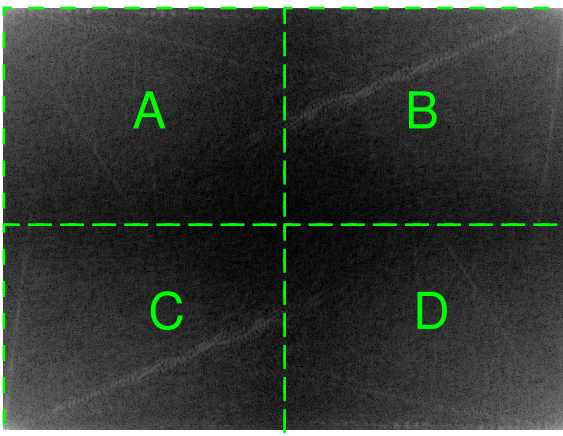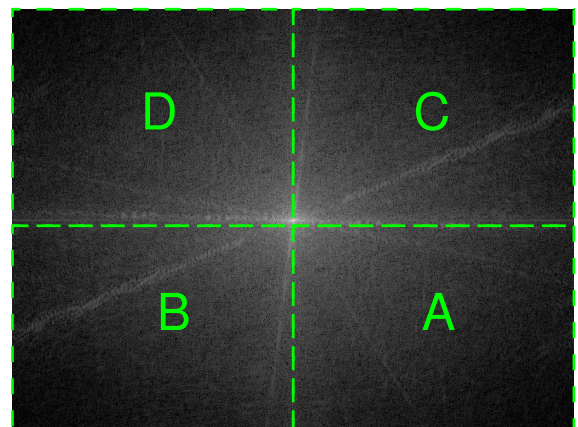
Centered Spectrum

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
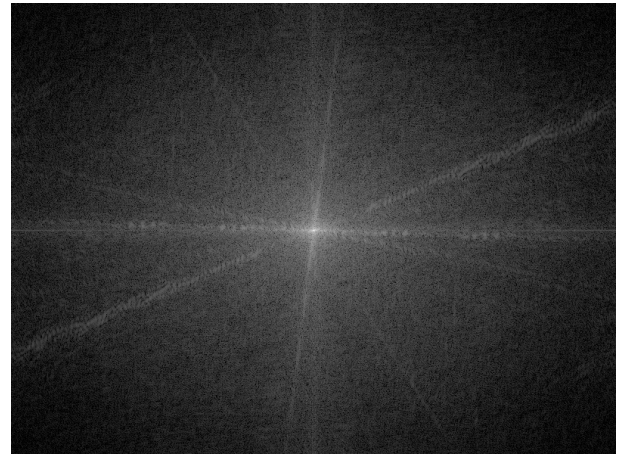Course on Image Analysis, winter term 2011/12
54/65

Centered Spectrum

Centered Spectrum

original spectrum:

centered spectrum:

Example

image f:

power spectrum $\mathcal{F}^{\text{power}}(f)$:

Symmetry of Fouier Power Spectra for Real Images

Usually images are real, i.e., $f(x) \in \mathbb{R}$ (not $\mathbb{C}$).

For real functions, we know that

$$\mathcal{F}(f)(-x) = \mathcal{F}^*(f)(x)$$

is hermitian (with $x^* := \Re x - i \Im x$).
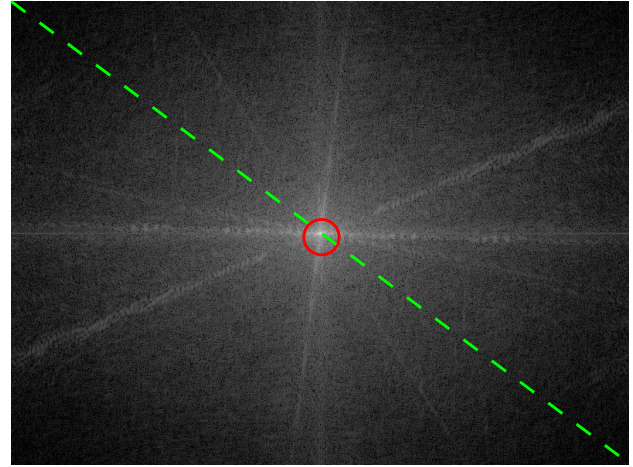
As $x \in \mathbb{C}$ has the same radius as x^*:

$$r(x) = \sqrt{(\Re x)^2 + (\Im x)^2} \overset{?}{=} r(x^*) = r(\Re x - i\Im x) = \sqrt{(\Re x)^2 + (-\Im x)^2}$$

for real functions $\mathcal{F}(f)$ and $\mathcal{F}^*(f)$ have the same radius and thus

$$\mathcal{F}^{\text{power}}(f)(-x) = \mathcal{F}^{\text{power}}(f)(x)$$

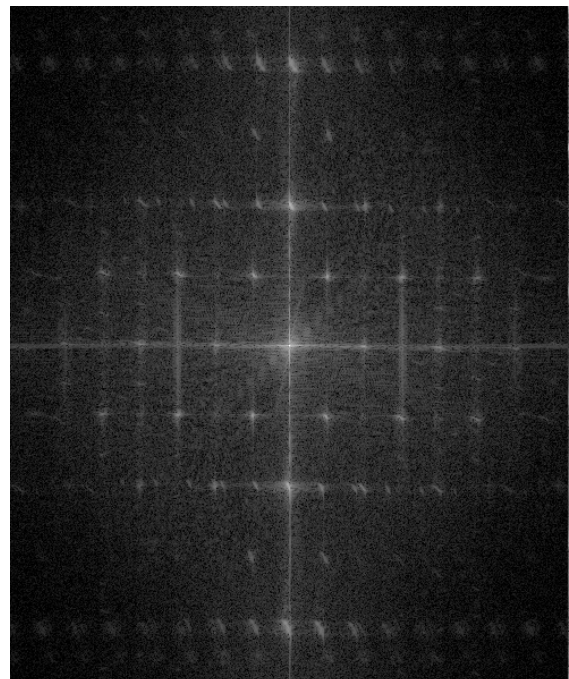i.e., the power spectrum is symmetric around the origin.

Symmetry of Fouier Power Spectra for Real Images

image f:

power spectrum $\mathcal{F}^{\text{power}}(f)$:

Spectra of real images are symmetric around the origin (red circle).

So storing just half of the power spectrum is sufficient
(e.g., above green line — any line through the origin will do).

Another Example

image f:

power spectrum $\mathcal{F}^{\text{power}}(f)$:

1. Fourier Series Representation

2. The Fourier Transform

3. Discrete Signals

4. Discrete Fourier Transform

5. Two-dimensional Fourier Transforms
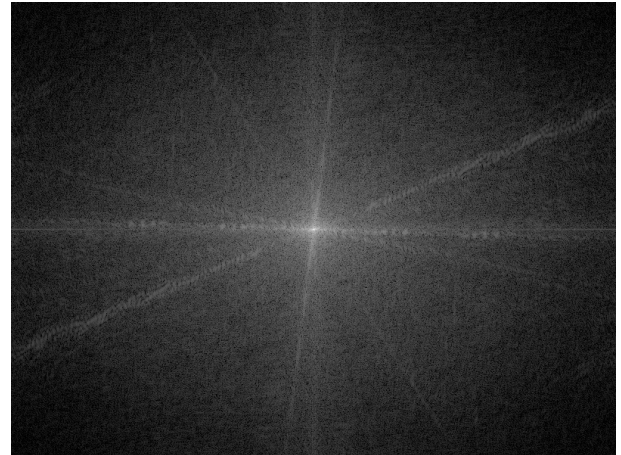
6. Applications

Low Pass Filters

High frequencies are responsible for sharp edges,
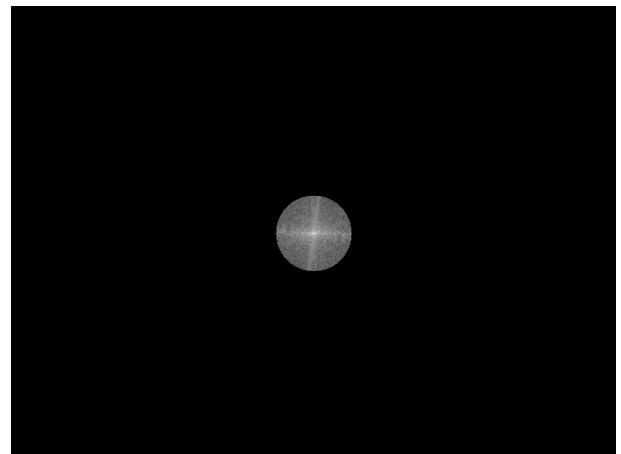low frequencies for constant and slowly changing areas.

Low pass filters

– retain only low frequencies $\omega \leq \omega_{\max}$,
– i.e., filter out high frequencies.
– and thus smooth / blur an image.

Low Pass Filters / Example

image f:

power spectrum $\mathcal{F}^{\text{power}}(f)$:

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
58/65

Low Pass Filters / Example

image f:

power spectrum $\mathcal{F}^{\text{power}}(f)$:

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
58/65

Low Pass Filters / Example

image f:

power spectrum $\mathcal{F}^{\mathsf{power}}(f)$:
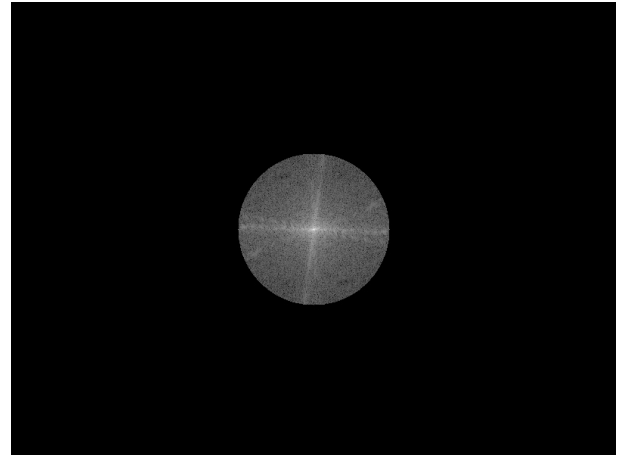
High Pass Filters

High frequencies are responsible for sharp edges,
low frequencies for constant and slowly changing areas.

High pass filters

– retain only high frequencies $\omega \geq \omega_{\mathsf{min}}$,

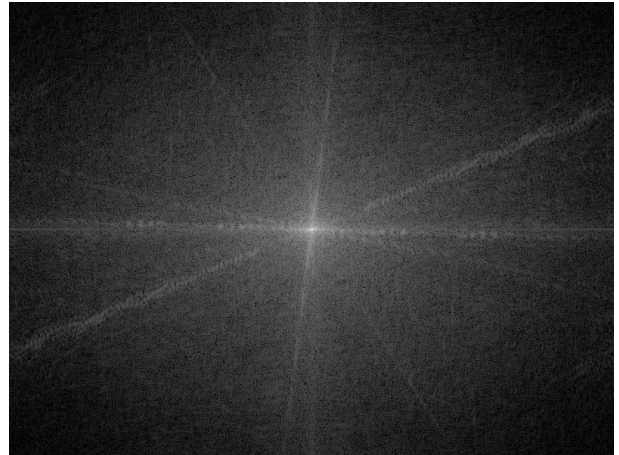– i.e., filter out low frequencies.

– and thus sharpen an image and detect edges.

High Pass Filters / Example

image f:

power spectrum $\mathcal{F}^{\text{power}}(f)$:

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
60/65

High Pass Filters / Example

image f:

power spectrum $\mathcal{F}^{\text{power}}(f)$:

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
60/65

High Pass Filters / Example

image f:

power spectrum $\mathcal{F}^{\text{power}}(f)$:

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
60/65

Band Pass Filters

High frequencies are responsible for sharp edges,
low frequencies for constant and slowly changing areas.

Band pass filters

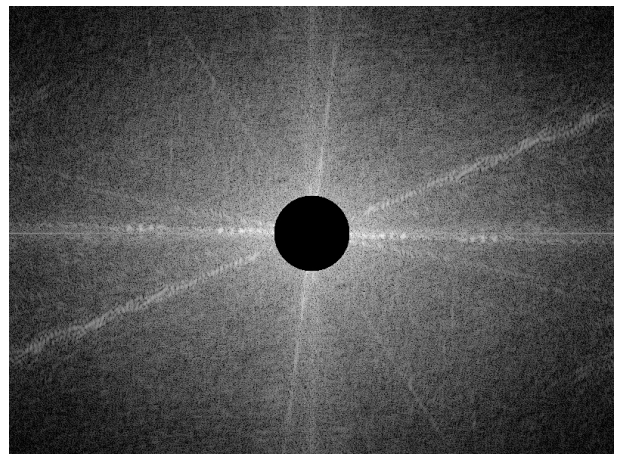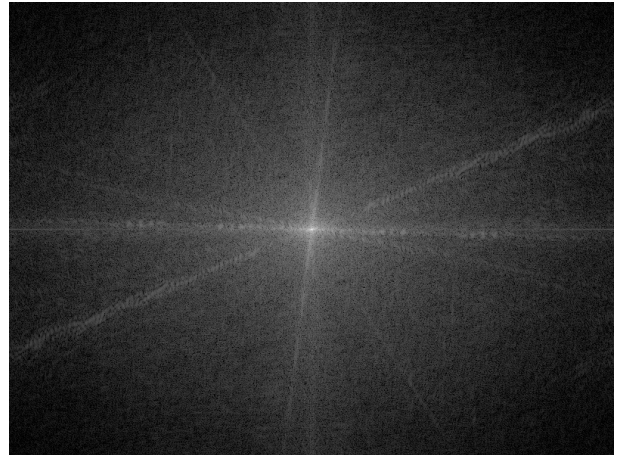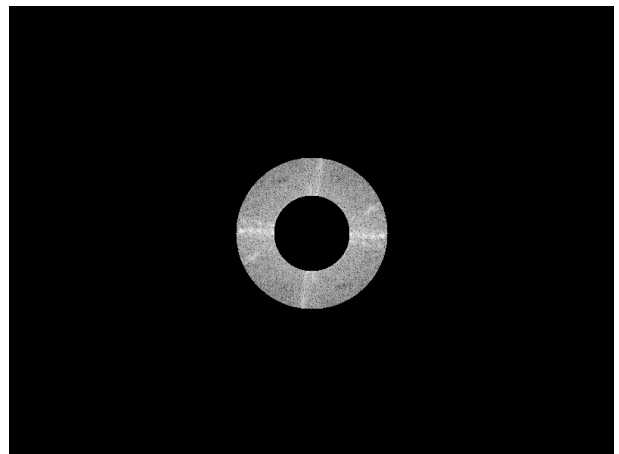– retain only frequencies $\omega \in [\omega_{\text{min}}, \omega_{\text{max}}]$ in a given interval
(the **frequency band**),

– i.e., filter out low and high frequencies.

– and thus detect edges.

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
61/65

Band Pass Filters / Example

image f:

power spectrum $\mathcal{F}^{\text{power}}(f)$:

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
62/65

Band Pass Filters / Example

image f:

power spectrum $\mathcal{F}^{\text{power}}(f)$:

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
62/65

Reducing Periodic Noise

image f:

power spectrum $\mathcal{F}^{\text{power}}(f)$:

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
63/65

Reducing Periodic Noise

image f:

power spectrum $\mathcal{F}^{\text{power}}(f)$:

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
63/65

Reducing Periodic Noise

image f:

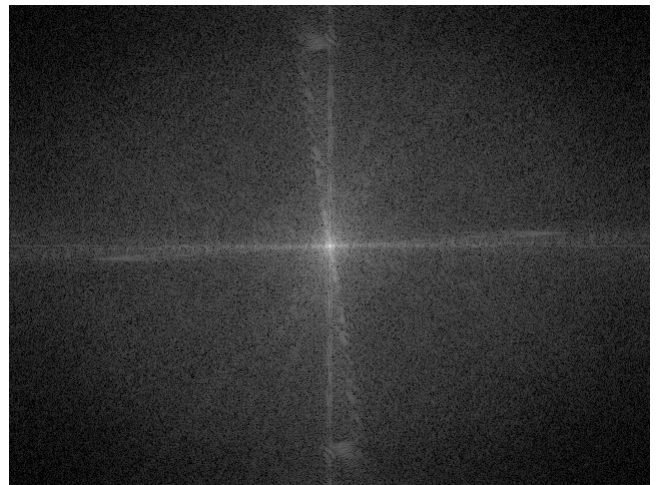power spectrum $\mathcal{F}^{\text{power}}(f)$:

Periodic noise can be reduced by filtering out the frequencies belonging to the periodic noise pattern.

This also can be understood as a simple method for **inpainting**.

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
63/65

Reducing Salt and Pepper Noise

image f:

power spectrum $\mathcal{F}^{\text{power}}(f)$:

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
64/65

Reducing Salt and Pepper Noise

image f:

power spectrum $\mathcal{F}^{\text{power}}(f)$:

Reducing non-periodic noise patterns via frequency filters is difficult.

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
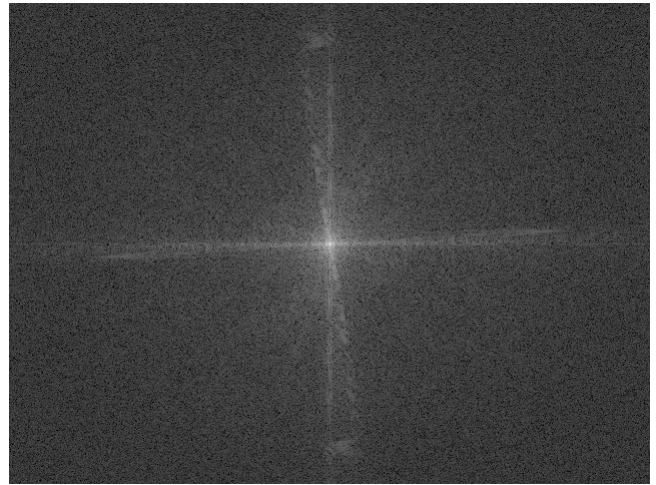Course on Image Analysis, winter term 2011/12
64/65

Deconvolution via Fourier Transform

Assume, an image f has been corrupted by a convolution with a kernel k (e.g., blurred):

$$g = k * f$$

If the kernel k is known, one can "undo" the convolution using the Fourier transform:

$$\mathcal{F}(g) = \mathcal{F}(k * f) = \mathcal{F}(k) \cdot \mathcal{F}(f)$$
$$\mathcal{F}(f) = \frac{\mathcal{F}(g)}{\mathcal{F}(k)}$$
$$f = \mathcal{F}^{-1}\mathcal{F}(f) = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(g)}{\mathcal{F}(k)}\right)$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim
Course on Image Analysis, winter term 2011/12
65/65

References

[BB08]   Wilhelm Burger and Mark J. Burge. *Digital Image Processing, An Algorithmic Intro-
           duction Using Java*. Springer, 2008.

[CLRS03] Thomas H. Corman, Charles E. Leiserson, Ronald R. Rivest, and Clifford Stein. *An
           Introduction to Algorithms*. The MIT Press, 2003.

[Kö90]   Konrad Königsberger. *Analysis 1*. Springer, 1990.