Fast Fourier Transform

Niklas J. Holzwarth [a,b]

[a] Division of Computer Assisted Medical Interventions (CAMI), German Cancer Research Center (DKFZ)
[b] Faculty of Physics and Astronomy, Heidelberg University, Germany

Gliederung

- Allgemeine Grundlagen der Fourier Analyse
- Beispiel aus der Bildverarbeitung
- FFTW (Fastest Fourier Transform in the West)
  - Cooley-Tukey Algorithmus
- Frequenzanalyse mit Root (TVirtualFFT)

Allgemeine Grundlagen

- Kontinuierliche FT

$$f(t) = \int_{-\infty}^{\infty} F(\omega) e^{2\pi\imath\omega t} d\omega\,; \qquad F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-2\pi\imath\omega t} dt$$

- Technische Umsetzung: diskrete Fourier Transformation (FFT)
- Faltung

$$(m \otimes g)(x) = \int_{-\infty}^{\infty} m(t) g(x-t) dt$$

- Faltungstheorem: F[f⊗g] = F[f] * F[g]

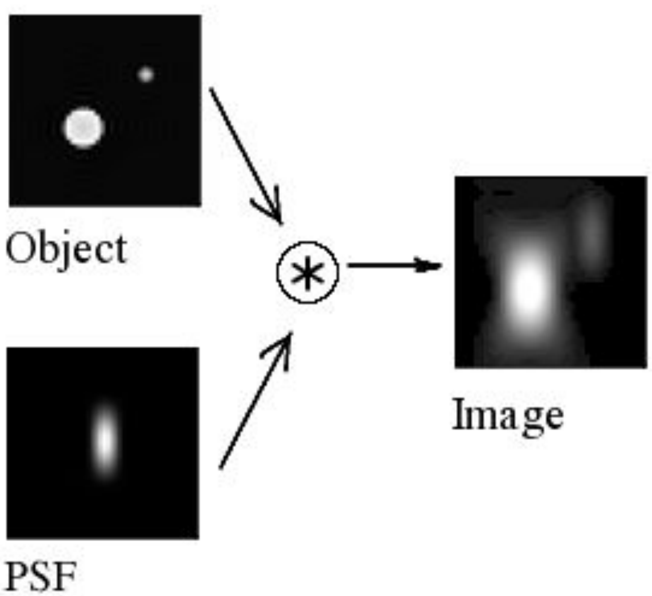
Beweis des Faltungstheorems

$$
\begin{aligned}
\mathcal{F}[f \otimes g](s) &= \mathcal{F}\left[\int_{-\infty}^{\infty} f(y) \cdot g(x-y)\,dy\right](s) \\
&= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(y) \cdot g(x-y)\,dy\right] e^{-2\pi i \cdot s \cdot x} dx \\
&= \int_{-\infty}^{\infty} f(y) \left[\int_{-\infty}^{\infty} g(x-y)\, e^{-2\pi i \cdot s \cdot x}\, dx\right] dy \\
&= \int_{-\infty}^{\infty} f(y) \left[\int_{-\infty}^{\infty} g(z)\, e^{-2\pi i \cdot s \cdot z} e^{-2\pi i \cdot s \cdot y}\, dz\right] dy \\
&= \int_{-\infty}^{\infty} f(y) \left[\int_{-\infty}^{\infty} g(z)\, e^{-2\pi i \cdot s \cdot z}\, dz\right] e^{-2\pi i \cdot s \cdot y}\, dy \\
&= \mathcal{F}[f](s) \cdot \mathcal{F}[g](s)
\end{aligned}
$$

http://wwwdh.informatik.uni-erlangen.de/IMMD8/Lectures/THINF3/Folien0405/fft-rev4.pdf

Beispiel aus der Bildverarbeitung*

- Inverses Problem: $g = PSF \otimes f$
  - g: Messwert
  - PSF: Punktbildverwaschungsfunktion (*point spread function*)
  - f: gesuchter Parameter
  - $\otimes$: Faltung



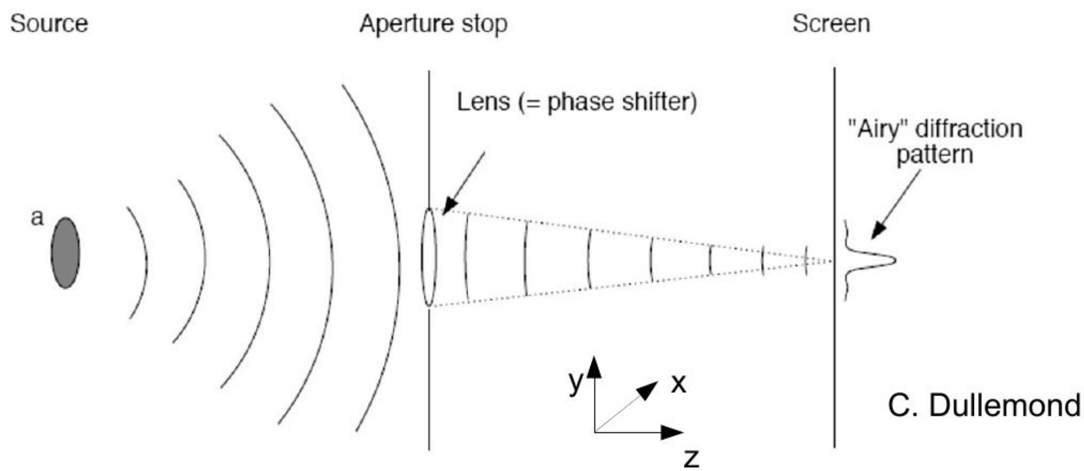


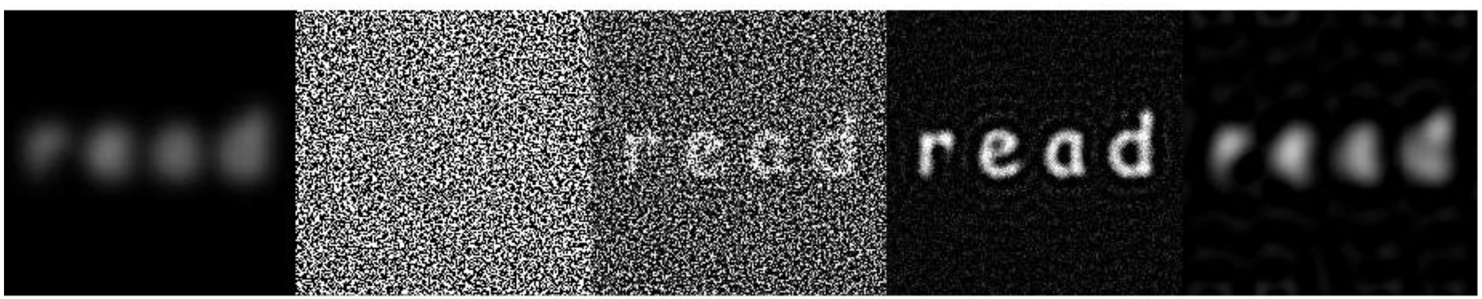
Bild links: Wiki

*Vorlesung: Inverse Probleme (Hesser)

Beispiel aus der Bildverarbeitung*

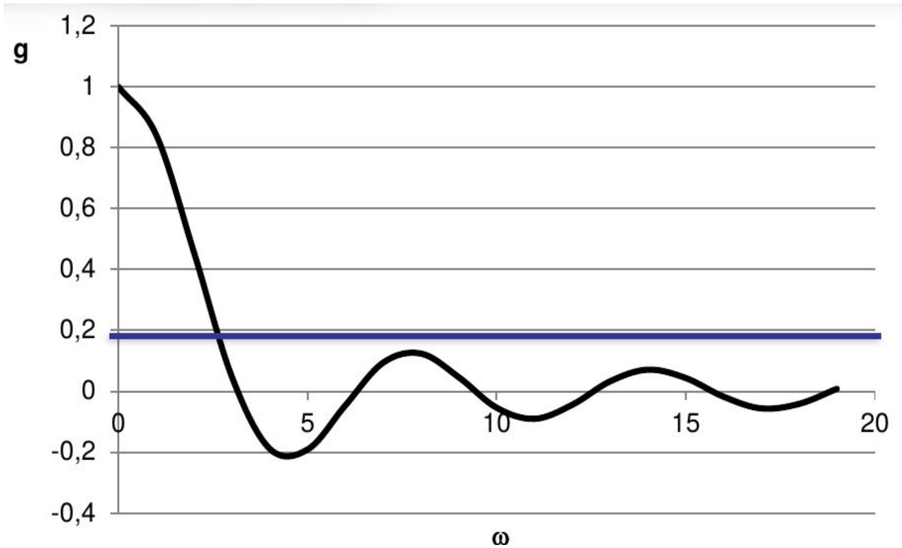
- Inverses Problem: g = PSF ⊗ f
  - g: Messwert
  - PSF: Punktbildverwaschungsfunktion (*point spread function*)
  - f: gesuchter Parameter
  - ⊗: Faltung
- Fourier Raum (⊗⟶ *): G = OTF · F
  - OFT: Optische Übertragungsfunktion
-

$$\frac{G(\omega)}{OTF(\omega)+s^2} = \hat{F}(\omega)$$

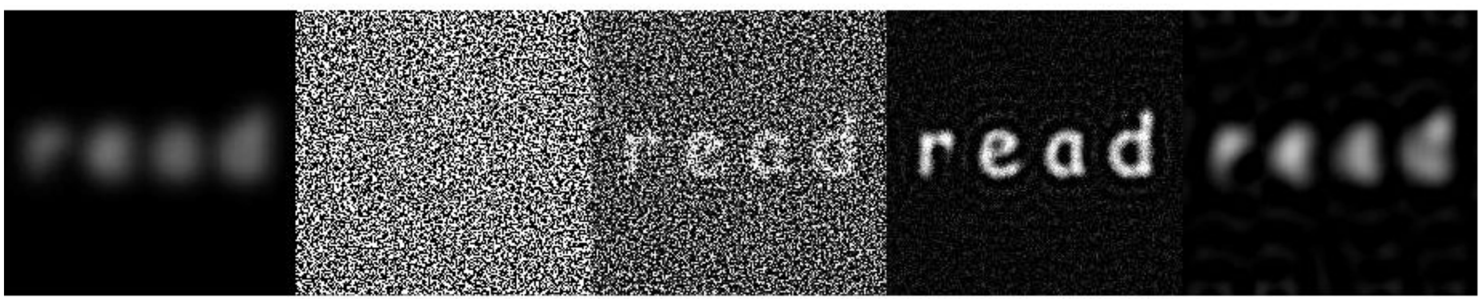


Original 0.0000001 0.000001 0.00001 0.1 [6]

Beispiel aus der Bildverarbeitung*



$$\frac{G(\omega)}{OTF(\omega)+s^2} = \hat{F}(\omega)$$



Original | 0.0000001 | 0.000001 | 0.00001 | 0.1 6

*Vorlesung: Inverse Probleme (Hesser)

FFTW

- freie C Bibliothek zur schnellen Berechnung der DFT
- “fastest Fourier Transform in the West”
- Komplexität: $O(N \log(N))$ vs. $O(N^2)$ (DFT)
- FFTW verwendet nicht EINEN einzigen Algorithmus
  - Schritt 0: “planner” (ermittelt optimale Datenstruktur für FFT)
  - Schritt 1: FFT

http://www.fftw.org/
http://www.fftw.org/fftw3.pdf

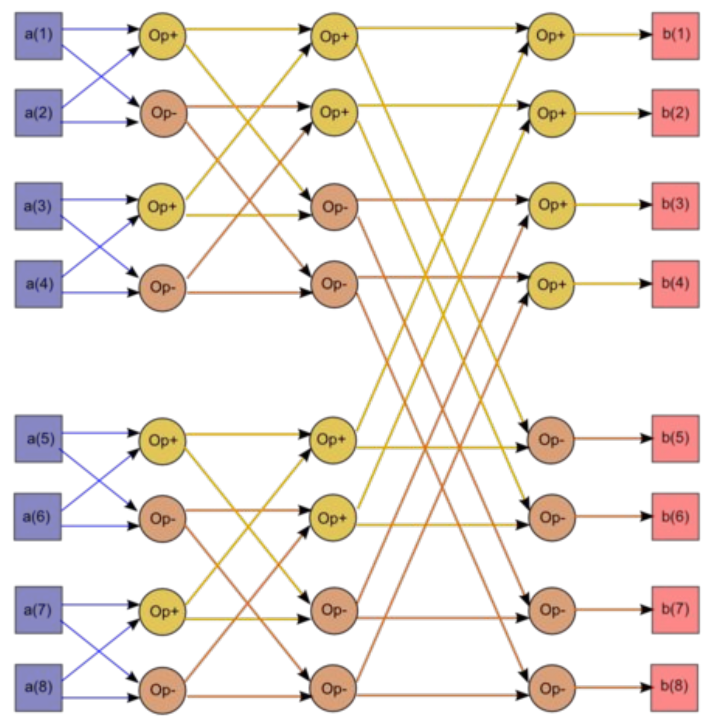
Beispiel FFT: Cooley-Tukey Algorithmus

procedure FFT(A)

Input: An array of complex values which has a size of 2^m for $m \geq 0$.

Output: An array of complex values which is the DFT of the input

N := A.length
if $N = 1$ **then return** A
else
$W_N := e^{2\pi i/N}$
W := 1
$A_{even} := (A_0, A_2, ..., A_{N-2})$
$A_{odd} := (A_1, A_3, ..., A_{N-1})$
Y_{even} := FFT(A_{even})
Y_{odd} := FFT(A_{odd})
for j:=0 to $N/2 - 1$ **do**
$Y[j] = Y_{even}[j]$ + W * $Y_{odd}[j]$
$Y[j + N/2] = Y_{even}[j]$ - W * $Y_{odd}[j]$
W := W * W_N
return Y



http://people.scs.carleton.ca/~maheshwa/courses/5703COMP/16Fall/FFT_Report.pdf

Niklas J. Holzwarth

https://algowiki-project.org/en/Cooley%E2%80%93Tukey_Fast_Fourier_Transform,_radix-2_case

Frequenzanalyse mit Root (TVirtualFFT)

TVirtualFFT

◆ FFT()

```
TVirtualFFT * TVirtualFFT::FFT ( Int_t      ndim,
                                 Int_t *    n,
                                 Option_t * option
                               )
```

static

Returns a pointer to the FFT of requested size and type.

Parameters

- [in] **ndim** number of transform dimensions
- [in] **n** sizes of each dimension (an array at least ndim long)
- [in] **option** consists of 3 parts - flag option and an option to create a new TVirtualFFT
  1. transform type option: Available transform types are: C2CForward, C2CBackward, C2R, R2C, R2HC, HC2R, DHT see class description for details
  2. flag option: choosing how much time should be spent in planning the transform: Possible options:
     - "ES" (from "estimate") - no time in preparing the transform, but probably sub-optimal performance
     - "M" (from "measure") - some time spend in finding the optimal way to do the transform
     - "P" (from "patient") - more time spend in finding the optimal way to do the transform
     - "EX" (from "exhaustive") - the most optimal way is found This option should be chosen depending on how many transforms of the same size and type are going to be done. Planning is only done once, for the first transform of this size and type.
  3. option allowing to choose between the global fgFFT and a new TVirtualFFT object "" - default, changes and returns the global fgFFT variable "K" (from "keep")- without touching the global fgFFT, creates and returns a new TVirtualFFT*. User is then responsible for deleting it.

Examples of valid options: "R2C ES K", "C2CF M", "DHT P K", etc.

Definition at line 132 of file TVirtualFFT.cxx.

https://root.cern.ch/doc/master/classTVirtualFFT.html

Frequenzanalyse mit Root (TVirtualFFT)

- #include "TVirtualFFT.h"
- Frequenzfunktion anlegen und diskretisieren (Histogramm)
- mag_hist->FFT(mag_hist, "MAG"); // “MAG”, “PH”
- TVirtualFFT *fft = TVirtualFFT::GetCurrentTransform();
- fft->GetPointsComplex(re_full,im_full);
- TVirtualFFT *fft_back = TVirtualFFT::FFT(1, &n, "C2R M K");
- fft_back->SetPointsComplex(re_full,im_full);
- fft_back->SetPoint(freq, re_filter, im_filter);
- fft_back->Transform(); → Histogramm oder Graph

https://root.cern.ch/doc/master/classTVirtualFFT.html