LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Prof. Dr. Andreas Butz | Prof. Dr. Ing. Axel Hoppe

Dipl.-Medieninf. Dominikus Baur
Dipl.-Medieninf. Sebastian Boring

Übung: Computergrafik 1

Fouriertransformation

Organisatorisches

- Neue Abgabefrist für Blatt 8: KOMMENDER Freitag, 3.7.

- Kommende Übungsblätter:
  - Blatt 9 (Fouriertransformationen) seit 26.6., Abgabe zwei Wochen später 10.7.
  - Blatt 10 (Region Growing), 3.7., Abgabe zwei Wochen später 17.7.
  - Blatt 11 (Klausurvorbereitung), 17.7. keine Abgabe!
- Kommende Übungen:
  - Region Growing, 3.7. + 6.7.
  - Klausurvorbereitung 24.7. + 27.7.

- Klausurtermin: 30.07.09, 16:30 - 18:30 Uhr

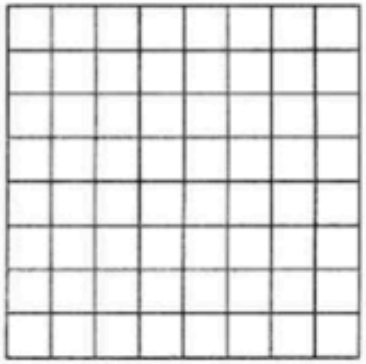
Bilder im Frequenzraum

Bilder im Frequenzraum

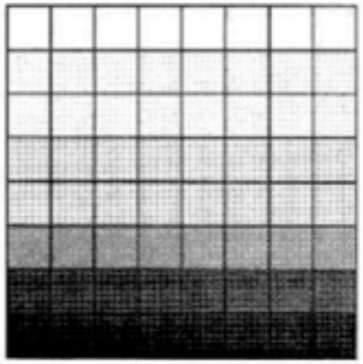
- Normalerweise arbeiten wir mit Bildern die als Helligkeitswerte (im Graustufenfall) in zwei Dimensionen definiert sind (Ortsraum)
- Bilder lassen sich allerdings auch in den Frequenzraum konvertieren, was manche Filter wie Weich- oder Scharfzeichner deutlich weniger rechenaufwändig macht
- Bisheriger Weg: Konvolutionen
  - Komplexität: O(N^4)
- Im Frequenzraum:
  - Komplexität: O(N^2) (+ Fouriertransformation)

Ortsfrequenz

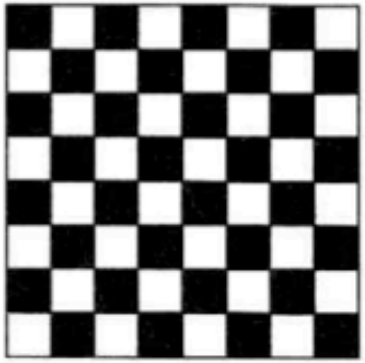
- Ortsfrequenz (oder: räumliche Frequenz, *spatial frequency*)
  - Häufigkeit der Wiederholung einer im Bild erkennbaren Eigenschaft über die räumliche Ausdehnung
  - Maßeinheit: 1/Längeneinheit
  - z.B. Dichte von Linien auf Papier: Anzahl Striche pro cm
- Meist: Anzahl von Helligkeitsschwankungen pro Längeneinheit
- 2-dimensionale Frequenz (horizontal und vertikal)



Ortsfrequenz 0

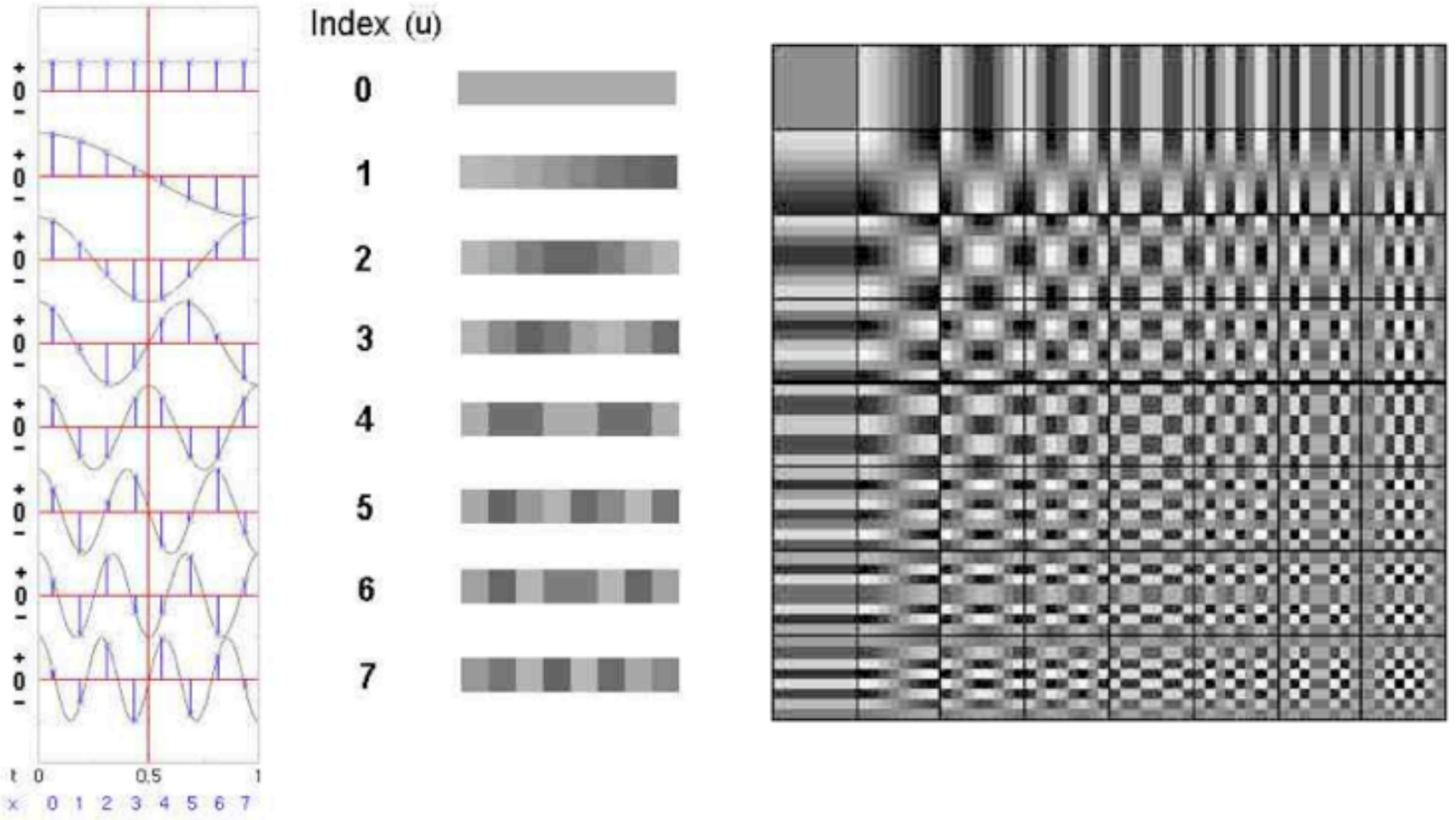


Ortsfrequenz
0 horizontal,
niedrig vertikal



Ortsfrequenz
hoch
horizontal und vertikal

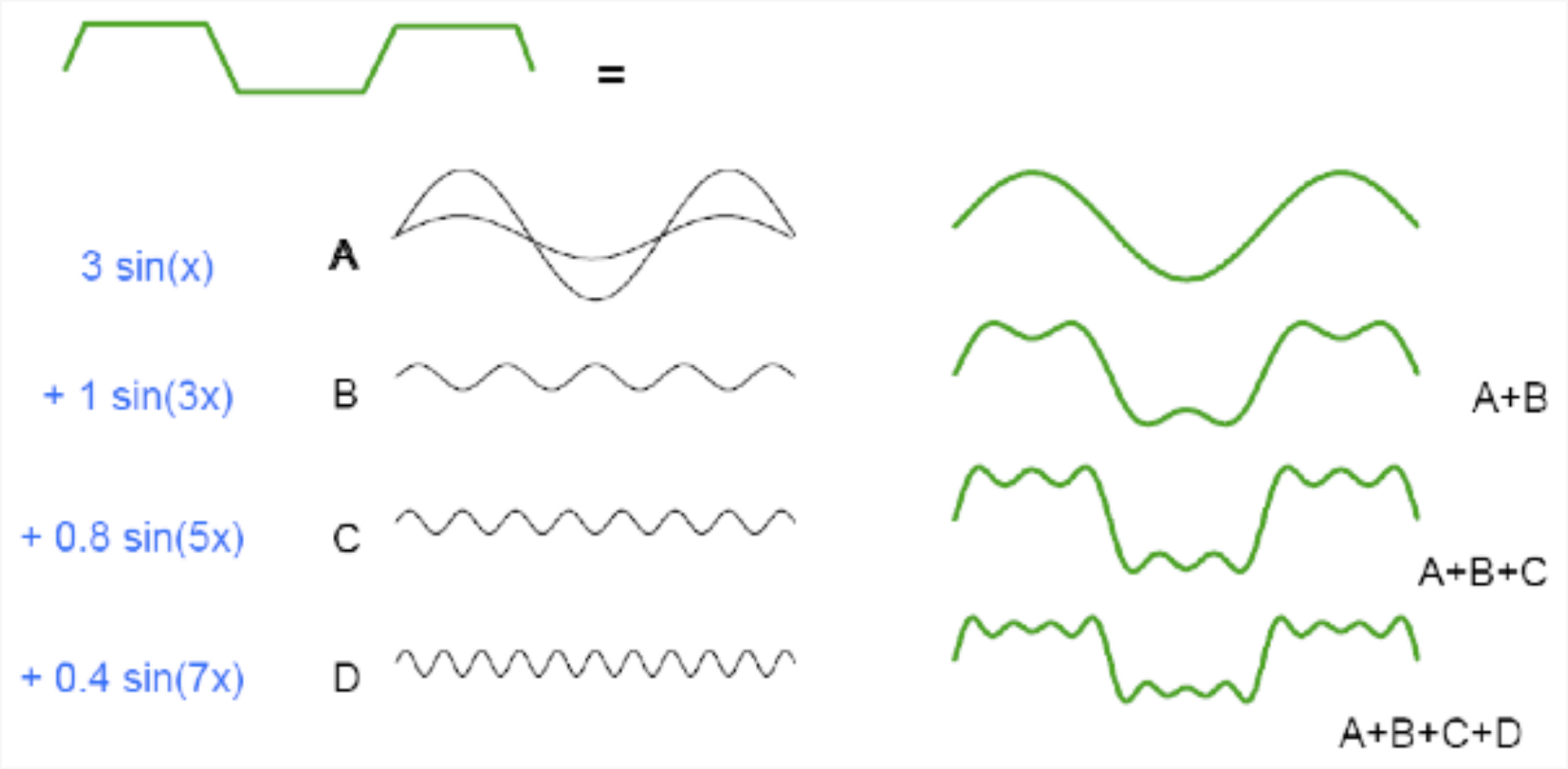
Basisfunktionen der DCT in 1D und 2D



Diskrete Fouriertransformation

DFT

- Approximiere Signal durch eine gewichtete Summe von periodischen Basisfunktionen



- Bilder können als Funktionen mit zwei Parametern aufgefasst werden:
  - f (x, y) = Helligkeitswert
- Eine Fourierreihe ist die Darstellung einer periodischen Funktion als gewichtete Summe von Kosinus- und Sinusfunktionen (hier: Funktion mit einem Parameter):

$$f_n(t) = \frac{a_0}{2} + \sum_{k=1}^{n} (a_k \cdot \cos(k\omega t) + b_k \cdot \sin(k\omega t)).$$

$$a_n = \frac{2}{T} \int_c^{c+T} f(t) \cdot \cos(n\omega t)\,\mathrm{d}t$$

$$b_n = \frac{2}{T} \int_c^{c+T} f(t) \cdot \sin(n\omega t)\,\mathrm{d}t$$

$$\omega = 2\pi / T$$

(Quelle: http://de.wikipedia.org/wiki/Fourier-Reihe)

- Da

$$\cos\theta + i\sin\theta = e^{-i\theta}$$

lässt sich die Formel vereinfachen (hier jetzt zweidimensional):

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(m,n) * e^{-i*2\pi(um+vn)}\, dmdn$$

(Quelle: http://de.wikipedia.org/wiki/Kontinuierliche_Fourier-Transformation)

DFT

- Die allgemeine Fourierreihe arbeitet auf einem kontinuierlichen Signal
- Wenn wir Bilder als Signale interpretieren haben wir diskrete Werte (an jedem Bildpunkt).
- Es ist also nicht mehr notwendig zu integrieren (d.h. die Fläche unter dem Graphen zu berechnen) sondern es reicht die einzelnen Messpunkte zu addieren:

$$F(u,v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) * e^{-i*2\pi*\left(\frac{um}{M}+\frac{vn}{N}\right)}$$

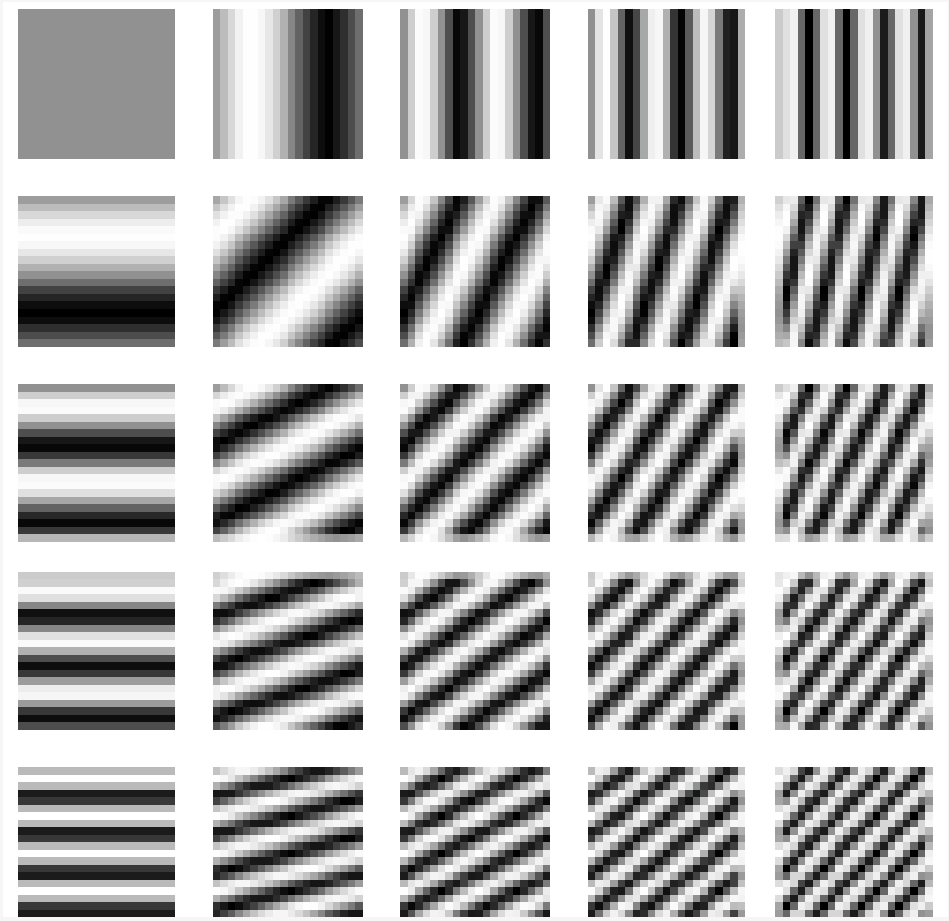
(Quelle: Efford - Digital Image Processing using Java)

- Dabei ist F(u,v) eine komplexe Zahl, d.h. sie hat einen Real- und Imaginärteil
- u ist die Anzahl der Frequenzen entlang der m-Achse (also 0, ..., M - 1)
- v ist die Anzahl der Frequenzen entlang der n-Achse (also 0, ..., N - 1)

$$F(u,v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) * e^{-i*2\pi*\left(\frac{um}{M} + \frac{vn}{N}\right)}$$

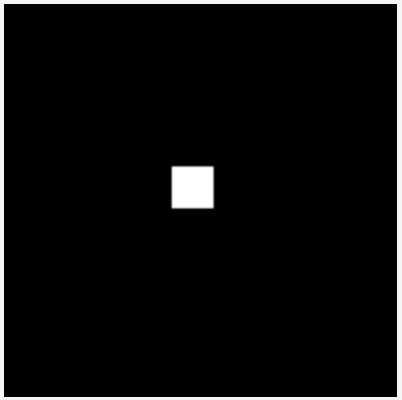
(Quelle: Efford - Digital Image Processing using Java)

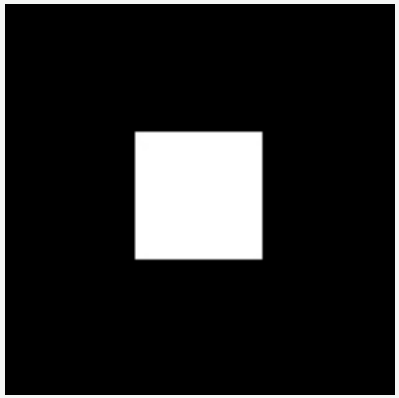
DFT – Graphisch

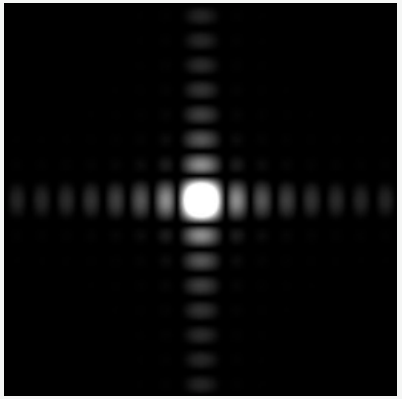


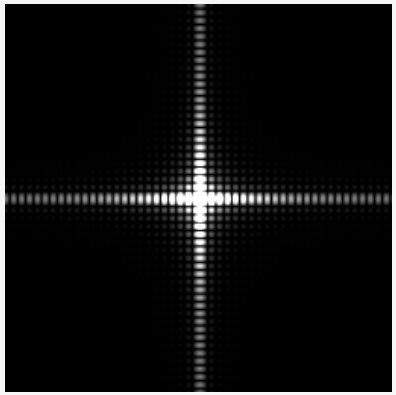
(Quelle: Efford - Digital Image Processing using Java)

DFT – Beispiele



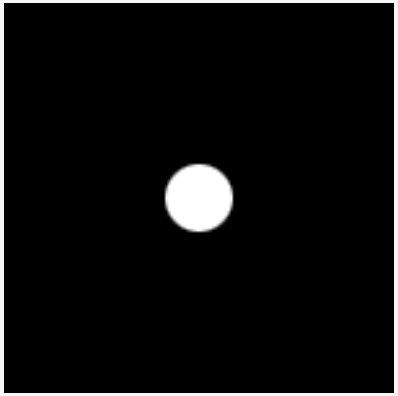


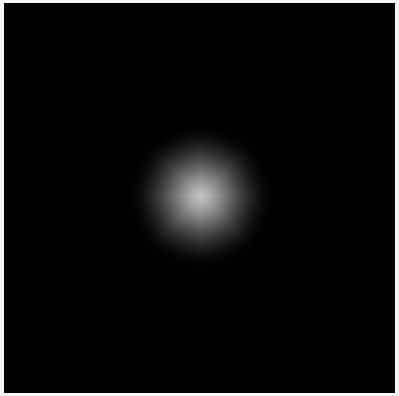


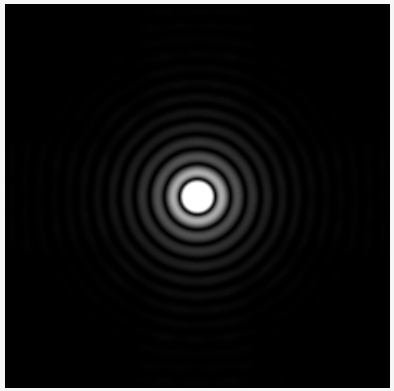


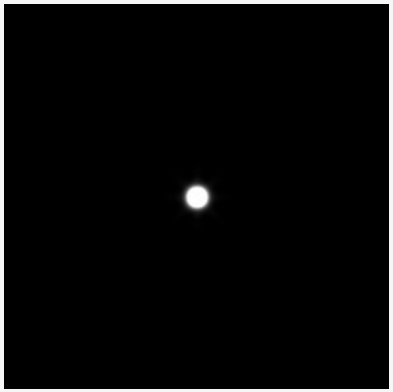
(Quelle: Efford - Digital Image Processing using Java)

DFT – Beispiele









(Quelle: Efford - Digital Image Processing using Java)

DFT – Spektrum

- F(u,v) ist eine komplexe Zahl, aber leider ist sie wenig anschaulich
- Daher konvertiert man F(u,v) normalerweise in das sog. Spektrum
- Dazu nutzt man diese Gleichung:

$$F(u,v) = R(u,v) + iI(u,v) = \left|F(u,v)\right| e^{i\phi(u,v)}$$

- Das Spektrum setzt sich aus Amplitude und Phase zusammen:

$$\left|F(u,v)\right| = \sqrt{R^2(u,v) + I^2(u,v)} \qquad \phi(u,v) = \operatorname{atan} 2(I(u,v), R(u,v))$$

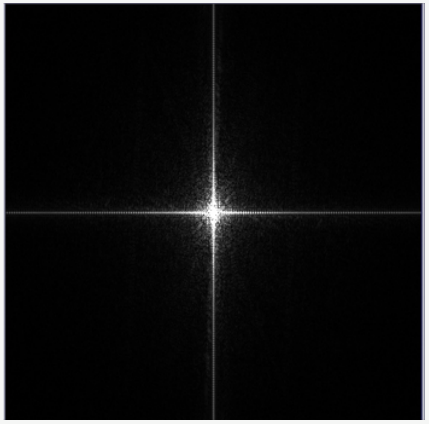
(Quelle: Efford - Digital Image Processing using Java)

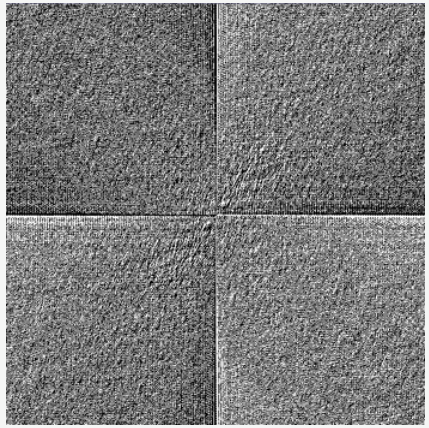
DFT – Spektrum

- Was sagen Amplitude und Phase aus?
- Amplitude gibt die Helligkeit der Bildpunkte wieder
  - Halbieren wir die Amplitude ist das Ergebnisbild nur noch halb so hell
  - Oft Ansatzpunkt für Kompression weil Helligkeitsunterschiede vom menschlichen Auge nicht sehr genau aufgelöst werden
- Phase bestimmt Positionen im Bild
  - Veränderungen verzerren das Bild - deshalb bleibt die Phase normalerweise unverändert

(Quelle: Efford - Digital Image Processing using Java)

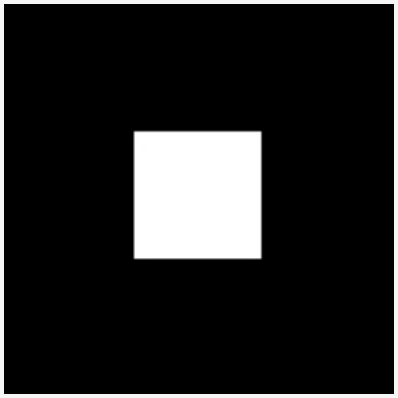
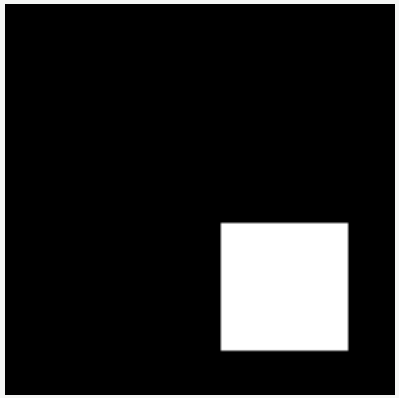
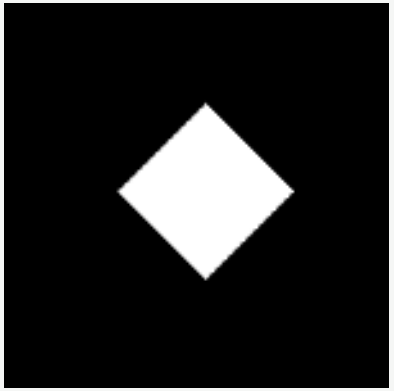
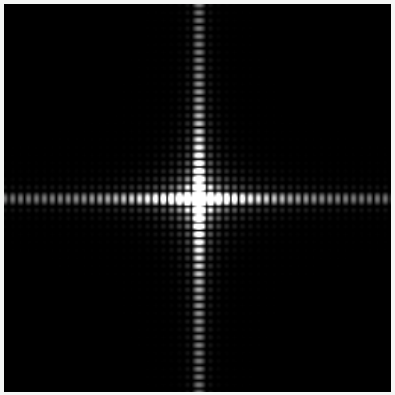
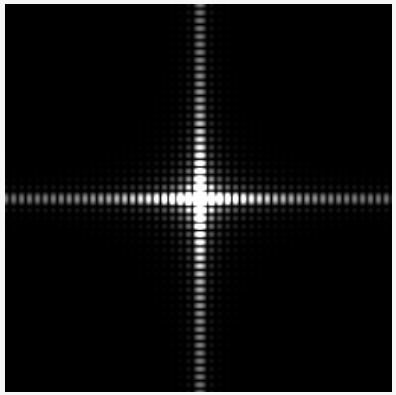
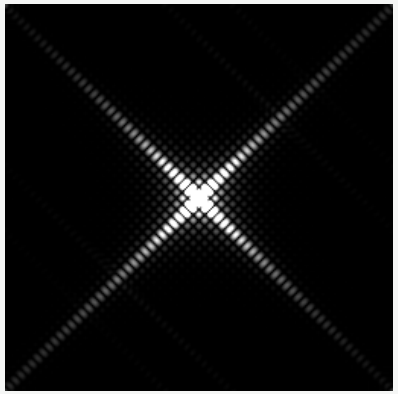
DFT – Spektrum



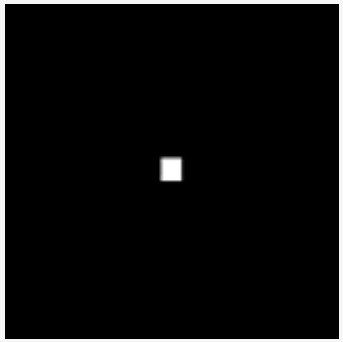
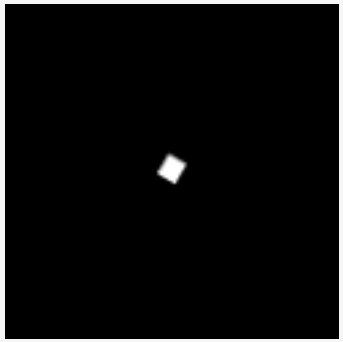
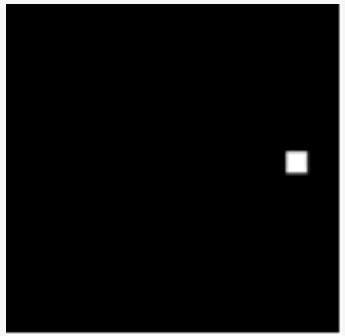
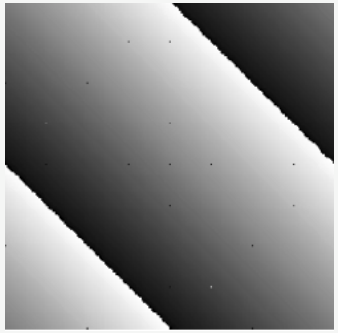
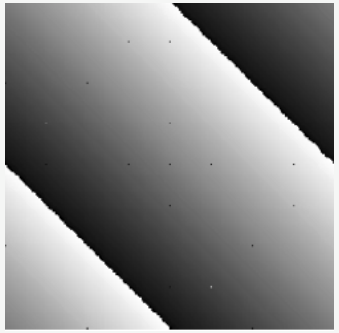
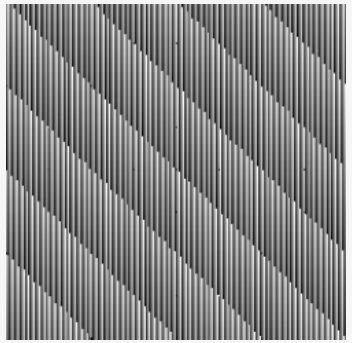


(Quelle: Efford - Digital Image Processing using Java)

DFT – Spektrum

DFT – Spektrum

Schnelle Fouriertransformation

DFT

- Größtes Problem der DFT: Performance!
- Sehr komplexer Algorithmus mit einer Laufzeit von O(N^4) für ein N x N Bild
- Um einen Wert von F(u,v) zu berechnen müssen alle Pixel des Bilds (also N x N) betrachtet werden. Und das muss natürlich für jedes Pixel passieren.
- Unter der Annahme das das Multiplizieren für komplexe Zahlen eine Microsekunde dauert, braucht DFT für ein 1024 x 1024 Bild 12 Tage (!)

(Quelle: Efford - Digital Image Processing using Java)

FFT

- Lösung: Schnelle (Fast) Fouriertransformation
- Verschiedene existierende Ansätze, bei denen verschiedene Eigenarten der Fouriertransformation ausgenutzt werden
- Im Folgenden wird der Algorithmus von Cooley und Tukey (1965) (ursprünglich: Gauss) beschrieben

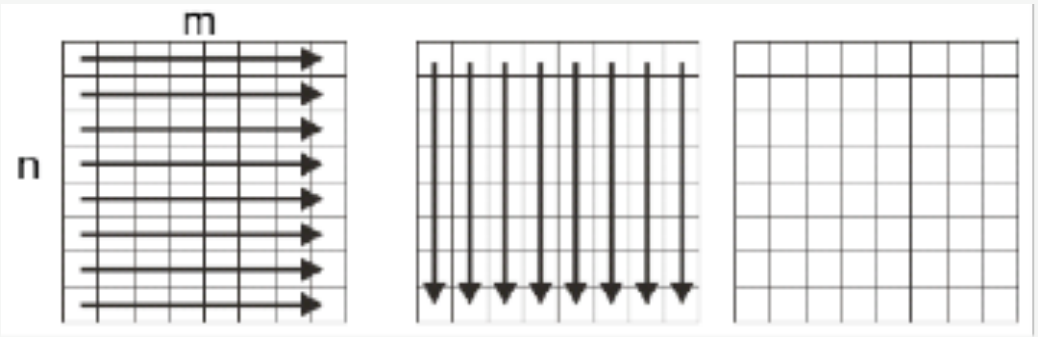
(Quelle: http://en.wikipedia.org/wiki/Cooley-Tukey_FFT_algorithm)

FFT

- Erste Idee: Ausnutzung der Separabilität der FT.
- Eine Funktion heißt linear separierbar wenn gilt:

$$f(x,y) = f(x)g(y)$$

- Die beiden geschachtelten Summationen der FT sind separabel (ähnlich wie manche Konvolutionskernel): Anstatt also die geschachtelte DFT mit O(N^4) auszuführen, werden Zeilen und Spalten nacheinander als lineare FTs berechnet.
- Die sich ergebende Komplexität ist damit O(N^3) (O(N) pro Pixel)



(Quelle: http://en.wikipedia.org/wiki/Cooley-Tukey_FFT_algorithm)

FFT

- Weiterer Trick: Divide-And-Conquer
- Nach dem Separierschritt arbeiten wir nur noch mit eindimensionalen FTs
- Da wir mit einer Summation arbeiten lässt sich die DFT umformulieren. Von:

$$F(u) = \frac{1}{N} \sum_{n=0}^{N-1} f(n)(W_N)^{un}, \quad \text{mit } W_N = e^{-i\frac{2\pi}{N}}$$

nehmen wir jeweils die geraden und ungeraden Koeffizienten (K = N/2):

$$F_{even}(u) = \frac{1}{K} \sum_{n=0}^{K-1} f(2n)(W_{2K})^{(2n)\,u}$$

$$F_{odd}(u) = \frac{1}{K} \sum_{n=0}^{K-1} f(2n+1)(W_{2K})^{(2n+1)\,u}$$

(Quelle: Tönnies - Grundlagen der Bildverarbeitung)

FFT – Divide...

- Nochmal anders formuliert: Die Fourierreihe lässt sich als Polynom betrachten:

$$F(u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3 + a_4 u^4 + \ldots + a_{n-1} u^{n-1}$$

- Wir nehmen dieses Polynom und teilen es in gerade und ungerade Koeffizienten auf:

$$F_{even}(u) = a_0 + a_2 u + a_4 u^2 + \ldots + a_{n-2} u^{n/2-1}$$

$$F_{odd}(u) = a_1 + a_3 u + a_5 u^2 + \ldots + a_{n-1} u^{n/2-1}$$

- Das lässt sich natürlich beliebig wiederholen
- Falls N eine Zweierpotenz ist können wir sogar teilen bis wir am Ende N Polynome haben

(Quelle: http://en.wikipedia.org/wiki/Cooley-Tukey_FFT_algorithm)

FFT – ... and ...

- Wenn wir die beiden Formeln etwas anders ausdrücken:

$$F_{even}(u) = \frac{1}{K}\sum_{n=0}^{K-1} f(2n)(W_{2K})^{(2n)\,u}$$

$$F_{odd}(u) = \frac{1}{K}\sum_{n=0}^{K-1} f(2n+1)(W_{2K})^{(2n)\,u}$$

können wir die ursprüngliche (komplette) Formel so schreiben:

$$F(u) = \frac{1}{2}\left(F_{even}(u) + F_{odd}(u)\,(W_{2K})^{u}\right)$$

(Quelle: Tönnies - Grundlagen der Bildverarbeitung)

FFT – ... Conquer

- Bisher haben wir noch nichts dadurch gewonnen, da wir weiterhin die gleiche Anzahl von Rechenschritten haben (statt einem großen Polynom haben wir N kleine Polynome).
- W_N heißt primitive n-te Einheitswurzel. Für diese gelten folgende Eigenschaften:

$$(W_K)^{u+N} = (W_K)^u \quad \text{und} \quad (W_{2K})^{u+N} = -(W_{2K})^u$$

- Das heißt also dass sich W_N periodisch wiederholt (nach einem Durchlauf beginnt es wieder von vorne)
- Wir können also die berechneten Werte von 0 bis K - 1 nutzen um die Werte von K bis N - 1 zu berechnen und sparen uns die Hälfte des Aufwands!

(Quelle: Tönnies - Grundlagen der Bildverarbeitung)

FFT – ... Conquer

- Wir wandeln nochmal die Formel um:

$$F(u+K) = \frac{1}{2}\left(F_{even}(u+K) + F_{odd}(u+K)(W_{2K})^u\right)$$

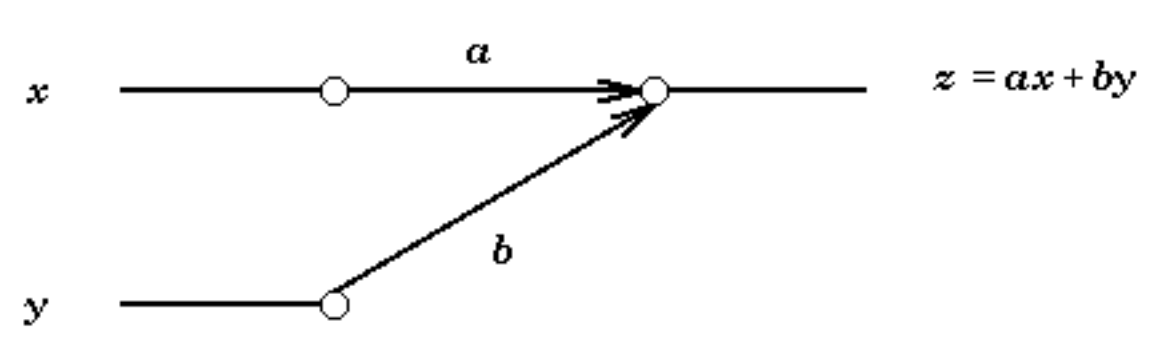
$$F(u+K) = \frac{1}{2}\left(F_{even}(u) - F_{odd}(u)(W_{2K})^u\right)$$

- Für 0, ..., K - 1 nutzen wir die ursprüngliche Formel, für K, ..., N - 1 die obenstehende.

(Quelle: Tönnies - Grundlagen der Bildverarbeitung)

FFT – ... Conquer

- Weil wir nicht nur einmal, sondern log(N) mal teilen brauchen wir auch entsprechend viele Conquer-Schritte und müssen die Ergebnisse miteinander verrechnen
- Dazu nutzt man die “Butterfly” (Schmetterlings) Notation.
  - Für jeden Wert wird eine Gerade angetragen
  - Ein Kreis mit zwei Pfeilen symbolisiert eine Addition der beiden Werte, kann aber auch beliebige andere Operatoren enthalten
  - Vorher wird noch mit eventuellen Faktoren die an den Pfeilen stehen multipliziert.



(Quelle: http://astro.berkeley.edu/~jrg/ngst/fft/fftbutfy.html)

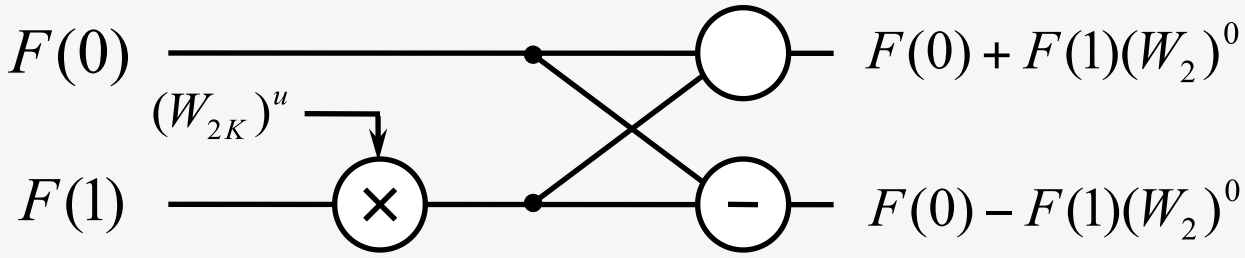
FFT – ... Conquer

- Der Conquer-Schritt für unsere Berechnungsformeln von vorher:

$$F(u) = \frac{1}{2}(F_{even}(u) + F_{odd}(u)(W_{2K})^u), \text{ für } u = 0,\ldots,K-1$$

$$F(u+K) = \frac{1}{2}(F_{even}(u) - F_{odd}(u)(W_{2K})^u), \text{ für } u = K,\ldots N-1$$

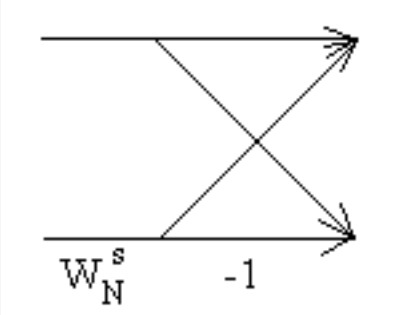
- lässt sich in dieses Diagramm umwandeln:



(Quelle: http://astro.berkeley.edu/~jrg/ngst/fft/fftbutfy.html)

FFT – ... Conquer

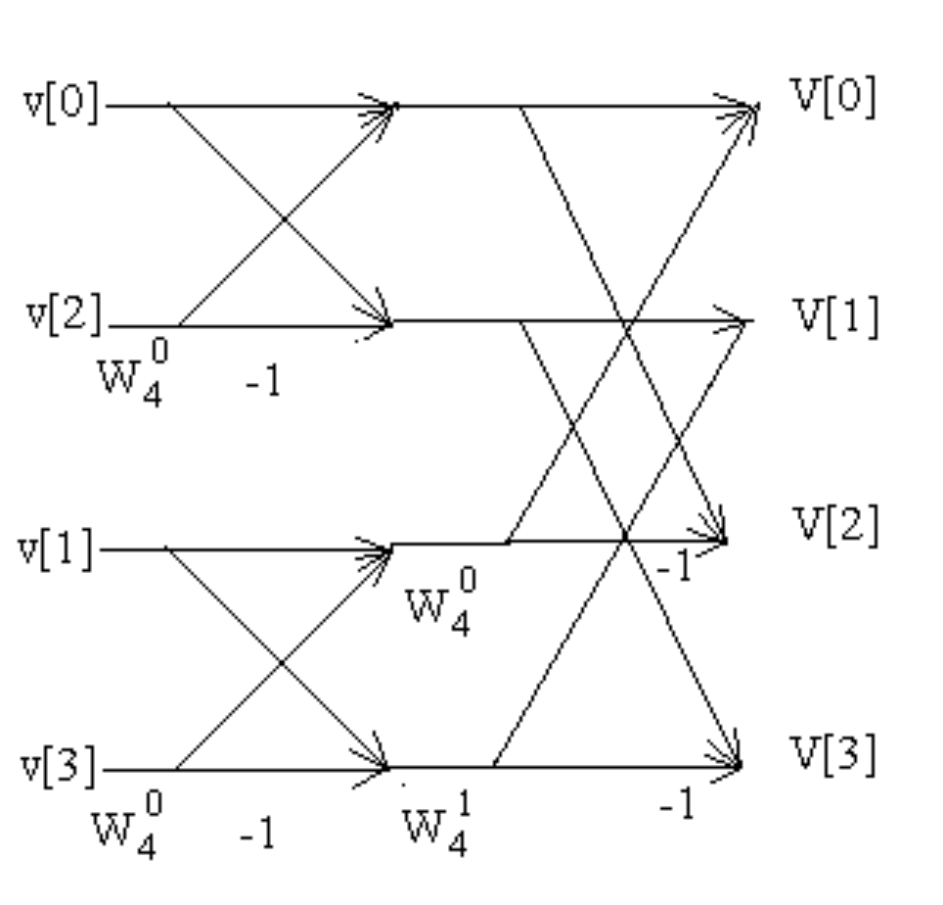
- Der allgemeine Butterfly ist:



- Dieser kann auch mehrfach hintereinander angewendet werden.

(Quelle: http://astro.berkeley.edu/~jrg/ngst/fft/fftbutfy.html)

FFT – ... Conquer



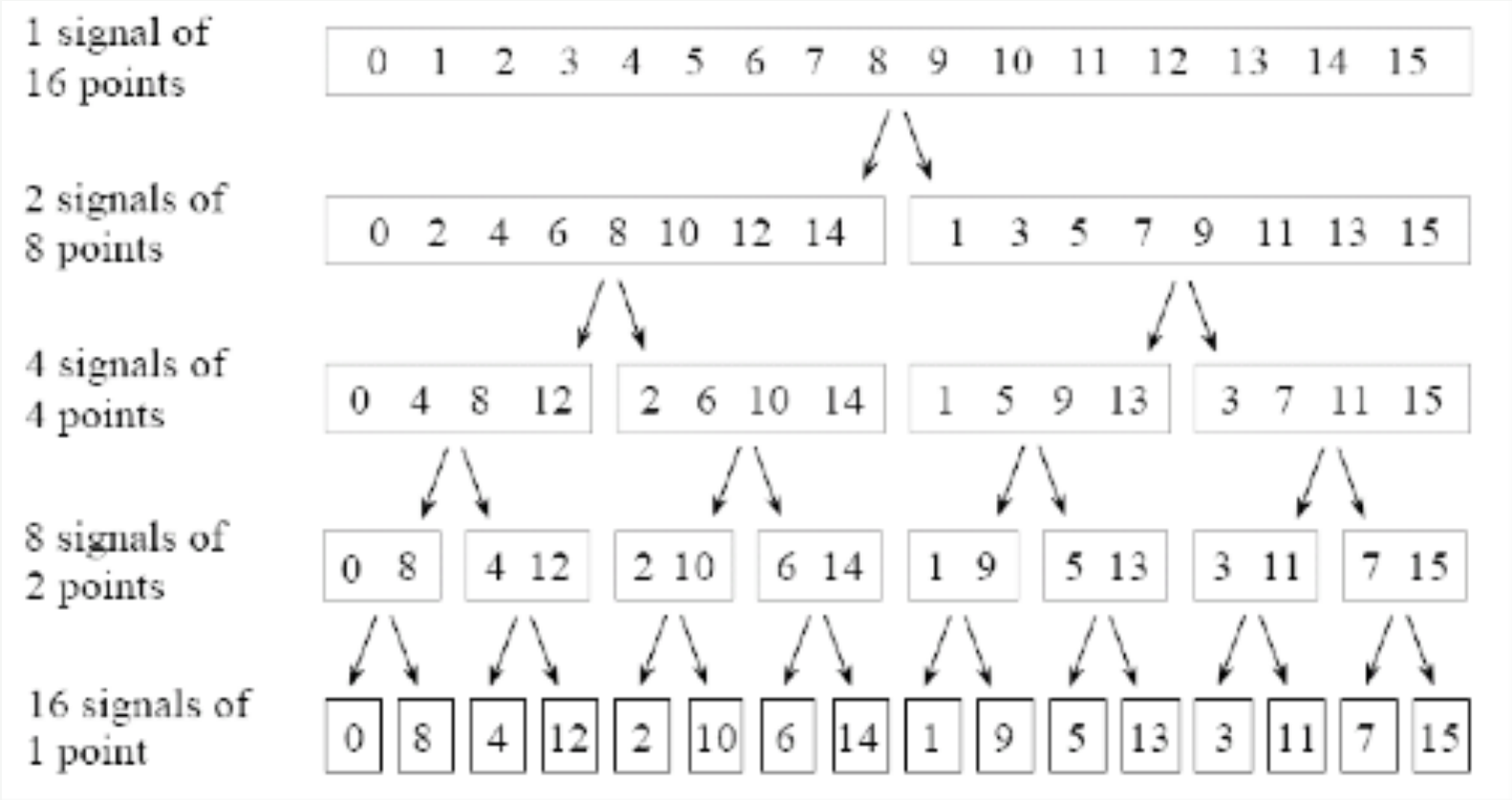
(Quelle: http://www.relisoft.com/science/Physics/fft.html)

FFT – ... Conquer

- Die FFT kann in einem Puffer mit N Werten durchgeführt werden.
- Dabei müssen zuerst die Signale umsortiert und danach der Butterfly Algorithmus mehrfach angewendet werden
- Diese Umsortierung läuft folgendermaßen (Beispiel: N=8):
  - (0, 1, 2, 3, 4, 5, 6, 7) wird aufgeteilt in (0, 2, 4, 6) und (1, 3, 5, 7)
  - (0, 2, 4, 6) (1, 3, 5, 7) werden zu (0, 4) (2, 6) (1, 5) (3, 7)
  - 0, 4, 2, 6, 1, 5, 3, 7 ist also die finale Reihenfolge im Puffer

(Quelle: http://www.relisoft.com/science/Physics/fft.html)

FFT – ... Conquer


1 signal of 16 points
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
2 signals of 8 points
0 2 4 6 8 10 12 14
1 3 5 7 9 11 13 15
4 signals of 4 points
0 4 8 12
2 6 10 14
1 5 9 13
3 7 11 15
8 signals of 2 points
0 8
4 12
2 10
6 14
1 9
5 13
3 11
7 15
16 signals of 1 point
0 8 4 12 2 10 6 14 1 9 5 13 3 11 7 15

FFT – Zusammenfassung

- Zusammenfassung FFT:
  - N als Zweierpotenz wählen (notfalls mit Nullen auffüllen)
  - Umsortieren der Werte
  - Ausführen des Butterfly Algorithmus mit benachbarten Paaren im Puffer
  - Ausführen des Butterfly Algorithmus mit Paaren von Abstand zwei
  - Ausführen des Butterfly Algorithmus mit Paaren von Abstand vier
  - etc.
  - Solange weitermachen bis die Teilung N / 2 ist - der Puffer enthält die FT

(Quelle: http://www.relisoft.com/science/Physics/fft.html)

Weiterführende Literatur:

- Nick Efford: “Digital Image Processing – a practical introduction using Java”, ISBN-13: 978-0201596236
- Klaus D. Tönnies: “Grundlagen der Bildverarbeitung”, ISBN-13: 978-3827371553
- *http://www.relisoft.com/science/Physics/fft.html*
- *http://astro.berkeley.edu/~jrg/ngst/fft/fftbutfy.html*