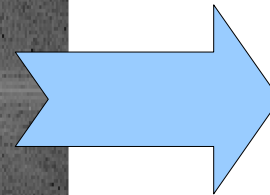
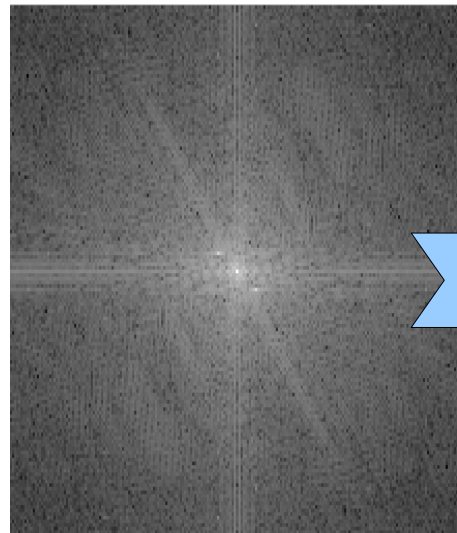
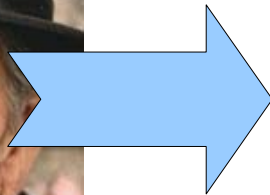


# *Fouriertransformation*

Radix2 fast fourier transform nach Cooley/Tukey





# Inhaltsübersicht

---

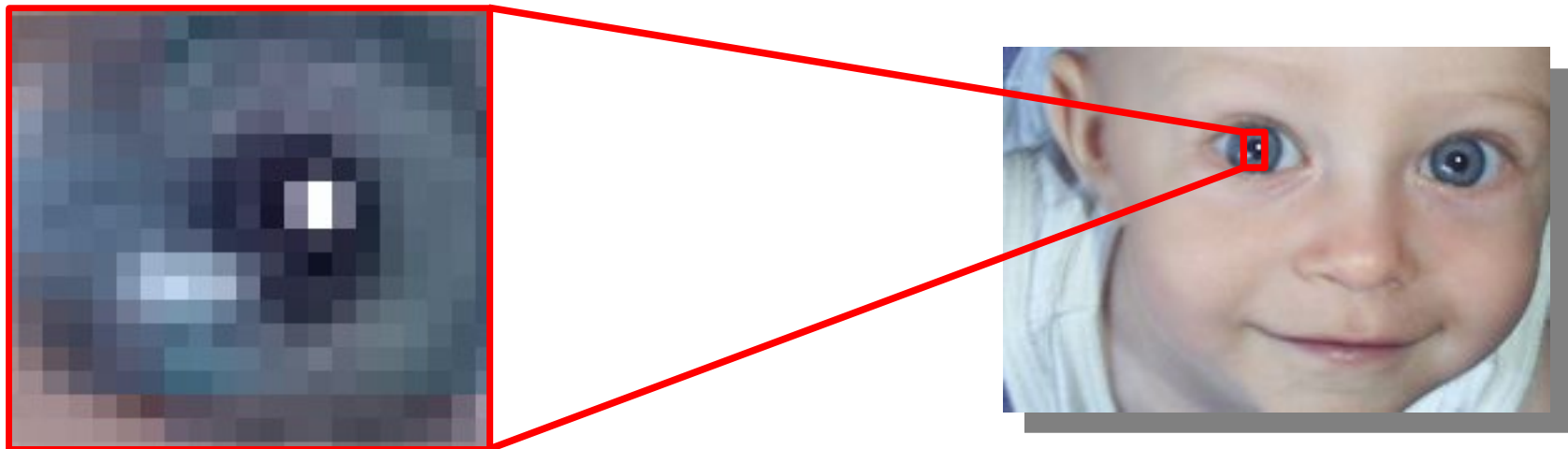
- ▶ Mathematische Grundlagen: Komplexe Zahlen und Einheitswurzeln
- ▶ Die diskrete Fouriertransformation
- ▶ Der Radix2-Algorithmus nach Cooley-Tukey
- ▶ Weiterführende Performanceüberlegungen
- ▶ Quellen und Literatur



# Ortsraum und Vektordarstellung

---

- ▶ Bilder werden im Allgemeinen als Kombination einzelner Pixel verstanden
- ▶ Jeder Pixel hat einen ihm zugeordneten Farbwert (z.B. RGB)
- ▶ Jeder Pixel hat eine Koordinate (x und y)

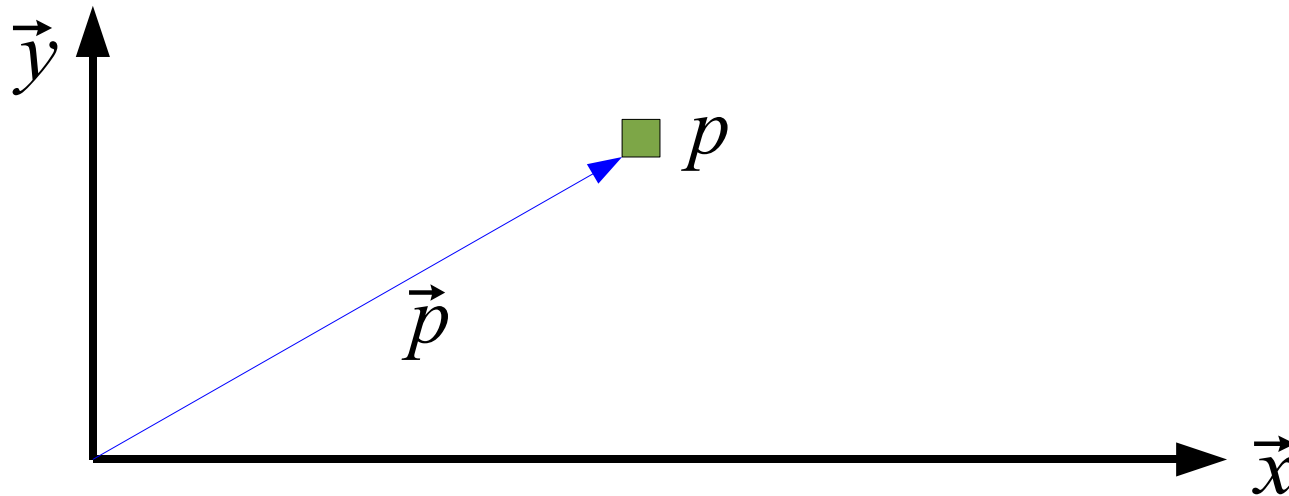




# Ortsraum und Vektordarstellung (2)

---

- Idee: Jeder Bildpunkt kann als Einzelbild angesehen werden
- ➔ Ein Bild der Größe  $n*m$  besteht dann aus  $n*m$  Einzelbildern
- Die Position jedes Bildpixels wird über einen Vektor im zweidimensionalen Raum bestimmt:





# Ortsraum und Vektordarstellung (3)

---

- ◆ Der so definierte Vektorraum wird als *Ortsraum* definiert
- ◆ Es ist leicht vorstellbar, ein Bild in einen anderen Raum zu übertragen, z.B. in einen 3-dimensionalen Vektorraum
- ◆ Viele Arten, ein Bild darzustellen, sind denkbar, aber nur wenige Darstellungen bieten Vorteile.
- ◆ Die Fouriertransformation überträgt ein Bild in den *Frequenzraum*. Hier ergeben sich große Vorteile in der Bild- und Signalverarbeitung.



# Komplexe Zahlen

---

- ◆ Kurz zur Grundlage:
  - ◆ Komplexe Zahlen erweitern den Zahlenraum der reellen Zahlen
  - ◆ Komplexe Zahlen erlauben Lösungen für die Wurzeln negativer Zahlen
  - ◆ Einführung der Einheit  $i$  als Lösung der Gleichung

$$x^2 = -1$$



# Einheitswurzeln

---

- ♦ Zahlen, deren n-te Potenz 1 ergibt, werden als n-te *Einheitswurzeln* bezeichnet.
- ♦  $\omega$  ist n-te Einheitswurzel, wenn gilt:  $\omega^n = 1$
- ♦ Es gilt (ohne Beweis):

$$\omega = \cos(k \cdot 2\pi/n) + i \cdot \sin(k \cdot 2\pi/n)$$

für  $k = 0 \dots n-1$

---



# Einheitswurzeln (2)

---

- ♦ Zahlen, deren n-te Potenz 1 ergibt, werden als n-te *Einheitswurzeln* bezeichnet.
- ♦  $\omega$  ist die primitive n-te Einheitswurzel, wenn gilt:

$$\omega^n = 1, \text{ aber: } \omega^k \neq 1 \text{ für } k \in \{1..n-1\}$$

- ♦ Beispiel: Sei  $n = 4$ , dann ist  $i$  die 4. primitive Einheitswurzel

$$i^0 = 1, i^1 = i, i^2 = -1, i^3 = -i$$





# Jean Baptiste Fourier

---

- ♦ Französischer Mathematiker
- ♦ Lebte von 1768 – 1830
- ♦ Wird oft als Genie verstanden: Im Alter von 14 Jahren hatte er bereits die Standardwerke der Mathematik studiert



*“Yesterday was my 21st birthday, at that age Newton and Pascal had already acquired many claims to immortality.”*



# Fourier Analyse

---

- ◆ Fouriers Idee: Jede periodische Funktion lässt sich durch sinus- und cosinus-Funktionen unterschiedlicher Frequenzen darstellen:





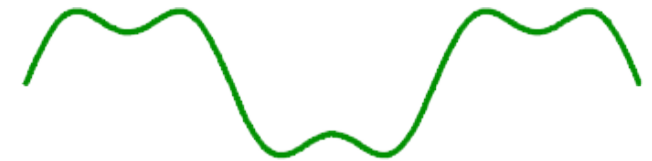
# Fourier Analyse (2)

---

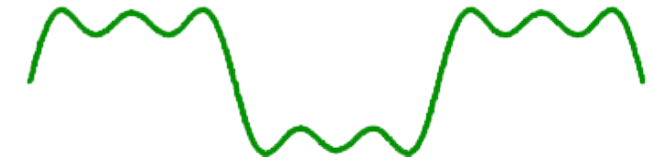
$$3\sin(x)$$



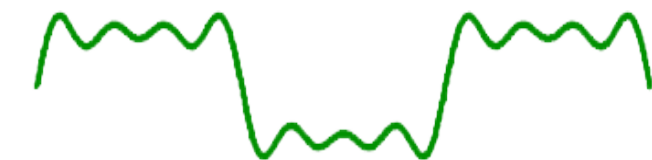
$$+\sin(3x)$$



$$+0.8\sin(5x)$$



$$+0.4\sin(7x)$$





# Transformation von Bildern

---

- ▶ Periodische Signale lassen sich somit einfach in sinus- und cosinus-Anteile zerlegen
- ▶ Bilder sind aber in aller Regel nicht periodisch
- ▶ Idee: Nicht periodische Bereiche einer Funktion lassen sich durch Aneinanderreihung von Kopien periodisieren

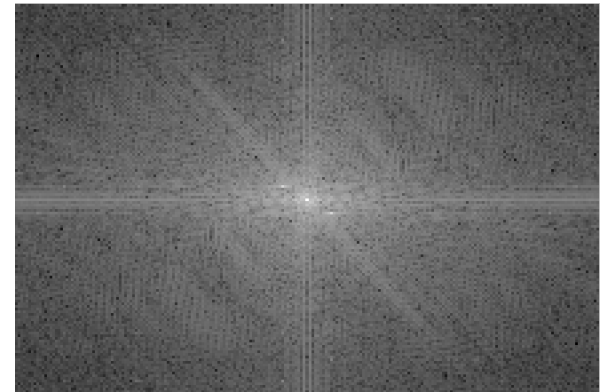
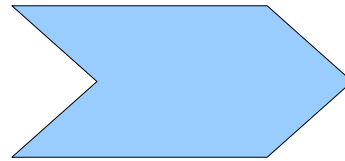
Ein Bild ist eine Matrix nicht-periodischer Funktionen



# Die Fouriertransformation

---

- ◆ Bislang wurde die Darstellung von Bildern im *Ortsraum* betrachtet – jetzt: *Frequenzraum*





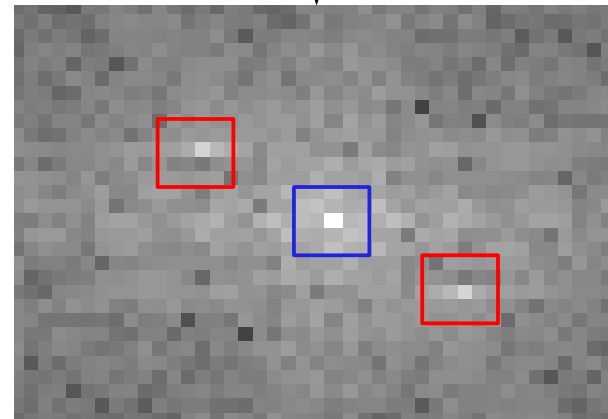
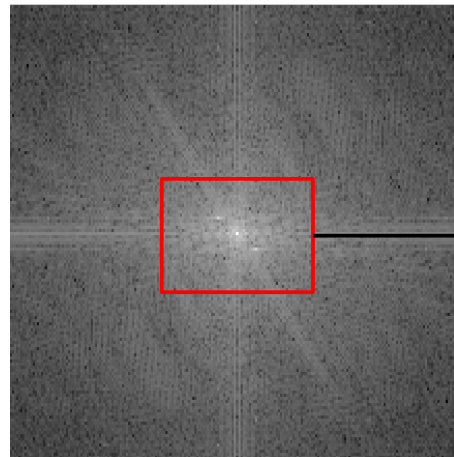
# Aber warum das Ganze?


---

- ▶ Auf den ersten Blick ist im Frequenzraum rein garnichts zu erkennen
- ▶ Die Übertragung in den Frequenzraum hat jedoch viele Vorteile, z.B.:
  - Viele Filteroperationen (z.B. Mittelwertfilter) können deutlich schneller durchgeführt werden
  - Periodische Störungen im Signal können beseitigt werden



# Ein Beispiel



 Summe der Pixelwerte

 Periodische Störungen



## Ein Beispiel (2)

---

- ◆ Überträgt man ein Bild mit einer periodischen Störung in den Frequenzraum, so wird die Störung als heller Punkt in der Nähe des Bildmittelpunktes sichtbar
  - ◆ Eine horizontale Störung liegt dann z.B. rechts und links neben dem Mittelpunkt (da sich die Störung in beide Richtungen unendlich fortpflanzt)
  - ◆ Die Entfernung vom Mittelpunkt gibt die Frequenz der Störung an: 10 Pixel entsprechen z.B. 10 Schwingungen innerhalb des Bildes
-





## Ein Beispiel (3)

---

- Werden nun diese Störungspixel aus dem Bild entfernt und das Bild anschließend zurücktransformiert, so ist die Störung vollständig verschwunden, ohne dass die eigentliche Information verloren geht
- Der helle Punkt in der Mitte entspricht der Summe aller Bildpixel, daher ist der Mittelpunkt immer klar als hellster (weißer) Punkt zu erkennen



# Diskrete Fouriertransformation

---

- Die Übertragung eines Signals in den Frequenzraum erfolgt mit Hilfe der folgenden Formel:

$$f(u) = \sum_{r=0}^{N-1} G_r \cdot e^{\frac{-i2\pi}{N} \cdot ru}$$

- Für jeden Wert  $x$  des Signals  $G$  müssen alle  $N$  Werte des Signals durchlaufen werden
  - Der Exponent der e-Funktion ist komplex
-



# Diskrete Fouriertransformation (2)

---

- Bilder sind nicht als eindimensionales Signal darstellbar, daher wird hier die Formel nochmals komplexer:

$$f(u, v) = \frac{1}{N} \sum_{c=0}^{N-1} \sum_{r=0}^{N-1} G_{rc} \cdot e^{\frac{-i2\pi}{N} \cdot (ur + vc)}$$

- Bei N Pixeln in einem Bild müssen danach  $N^2$  komplexe Exponenten zu e berechnet werden



# Diskrete Fouriertransformation (3)

---

Die komplexe e-Funktion

$$e^{\frac{-i 2 \pi}{N} \cdot ru}$$

lässt sich dank der eulerschen Identität

$$e^{i \cdot \alpha} = \cos(\alpha) + i \cdot \sin(\alpha)$$

in eine komplexe Zahl umwandeln, deren Realteil durch die Sinus- und deren Imaginärteil durch die Cosinusfunktion bestimmt wird.

---



# Diskrete Fouriertransformation (4)

---

- ◆ Unser  $\alpha$  ist in diesem Falle

$$\alpha = \frac{-2\pi ur}{N}$$

- ◆ Damit lässt sich die e-Funktion darstellen als

$$e^{\frac{-i2\pi}{N} \cdot ru} = \cos\left(\frac{-2\pi ur}{N}\right) + i \cdot \sin\left(\frac{-2\pi ur}{N}\right)$$



# Diskrete Fouriertransformation (5)

---

- ♦ Vergleichen wir nun den Term

$$\cos\left(\frac{-2\pi ur}{N}\right) + i \cdot \sin\left(\frac{-2\pi ur}{N}\right)$$

mit der bekannten Formel für die Einheitswurzel  $\omega$ :

$$\omega^k = \cos\left(\frac{2\pi \cdot k}{N}\right) + i \cdot \sin\left(\frac{2\pi \cdot k}{N}\right)$$





# Diskrete Fouriertransformation (6)

---

- Die e-Funktion für die Fouriertransformation entspricht somit der Berechnung der Einheitswurzeln
- Idee: Die Werte der e-Funktion (und damit der Einheitswurzeln) können gecached werden, um die Berechnungszeit zu verkürzen.
- Für Signale gleicher Länge  $N$  lässt sich die berechneten Werte einfach wiederverwenden



# Fourier-Matrix

---

- Die primitive Einheitswurzel bzw. deren Potenzen lassen sich in einer Matrix darstellen, der sogenannten Fouriermatrix
- Soll ein Vektor in den Fourierraum übertragen werden, so genügt eine Multiplikation des Vektors mit der Matrix

$$\omega_4 = i$$

$$F = \begin{bmatrix} w^0 & w^0 & w^0 & w^0 \\ w^0 & w^1 & w^2 & w^3 \\ w^0 & w^2 & w^4 & w^6 \\ w^0 & w^3 & w^6 & w^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$





# Fourier-Matrix (2)

- Die Fouriermatrix entspricht der sogenannten *Vandermondematrix* von  $\omega$
- Beispiel:

$$\begin{array}{c|c} & \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} \leftarrow F \\ \hline \vec{a} \nearrow [1 & 1 & 1 & 0] & [3 & i & 1 & -i] \leftarrow \vec{b} = \vec{a} \cdot F \end{array}$$



# Fourier-Matrix (3)

---

- ◆ Eine Eigenschaft der zweidimensionalen DFT ist es, dass sie auch als Konkatination eindimensionaler DFTs dargestellt werden kann.
- ◆ Ein Bild wird dann erst zeilenweise, dann spaltenweise transformiert
- ◆ Die Zeilen/Spalten eines Bildes haben stets die gleiche Länge  
→ hier kann die gleiche Fouriermatrix für alle Zeilen/Spalten angewendet werden.



# Der Weg zur FFT

---

- ♦ Auch bei einer denkbar günstigen Zwischenlagerung der Fouriermatrix bleibt die Laufzeit des Algorithmus bei  $O(n^2)$
- ♦ Interessant wäre ein divide-and-conquer-Verfahren zur Berechnung, aber: Wie bzw. Wo trennt man das Signal?
- ♦ Eine Trennung in der Hälfte kann nicht funktionieren: Eine periodische Schwingung, die durch das ganze Bild verläuft, könnte so getrennt werden, nicht aber alle übrigen Signalwerte



# Der Weg zur FFT (2)

---

- ▶ 1965 veröffentlichten *James W Cooley* und *James W. Tukey* ein Verfahren zur Berechnung der Fouriertransformation in  $O(n \cdot \log_{(n)})$  Laufzeit
- ▶ Der entsprechende Artikel erschien unter dem Titel '*An algorithm for the machine calculation of complex fourier series*' und ist im Anhang als PDF enthalten
- ▶ Der Algorithmus von Cooley-Tukey war der erste FFT-Algorithmus und machte die FFT für die EDV erst interessant



# Eigenschaften der FFT

---

- ♦ Der Cooley-Tukey-Algorithmus ist bis heute der schnellste bekannte FFT-Algorithmus
  - ♦ Die CFFT funktioniert lediglich, wenn die Länge des Signals eine Potenz von 2 ist (andere, langsamere Algorithmen können auch mit anderen Signallängen umgehen)
  - ♦ => In der Regel ist es notwendig, das Signal zu 'padding', d.h. künstlich mit 0 bis zur nächsten vollen 2er-Potenz aufzufüllen
  - ♦ Für viele Anwendungsfälle macht das padding die CFFT wieder uninteressant im Vergleich zu anderen FFTs
-



# Eigenschaften der FFT (2)

---

- ♦ Eine besondere Stärke der Cooley-Tukey-FFT ist, dass kein zusätzlicher Speicherplatz für die Transformation eines Signals verbraucht wird
- ♦ Dieser Aspekt ist grade für die Bildverarbeitung interessant, da der Speicherplatzbedarf bei großen Bildern schnell 'explodiert'
- ♦ Auch sehr große Signale können somit mit relativ geringem Verbrauch an Hauptspeicher verarbeitet werden



# Cooley-Tukey-FFT - Idee

---

- Die Idee der CTFFT ist es, die Matrix-Vektor-Multiplikationen in einer bestimmten Reihenfolge auszuführen, die die Wiederverwendung einzelner Zwischenergebnisse erlaubt
- Es können zwei Eigenschaften der Einheitswurzeln dazu ausgenutzt werden:

$$w^{n/2} = -1 \quad [1]$$

$$w_n^2 = w_{n/2} \quad [2]$$



# Cooley-Tukey-FFT

---

- Sei  $b$  die Fouriertransformierte eines Signalvektors  $a$ , d.h.

$$b = a \cdot F$$

- Zur Erinnerung: Berechnung der  $k$ -ten Komponente des Vektors  $b$ :

$$b_k = \sum_{i=0}^{n-1} a_i \cdot w^{ik} \quad \text{für } k = 0 \dots n-1$$

- Berechnet werden nun zunächst alle Komponenten von  $b$  mit geradem Index:

$$b'_k = b_{2k} = \sum_{i=0}^{n-1} a_i \cdot w^{i2k} \quad \text{für } k = 0 \dots n/2-1$$





# Cooley-Tukey-FFT (2)

---

- ◆ Diese Summe lässt sich problemlos auch in zwei Teile aufspalten:

$$b_k' = \sum_{i=0}^{n/2-1} a_i \cdot w^{i2k} + \sum_{i=0}^{n/2-1} a_{i+n/2} \cdot w^{(i+n/2) \cdot 2k}$$

- ◆ Es gilt aber:

$$w^{(i+n/2) \cdot 2k} = w^{i2k + nk} = w^{i2k} \cdot w^{nk} = w^{i2k}$$

- ◆ da  $w^{nk}$  wie schon definiert gleich 1 ist
-



# Cooley-Tukey-FFT (3)

---

- ▶ Mit diesem Wissen können wir den Term

$$b_k' = \sum_{i=0}^{n/2-1} a_i \cdot w^{i2k} + \sum_{i=0}^{n/2-1} a_{i+n/2} \cdot w^{(i+n/2) \cdot 2k}$$

- ▶ vereinfachen zu

$$b_k' = \sum_{i=0}^{n/2-1} (a_i + a_{i+n/2}) \cdot w^{i2k}$$



# Cooley-Tukey-FFT (4)

---

- Sei nun  $m = n/2$  und  $v = w^2$ , so ist  $v$  damit nach [2] die  $m$ -te primitive Einheitswurzel

$$b_k' = \sum_{i=0}^{m-1} (a_i + a_{i+m}) \cdot v^{ik}$$

- Das heißt aber nichts anderes, als dass  $b_k'$  die  $k$ -te Komponente des Vektors  $(a_i + a_{i+m})$  mit  $i = 0..m-1$  ist
  - Dieser Vektor hat die Länge  $m$  oder besser  $n/2$ : Damit wird ein divide-and-conquer-Verfahren möglich!
-



# Cooley-Tukey-FFT (5)

---

- Nun gilt es, eine entsprechende Berechnung für die Komponenten des Vektors  $b$  zu finden, die einen ungeraden Index besitzen. Wir beginnen dazu analog mit:

$$b_k'' = b_{2k+1} = \sum_{i=0}^{n-1} a_i \cdot w^{i \cdot (2k+1)}$$

$$b_k'' = \sum_{i=0}^{n/2-1} a_i \cdot w^{i \cdot (2k+1)} + \sum_{i=0}^{n/2-1} a_{i+n/2} \cdot w^{(i+n/2) \cdot (2k+1)}$$



# Cooley-Tukey-FFT (6)

---

- Wiederum lässt sich der Exponent der Einheitswurzel im zweiten Summanden vereinfachen:

$$w^{(i+n/2) \cdot (2k+1)} = w^{i2k+i+nk+n/2}$$

- da aber gilt  $w^{nk} = 1$  und  $w^{n/2} = -1$  ergibt sich

$$w^{(i+n/2) \cdot (2k+1)} = -w^i + w^{i2k}$$



# Cooley-Tukey-FFT (7)

---

♦ Damit ist dann

$$b_k'' = \sum_{i=0}^{n/2-1} a_i \cdot w^i \cdot w^{i2k} + \sum_{i=0}^{n/2-1} -a_{i+n/2} \cdot w^i \cdot w^{i2k}$$

$$b_k'' = \sum_{i=0}^{n/2-1} (a_i - a_{i+n/2}) \cdot w^i \cdot w^{i2k}$$



# Cooley-Tukey-FFT (8)

---

- ◆ Setzen wir nun wieder  $m=n/2$  und  $v=w^2$ , so erhalten wir den Term

$$b_k'' = \sum_{i=0}^{m-1} (a_i - a_{i+m}) \cdot w^i \cdot v^{ik}$$

- ◆ Damit ist die k-te Komponente von b nichts anderes als die Fouriertransformierte des Vektors

$$\vec{a}'' = w^i \cdot (a_i - a_{i+m}) \text{ für } i = 0..m-1$$



# Rekursion

---

- Die Vektoren  $a'$  und  $a''$  mit je der Länge  $m$  können nun rekursiv berechnet werden
  - Die Rekursion terminiert, wenn die Länge  $m$  des Vektors  $a'$  kleiner als 2 ist (dann muss schließlich nichts mehr transformiert werden)
  - Das Ergebnis des Algorithmus besteht aus den Vektoren  $b'$  und  $b''$ , die erst zum Ergebnis zusammengefügt werden müssen
  - Für die Vereinigung der beiden Vektoren wendet man ein Reißverschlußartiges Verfahren an
-





# Zeitkomplexität

---

- Es ist nun gezeigt worden, dass sich die Berechnung eines Vektors  $b$  auf die Berechnung der kleineren Teilvektoren  $b'$  und  $b''$  abbilden lässt. Offen ist die Frage nach der Laufzeit des Algorithmus:
- Um die Fouriertransformation für einen Vektor  $a$  zu berechnen ist es notwendig, einen Vektor  $a'$  der Länge  $m$  zu berechnen:

$$a_i' = a_i + a_{i+m}$$

$$a_{i+m}' = w^i \cdot (a_i - a_{i+m})$$



## Zeitkomplexität (2)

---

- ♦ Für die Erstellung des Vektors werden
  - ♦ m Additionen,
  - ♦ m Subtraktionen sowie
  - ♦  $2 \cdot m = n$  Multiplikationen benötigt (m für die Berechnung des Vektors, m für die Berechnung von  $w^i$ )
- ♦ Die beiden Ergebnisvektoren  $b'$  und  $b''$  müssen allerdings noch zu einem neuen Signal zusammengefügt werden – dies ist idealerweise in  $O(n) = \frac{3}{2}n$  möglich.



## Zeitkomplexität (3)

---

- ▶ Fasst man alle Faktoren zusammen ergibt sich für die CTFFT eine Zeitkomplexität von

$$O(n) = 3.5n + 2 \cdot O(n/2)$$

$$O(1) = 1$$

- ▶ Löst man die Rekursion auf, so ergibt sich

$$O(n) = 3.5n \cdot \log(n) = n \cdot \log(n)$$



# Inverse FFT

---

- Offen bleibt damit noch die Frage nach der Umkehrung der Fouriertransformation
  - Die Inverse FFT berechnet sich analog zur inversen DFT
  - Anstatt der  $n$ ten primitiven Einheitswurzel  $w$  wird dann die Inverse der Einheitswurzel  $w^{-1}$  verwendet
  - Die Inverse einer Einheitswurzel ist gerade die dazu konjugierte Zahl  $w' = w^{-1} = \frac{1}{w}$
  - Die Werte des Fouriersignals werden zuvor noch durch  $n$  dividiert
-



# Andere Anwendungsfälle

---

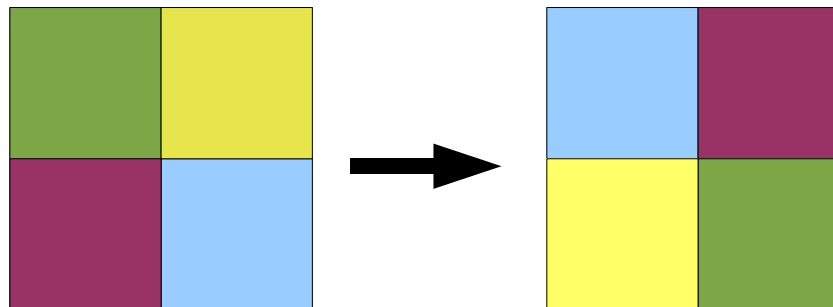
- ♦ Die Bildverarbeitung ist natürlich nicht der einzige Bereich, in dem der FFT eine große Bedeutung zukommt:
  - ♦ Polynommultiplikationen lassen sich mit Hilfe der FFT sehr schnell ausführen
  - ♦ Aufgrund der hohen Redundanz eines Signals im Frequenzraum lässt sich eine Kompression verwirklichen
  - ♦ In der Spracherkennung wird die FFT zur Bereinigung des Signals verwendet
  - ♦ Und vieles mehr...
-



# Anmerkungen

---

- ◆ Das Ergebnis der Fouriertransformation ist ein komplexes Signal – eine einfache Darstellung als Bild ist damit nicht möglich ==> Darstellung des Betrages der komplexen Zahlen
- ◆ Damit sich ein augenfreundliches Frequenzbild ergibt müssen die Sektoren des Bildes geschiftet werden:





# Anmerkungen

---

- ◆ Es existieren zahlreiche weitere FFT-Algorithmen wie das Radix-4-Verfahren oder Algorithmen auf Primzahlbasis. Diese zu erläutern würde aber den Rahmen dieser Arbeit sprengen
  - ◆ Da der Originalartikel für sich recht schwer zu entziffern ist: H.W. Lang's Algorithmen in Java bietet eine hervorragende Aufarbeitung des Artikels
  - ◆ Das Projekt fftw (fastest fourier-transform in the west) bietet eine enorm schnelle Implementation unter der GPL  
==> [www.fftw.org](http://www.fftw.org)
-



# Quellen

---

- ♦ J.W.Cooley & J.W. Tukey: An Algorithm for the Machine Calculation of Complex Fourier Series, Math. Computation, 1965 – Originalartikel zum Thema, ist im Anhang enthalten
  - ♦ A.V.Aho, J.E.Hopcroft, J.D.Ullman: *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974 – guter und verhältnismäßig menschenlesbarer Überblick über den Algorithmus
  - ♦ H.W.Lang: Algorithmen in Java, Oldenbourg, 2003 – exzellente Aufarbeitung des Originalartikels mit Pseudocode
-





## Quellen (2)

---

- ◆ T. Cormen et al.: Introduction to algorithms, 1990 – sehr knappe Zusammenfassung, aber interessant im Hinblick auf die Anwendungen zur Polynomberechnung
  - ◆ B. Jähne: Digitale Bildverarbeitung, 1997 – Ansatz zur Verwendung der Fouriertransformation zur Bildverbesserung, leider sehr schlechte und unvollständige Erklärung
  - ◆ W. Konen: Bildverarbeitung, Vorlesung SS 2005 – leicht verständliche Erläuterung zur Verwendung der FFT im Rahmen der Bildverarbeitung
-



## Quellen (3)

---

- ♦ Wikipedia, 2006 – Wiki-Artikel zur CTFFT, lesenswert  
[http://en.wikipedia.org/wiki/Cooley-Tukey\\_FFT\\_algorithm](http://en.wikipedia.org/wiki/Cooley-Tukey_FFT_algorithm)
  - ♦ Wikipedia, 2006 – eher auf Signalverarbeitung ausgerichteter Artikel zur FFT, für den Anwendungsfall Bildverarbeitung eher schlecht. [http://en.wikipedia.org/wiki/Fast\\_Fourier\\_transform](http://en.wikipedia.org/wiki/Fast_Fourier_transform)
  - ♦ Wikipedia, 2006 – ebenfalls auf die Signalverarbeitung gerichteter, aber schon deutlich umfassenderer Artikel zur diskreten Fouriertransformation.  
[http://en.wikipedia.org/wiki/Discrete\\_Fourier\\_transform](http://en.wikipedia.org/wiki/Discrete_Fourier_transform)
-

