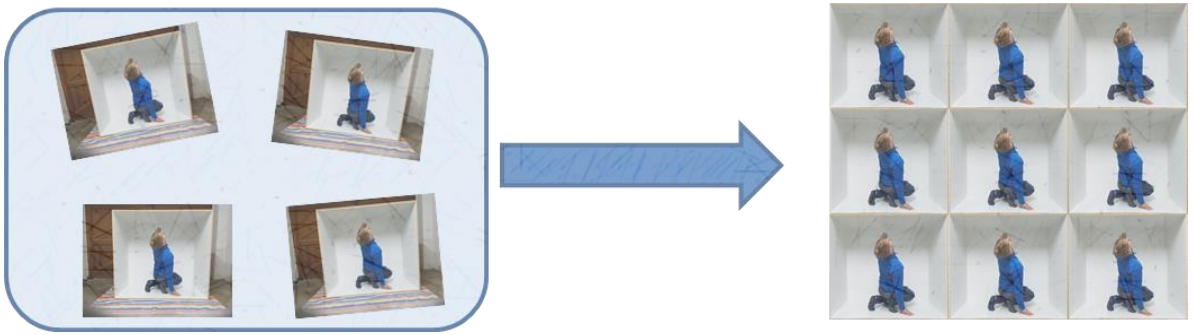
Digitale Bildverarbeitung 1

Einführung in die digitale Bilderverarbeitung
für Informatikstudierende im Bachelor

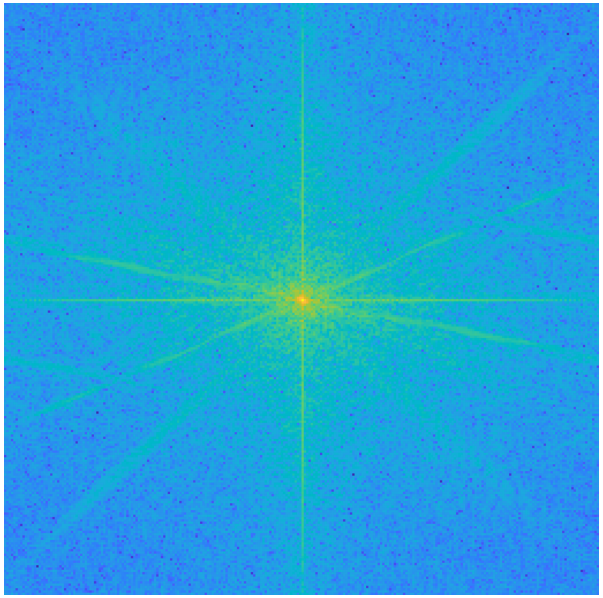
Vorlesung: Michael Möller – michael.moeller@uni-siegen.de
Übungen: Hannah Dröge – hannah.droege@uni-siegen.de



Fourier Transformation

Zuletzt ausführlich diskutiert: Fourier Transformation

Ortsdarstellung



Frequenzdarstellung
(Fourier Koeffizienten)

In 1d:
Basiert auf Darstellung $\vec{f} = \frac{1}{n} \sum_{k=0}^{n-1} c_k \vec{z}^{\,k}$ mittels $\vec{z}^{\,k} \in \mathbb{C}^n$ mit $z_l^k = \exp\left(2\pi i l \frac{k}{n}\right)$

Komplexe Exponentialfunktion zur Darstellung von reellen Bildern nicht intuitiv.

Diskrete Cosinustransformation

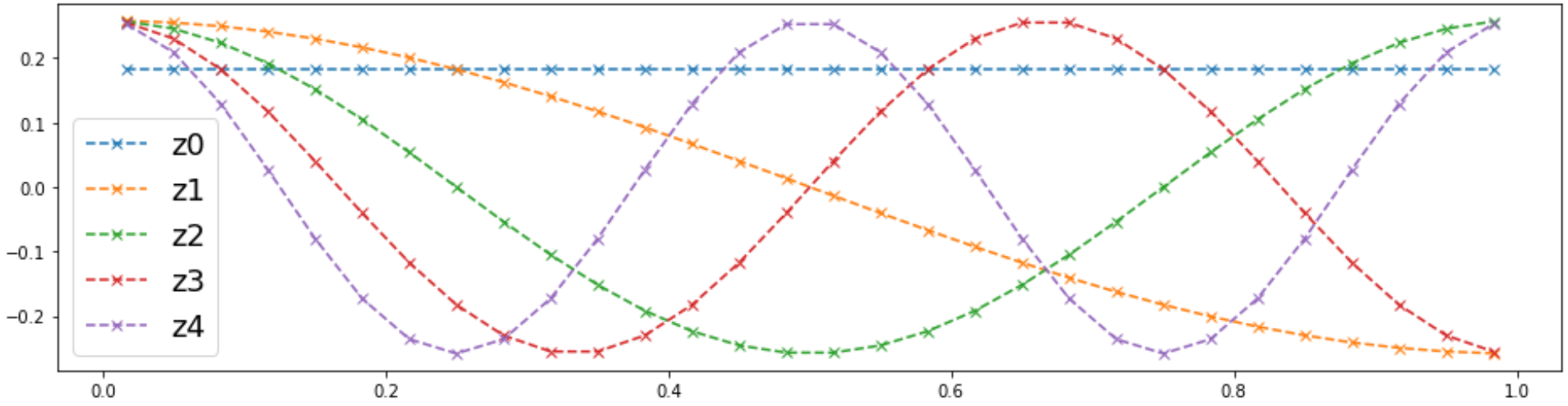
Andere Möglichkeit einer Art diskreten Frequenzanalyse:

$$\vec{f} = \sum_{k=0}^{n-1} c_k \vec{z}^{\,k}$$

$$\vec{z}^{\,k} \in \mathbb{R}^n \quad \text{mit} \quad z_l^0 = \frac{1}{\sqrt{n}} \quad \text{und} \quad z_l^k = \sqrt{\frac{2}{n}} \cos(\pi k x_l)$$

und einem etwas anderen diskreten Sampling:

$$x_l = \frac{(l + \frac{1}{2})}{n}, \quad l \in \{0, 1, \ldots, n-1\}$$



Diskrete Cosinustransformation

Oder eingesetzt / anders geschrieben:

$$f_l = \frac{1}{\sqrt{n}} c_0 + \sqrt{\frac{2}{n}} \sum_{k=1}^{n-1} c_k \cos\left(\pi k \frac{l+\frac{1}{2}}{n}\right)$$

Inverse DCT

Wobei man die Koeffizienten $c \in \mathbb{R}^n$ als *diskrete Cosinustransformation (DCT)* von $f \in \mathbb{R}^n$ bezeichnet.

Ähnlich zur Fourier Transformation kann man auch bei der DCT zeigen, dass die $\vec{z}^{\,k} \in \mathbb{R}^n$ eine Basis bilden. Für obige Normalisierung gilt sogar

$$\langle z^k, z^s \rangle = \begin{cases} 1 & \text{falls } k = s \\ 0 & \text{falls } k \neq s \end{cases}$$

Und es ergibt sich

$$c_k = \langle f, z^k \rangle = \begin{cases} \frac{1}{\sqrt{n}} \sum_{l=0}^{n-1} f_l & \text{falls } k = 0 \\ \sqrt{\frac{2}{n}} \sum_{l=0}^{n-1} f_l \cos\left(\pi k \frac{l+\frac{1}{2}}{n}\right) & \text{sonst.} \end{cases}$$

DCT

Diskrete Cosinustransformation

Gehen wir einfach mal direkt zum Code über!

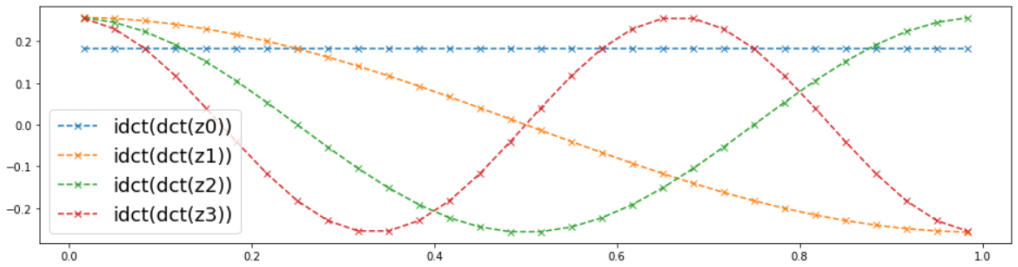
jupyter DCT Examples Last Checkpoint: letzten Donnerstag um 22:11 Uhr (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

Sanity check that the inverse dct indeed inverts the dct

In [4]:
```
plt.figure(figsize=(16, 4))
plt.plot(x,idct(dct(z0)),'x--')
plt.plot(x,idct(dct(z1)),'x--')
plt.plot(x,idct(dct(z2)),'x--')
plt.plot(x,idct(dct(z3)),'x--')
plt.legend(['idct(dct(z0))','idct(dct(z1))','idct(dct(z2))','idct(dct(z3))'], prop={'size': 18})
plt.show()
```



Diskrete Cosinustransformation

Um es auch auf den Folien festzuhalten:

Beispielnutzung der Diskreten Cosinustransformation (DCT) für die Bildkompression

Kodierung

Aufteilen in Patches (kleine Bildflächen), z.B. der Größe 8x8

Ausrechnen der DCT auf jedem Patch einzeln

Quantisieren der Koeffizienten (individuell für jeden DCT Koeffizienten)

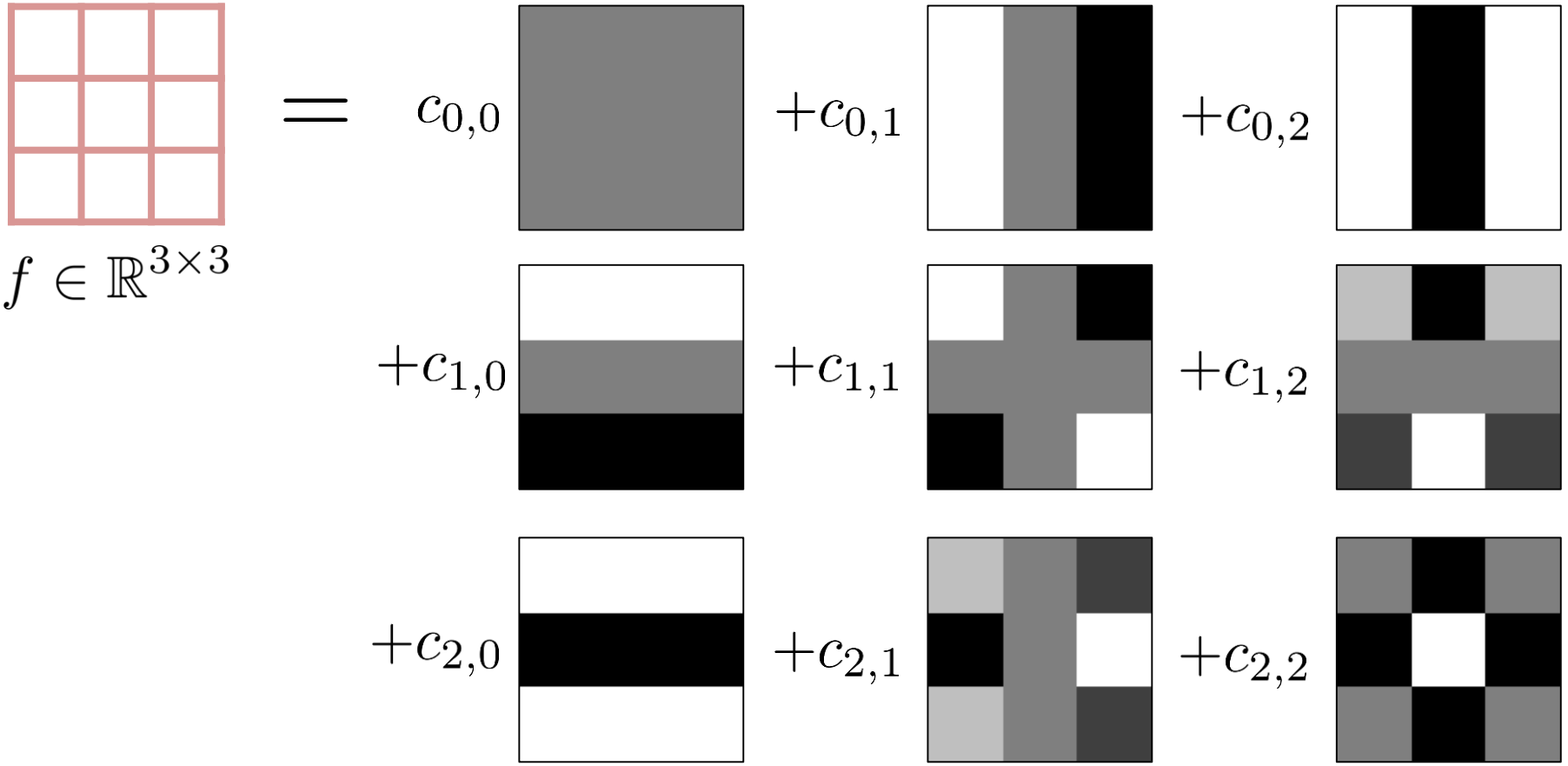
Effizientes Codieren der Koeffizienten (nicht besprochen, siehe Huffmann Coding)

Einfache Dekodierung

Inverse DCT mit quantisierten Koeffizienten (Mittelwert des zugehörigen Intervalls)

Diskrete Cosinustransformation

Interpretation der DCT auf einem Patch (hier für einen 3x3 Patch):



Diskrete Cosinustransformation

Vielfältige Anwendungen jenseits der Bildkompression!

Z.B. „**Entrauschen**“ von Bildern

Implementierung hier:

http://demo.ipol.im/demo/ys_dct_denoising/

- DCT auf 16x16 Patches
- Thresholding der Koeffizienten (setze die Koeffizienten die kleiner sind als ein Schwellwert auf 0)
- Inverse DCT
- Trick im Paper: Verwende keine disjunkten Patches, sondern überlappende. Den Bilde den Durchschnitt über alle Kandidaten.

Diskrete Cosinustransformation

Diskrete Cosinustransformation

Nichtlineare Patchweise Operationen

Die Idee ein Bild in Patches aufzuteilen und lokal zu Bearbeiten, scheint recht leistungsstark zu sein.

Diese Idee ist uns eigentlich auch schon bekannt: Faltungs- bzw. Korrelationsfilter arbeiten genau so!

Nächste Vorlesung: Einige weitere Beispiele für lokale (aber nicht-lineare) Filter. Mehr Details zum Thema Bildentrauschen!