Diskrete Kosinustransformation

Semesterarbeit Teil III

Fernfachhochschule Schweiz

Semesterarbeit

im Studiengang BSc Informatik

von

Arber Osmani

Bern, 18. November 2021

Eingereicht bei: Prof. Dr. Jörg Osterrieder

Inhaltsverzeichnis

1 Einleitung

Die diskrete Kosinustransformation (DCT) ist ein Verfahren aus der numerischen Mathematik für die Transformation von Signalen. Die DCT ist eine Sonderform der diskreten Fourier-Transformation (DFT), die aber im Gegensatz zu der DFT, nicht mit komplexen Funktionswerten arbeitet [1] und nur Kosinus-Terme verwendet. Die DCT eignet sich besonders gut für die Kompression von Bilddaten. Die Grundidee besteht darin, Signale aus dem Zeitbereich in den Frequenzbereich zu transformieren.

Diese Semesterarbeit beschäftigt sich mit der diskreten Kosinustransformation. In den nachfolgenden Kapitel werden die theoretischen Grundlagen behandelt, mittels derer wir Schrittweise die eindimensionale- und zweidimensionale Kosinustransformation definieren wollen. Weiter sehen wir einige Anwendungsfälle aus der digitalen Bildverarbeitung an. Im letzten Kapitel wird der Python-Code, für die Erzeugung einer eindimensionalen Transformationsmatrix vorgestellt.

Diese Semesterarbeit, der Python-Code und weitere Ressourcen, stehen online unter der nachfolgenden Adresse zur Verfügung.

`https://linalg.arberosmani.ch`

2 Fourier-Transformation

Fourier[1] hat erkannt, dass jede periodische Funktion als Summe von (unendlich vieler) Sinus- und Kosinusfunktionen dargestellt werden kann [2]. Anders ausgedrückt: Eine periodische Funktion $f(t)$ mit Periode p lässt sich als Überlagerung unendlich vieler Sinus- und Kosinusschwingungen darstellen[2]. Diese Überlagerung wird Fourier-Reihe genannt. Die Fourier-Transformation ist also ein Näherungsverfahren für periodische Funktionen. Die Funktion $f_f(t)$ ist dann die entwickelte Funktion, die die Funktion $f(t)$ annähert (2.1).

$$f_f(t) = \sum_{n=0}^{\infty} [a_n \cdot \cos(n\omega t) + b_n \cdot \sin(n\omega t)] \tag{2.1}$$

[1] Joseph Fourier (geb. 1768) war ein französischer Mathematiker und Physiker.

[2] Als Schwingungen werden repetitive Schwankungen bezeichnet, die typischerweise in Abhängigkeit von Zeit und bei Messung eines Wertes auftreten.

Eine Funktion f ist periodisch mit Periode p wenn

$$f(t) = f(t + np) \tag{2.2}$$

gilt [4]. Beispiele für periodische Funktionen sind Sinus und Kosinus als 2π-periodische Funktionen. Diese Transformation funktioniert nicht nur in der oben beschriebenen Richtung, sondern auch umgekehrt. Die Fourier-Transformation ist reversibel.

Im Beispiel aus Abbildung 2.1 sind zwei Diagramme dargestellt. Im unteren Teil der Grafik sind drei unterschiedliche Schwingungen zu erkennen. Der obere Teil der Grafik zeigt die Schwingungen in überlagerter Form (2.3).

$$\sin(50 \cdot 2t\pi) + 2 \cdot \sin(100 \cdot 2t\pi) + 3 \cdot \cos(10 \cdot 2t\pi) \tag{2.3}$$

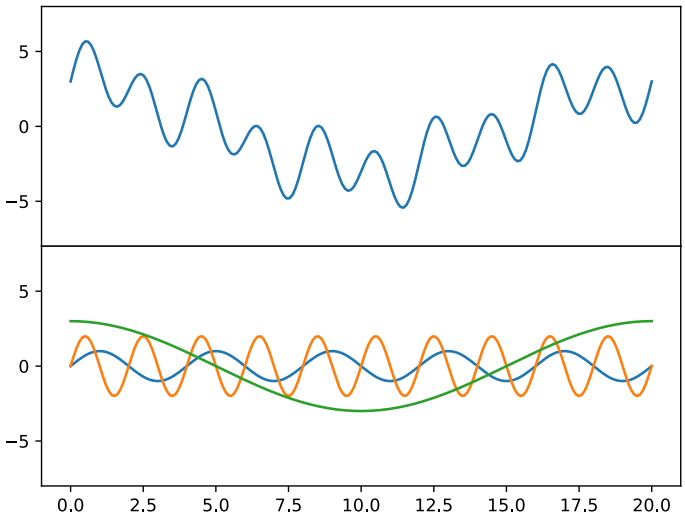

Abbildung 2.1: Periodische Funktionen überlagert und einzeln dargestellt

Abbildung 2.2 zeigt die Funktion (2.3) transformiert aus dem Zeit- in den Frequenzbereich.

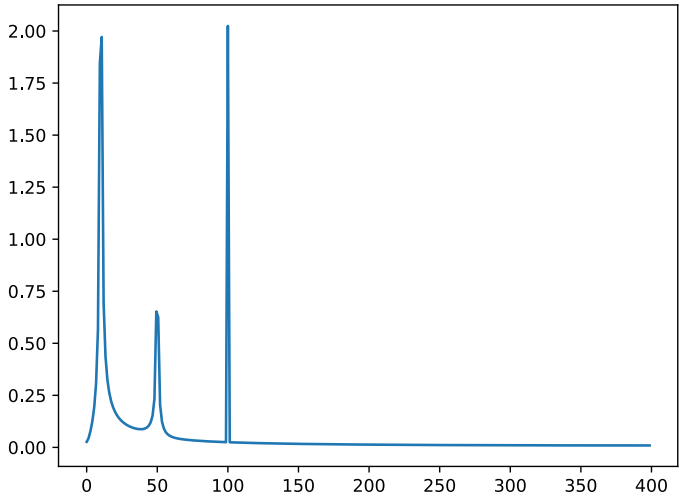

Abbildung 2.2: Periodische Funktion aus (2.3) im Frequenzbereich (ermittelte Frequenzen 10, 50 und 100)

2.1 Diskrete Fourier-Transformation

Die diskrete Fourier-Transformation (2.4) und so auch die diskrete Kosinustransformation, sind lineare Abbildungen. Das Eingangssignal, beispielsweise ein Bild oder ein Audio, kann als Vektor betrachtet werden das mit einer Matrix, die aus (2.4) hergeleitet werden kann, multipliziert wird.

$$F(u) = \sum_{x=0}^{N-1} f(x) \left[\cos\left(\frac{2\pi xu}{N}\right) + \mathrm{i}\sin\left(\frac{2pixu}{N}\right) \right] \tag{2.4}$$

Das Resultat $F(u)$ entspricht dem Koeffizienten zur Frequenz u.

Aus (2.4) erkenne wir, dass die diskrete Fourier-Transformation, komplexwertige Werte in komplexwertige Werte transformiert. Das Eingangssignal kann auch aus reellwertigen Werten sein, das Resultat entspricht dennoch komplexwertigen Werten. Diese Transformation ist für die Signalübertragung oder Signalspeicherung ungeeignet. Im nachfolgenden Kapitel betrachten wir die diskrete Kosinustransformation.

3 Diskrete Kosinustransformation

Wir haben gesehen, dass die diskrete Fourier-Transformation einen Imaginärteil mit sich bringt. Bei einem reellwertigen Eingangssignal, benötigt die Ausgabe doppelt soviel Speicher. Im nächs-

ten Unterkapitel probieren wir, zunächst durch Vereinfachung, die diskrete Kosinustransformation herzuleiten.

3.1 Herleitung der diskreten Kosinustransformation

Als erstes lassen wir den Imaginärteil weg (3.1) und fragen uns, ob die resultierenden Basisvektoren, in jedem Fall linear unabhängig sind.

$$F'(u) = \sum_{x=0}^{N-1} f(x) \left[\cos\left(\frac{2\pi x u}{N}\right) \right] \tag{3.1}$$

Wir probieren das einfach mal mit einem Signal $N = 1$ und setzen dann weiter fort ($N = 2$, $N = 3$ usw.) um zu sehen ob die Basisvektoren der Matrix linear unabhängig sind.

Für den einfachsten Fall, mit nur einem Basisvektoren, funktioniert es offensichtlich. Ein Vektor alleine im eindimensionalen Raum ist immer linear unabhängig.

$$F'(0) = f(0) \cdot (1)$$

Als nächstens probieren wir ein Signal mit zwei Werten ($N = 2$). Die resultieren Basisvektoren sind auch für diesen Fall linear unabhängig und die Matrix ist invertierbar.

$$\begin{aligned}\begin{pmatrix} F'(0) \\ F'(1) \end{pmatrix} &= \begin{pmatrix} 1 & 1 \\ \cos\left(\frac{2\pi 0}{2}\right) & \cos\left(\frac{2\pi 1}{2}\right) \end{pmatrix} \begin{pmatrix} f(0) \\ f(1) \end{pmatrix} \\ &= \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} f(0) \\ f(1) \end{pmatrix}\end{aligned}$$

Bei $N = 3$ funktioniert das nicht mehr. Wir sehen dass die Basisvektoren der Matrix nicht linear unabhängig sind. Wir könnten keine inverse Matrix angeben.

$$\begin{aligned}\begin{pmatrix} F'(0) \\ F'(1) \\ F'(2) \end{pmatrix} &= \begin{pmatrix} 1 & 1 & 1 \\ \cos\left(\frac{2\pi 0}{3}\right) & \cos\left(\frac{2\pi 1}{3}\right) & \cos\left(\frac{2\pi 2}{3}\right) \\ \cos\left(\frac{2\pi 0}{3}\right) & \cos\left(\frac{2\pi 2}{3}\right) & \cos\left(\frac{2\pi 4}{3}\right) \end{pmatrix} \begin{pmatrix} f(0) \\ f(1) \\ f(2) \end{pmatrix} \\ &= \begin{pmatrix} 1 & 1 & 1 \\ 1 & -1/2 & -1/2 \\ 1 & -1/2 & -1/2 \end{pmatrix} \begin{pmatrix} f(0) \\ f(1) \\ f(2) \end{pmatrix}\end{aligned}$$

Daraus schliessen wir, dass das weglassen des Imaginärteil der FT alleine nicht genügt. Der Trick besteht darin, den Realteil der FT so zu verändern, dass die Symmetrie gebrochen wird (3.2).

Dafür wird der Faktor 2 entfernt und 0.5 zu x addiert [3]. Die Formel ist so gestaltet, dass die Basisvektoren orthogonal zueinander stehen.

$$F^{DCT}(u) = \sum_{x=0}^{N-1} f(x) \left[\cos\left(\frac{\pi u(x + 1/2)}{N} \right) \right] \tag{3.2}$$

Die Abbildung 3.1 visualisiert die Basisvektoren. Im oberen Teil ist zu erkennen, dass die Basisvektoren nicht linear unabhängig sind, da die Punkte an den Stellen $x = 1$ und $x = 2$ nicht unterschieden werden können. Der untere Teil, zeigt die Basisvektoren gemäss (3.2).

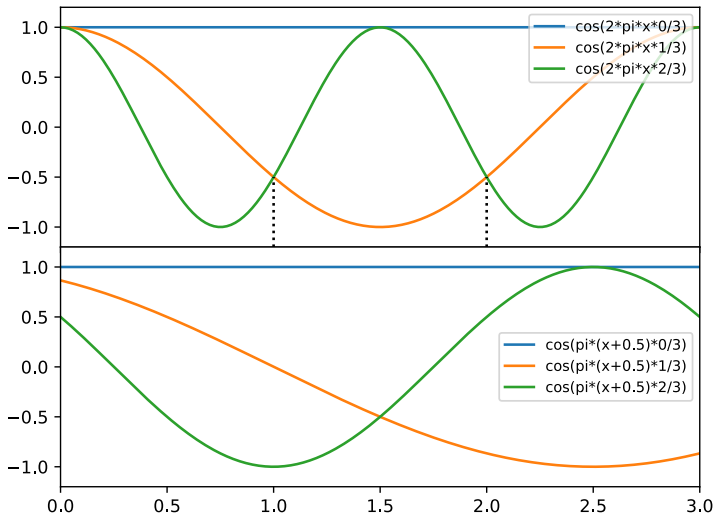

Abbildung 3.1: Basisvektoren gemäss (3.1) und (3.2)

3.2 DCT im zweidimensionalen

Im vorigen Abschnitt haben wir die DCT im eindimensionalen betrachtet. Die DCT kann aber auch im mehrdimensionalen angewendet werden. Dazu wird die DCT spalten- oder Zeilenweise angewendet. Die Formel für die Ausführung im 2D sieht wie folgt aus:

$$F^{DCT}(u, v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \left[\cos\left(\frac{\pi u(x + 1/2)}{N} \right) \cos\left(\frac{\pi v(y + 1/2)}{N} \right) \right] \tag{3.3}$$

3.2.1 Anwendungen im 2D: JPEG

Die 2D-DCT wird hauptsächlich für die Kompression von Bildern im JPEG eingesetzt. Die DCT an sich ist kein Kompressionsverfahren. Die Pixel eines Bildes werden aber in Ortsfrequenzen

transformiert. Tiefe Ortsfrequenzen meinen grobe Strukturen und geringe Unterschiede zwischen benachbarten Pixeln. Hohe Ortsfrequenzen meinen feine Strukturen, harte Kanten und grosse Unterschiede zwischen benachbarten Pixeln. Anhand dieser Informationen, kann bei geeigneten Bildern viel Information weggelassen werden, so dass für das menschliche Auge, keine Unterschiede auszumachen sind.

An der Formel (3.3) lässt sich erkennen, dass der Algorithmus eine Zeitkomplexität von $O(n^2)$ hat. Das ist für Bilder, beispielsweise mit einer Grösse von 1000×1000 Pixel, ungeeignet. In der Praxis werden für die JPEG Kompression daher die Bilder in 8×8-Blöcke zerlegt. Die Abbildung 3.2 visualisiert die 64 Basisschwingungen für die 2D-DCT. Diese stellen gleichsam die Koeffizienten dar. Durch gewichten der Basisvektoren mit ihren Koeffizienten und Addieren, kann jedes 8×8 Pixelbild generiert werden [3].

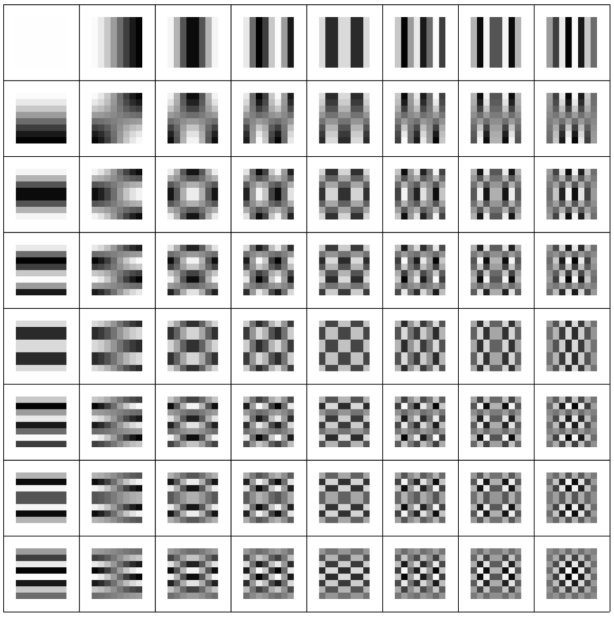

Abbildung 3.2: 64 Basisschwindungen der 2D-DCT

4 Python-Code

4.1 Implementierungsidee

Die Implementierung der Funktion, für die Generierung der DCT Matrix, richtet sich vollständig nach der Formel (3.2) aus Abschnitt 3.1. Für die Berechnung der Inverse (DCT-III) wird die bereits entwickelte Klassenfunktion `inv()` der Klasse `linalg.Matrix` verwendet. Die Transformation ist eine Matrix-Vektor Multiplikation. Diese Art Multiplikation ist ebenfalls bereits in der Klasse `lingalg.Matrix`, durch überladen der Methode `__mul__()` implementiert.

4.2 Programmcode

Die Funktion für die eindimensionale DCT (3.2) kann sehr einfach programmiert werden. Nachfolgend werden die Python Funktionen aus dem Package `sema.teil3.dct` vorgestellt und ihre Implementation.

4.3 Package `sema.teil3.dct`

Das Package `sema.teil3.dct` enthält die geforderten Funktionen.

* `dct2Matrix(N: int) -> linalg.Matrix`
* `dct3Matrix(N: int) -> linalg.Matrix`
* `dct2(v: ARRAY_LIKE[1]) -> linalg.Vector`
* `dct3(v: ARRAY_LIKE) -> linalg.Vector`
* `dct2_2d(m: MATRIX_LIKE[2]) -> np.ndarray`
* `dct3_2d(m: MATRIX_LIKE) -> np.ndarray`
* `dct2_blockWise(filename: str, threshold=0.01) -> np.ndarray`
* `plot_dct2_image(filename: str, threshold=0.01, destination=None) -> None`

4.3.1 Funktion `sema.teil3.dct.dct2Matrix`

Gibt die DCT-II Matrix der Grösse `N`×`N` für ein gegebenes `N`.

```
def dct2Matrix(N: int) -> Matrix:
    M = identity(N)
    for k in range(0, N):
        for n in range(0, N):
            M[k][n] = np.cos((np.pi * (n + 0.5) * k) / N)
    return M
```

4.3.2 Funktion `sema.teil3.dct.dct2Matrix`

Gibt die DCT-III Matrix der Grösse `N`×`N` für ein gegebenes `N`. Die DCT-III Matrix ist die Umkehrung der DCT-II Matrix[3].

[1] `ARRAY_LIKE = typing.Union[typing.List[float], np.ndarray, linalg.Vector]`
[2] `MATRIX_LIKE = typing.Union[typing.List[ARRAY_LIKE], np.matrix, Matrix])` -> linalg.Vector
[3] `https://de.wikipedia.org/wiki/Diskrete_Kosinustransformation#DCT-III`

```
def dct3Matrix(N: int) -> Matrix:
    return dct2Matrix(N).inv()
```

4.3.3 Funktion `sema.teil3.dct.dct2`

Wendet die diskrete Kosinustransformation (DCT-II) auf einen gegebenen Vektor oder ein Array-ähnliches Objekt `v` an. Zurückgegeben wird ein transformierter Vektor.

```
def dct2(v: ARRAY_LIKE) -> Vector:
    return dct2Matrix(len(v)) * Vector(*v)
```

4.3.4 Funktion `sema.teil3.dct.dct2`

Wendet die inverse diskrete Kosinustransformation (DCT-III) auf einen gegebenen Vektor oder ein Array-ähnliches Objekt `v` an. Zurückgegeben wird ein transformierter Vektor.

```
def dct3(v: ARRAY_LIKE) -> Vector:
    return dct3Matrix(len(v)) * Vector(*v)
```

4.3.5 Funktion `sema.teil3.dct.plot_dct2_image`

Plotet das angegebene Bild im original und mittels DCT komprimiert. Für die Kompression kann ein Schwellwert angegeben werden, um anzugeben, ab welcher Schwelle die DCT Koeffizienten behalten werden sollen. Je höher die Schwelle, um so höher die Kompression und Informationsverlust. Optional kann auch eine `destination` angegeben werden, als Zielname um das Bild abzuspeichern.

Die Funktion verwendet `sema.teil3.dct.dct2_blockWise` was wiederum `dct2_2d` und `dct3_2d` aus dem selben Package verwendet. Letztere zwei setzen direkt `scipy.fftpack.dct` ein.

4.4 Bilder mit `scipy.fftpack.dct` transformieren

Für die Transformation der Bilder wird im Hintergrund `scipy.fftpack.dct` verwendet. Der Schwellwert (`threshold`) ist entscheidend dafür, welche DCT Koeffizienten verwendet werden. Je höher der Schwellwert, umso höher die Kompressionsrate, da es wahrscheinlicher ist, dass Koeffizienten weggelassen werden.

Wenn das Bild grobe Strukturen aufweist, hat es eine tiefe Ortsfrequenz. Das ist dann der Fall, wenn typische Bilder von Orten, Landschaften oder Gegenständen aufgenommen werden. Die benachbarten Pixel haben nur geringe Unterschiede zu einander. Es kann eine gute Kompressionsrate erzielt werden, ohne dass vom menschlichen Auge, ein besonderer Unterschied wahrgenommen werden kann. Hohe Ortsfrequenzen hingegen entsprechen feinen Strukturen. Das sind z.B. harte Kanten oder Strukturen bei denen die Farbwerte sehr schnell ändern. Nachfolgend ein paar Beispiele.

Abbildung 4.1: 6.1829% der DCT Koeffizienten beibehalten

Abbildung 4.2: 1.8815% der DCT Koeffizienten beibehalten

Abbildung 4.3: 1.6429% der DCT Koeffizienten beibehalten

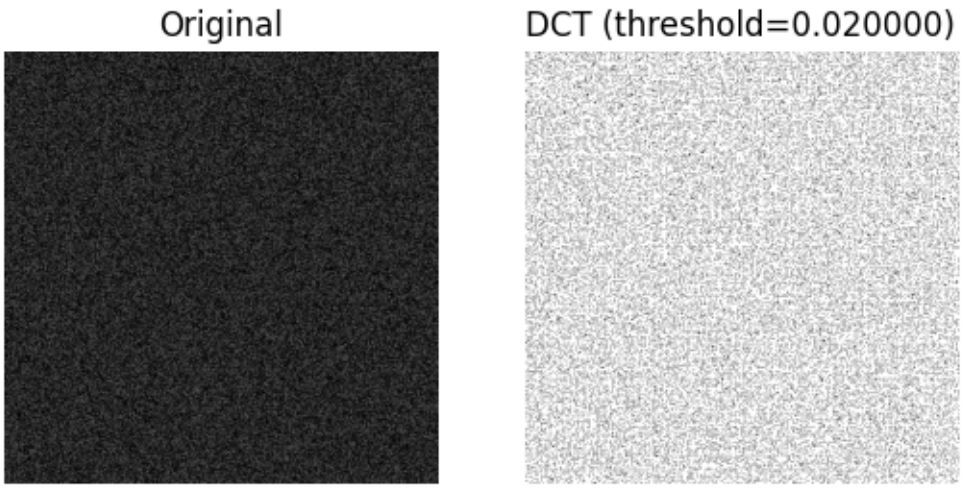

Abbildung 4.4: 84.4992% der DCT Koeffizienten beibehalten

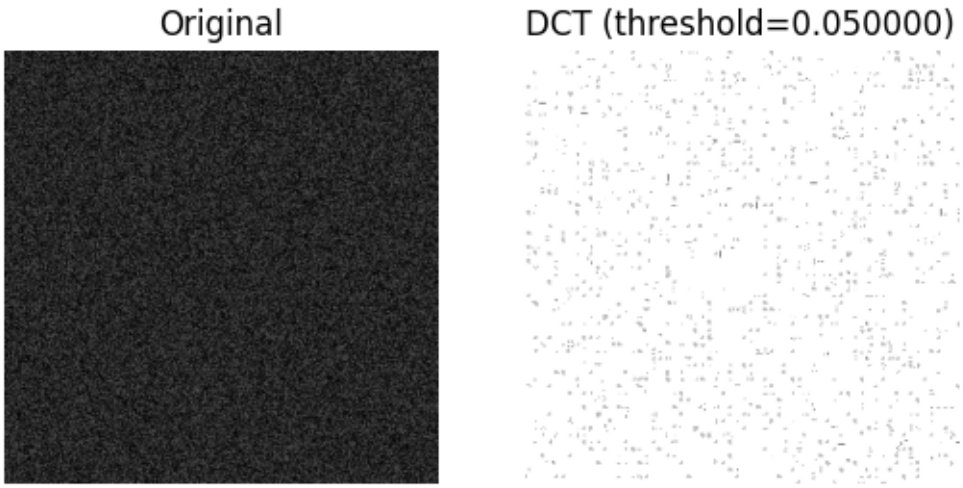

Abbildung 4.5: 12.0072% der DCT Koeffizienten beibehalten

Das Bild vom Kampfflieger hat eher grobe Strukturen und weist eine gute Kompressionsrate auf. In Abbildung 4.1 ist zwischen dem original (links) und der komprimierten Version (rechts) kaum ein Unterschied zu erkennen. Dabei werden nur 6.1829% der DCT Koeffizienten beibehalten bei einem Schwellwert von 0.005, was zu einer guten Kompressionsrate führt. Das erhöhten des Schwellwertes bringt nicht mehr sehr viel und die Qualität nimmt stark ab. Das zweite Bild ist ein zufällig generiertes Bild[4] mit hoher Ortsfrequenz. Das Bild kann praktisch nicht komprimiert werden.

Die Beispiele sind via Google Colab zugänglich[5].

[4] `http://kitfox.com/projects/perlinNoiseMaker`
[5] `https://linalg.arberosmani.ch`

Literaturverzeichnis

[1] Wilhelm Burger and Mark James Burge. *Diskrete Kosinustransformation (DCT)*, pages 535–544. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. ISBN 978-3-642-04604-9. doi: 10.1007/978-3-642-04604-9_20. URL https://doi.org/10.1007/978-3-642-04604-9_20.

[2] Tilman Butz. *Fouriertransformation für Fußgänger*. Vieweg & Teubner, Wiesbaden, Germany, 4 edition, October 2005. doi: 10.1007/978-3-663-10086-7. URL https://doi.org/10.1007/978-3-663-10086-7.

[3] Prof. Dr. Haenselmann, Thomas. Bildverarbeitung, basistransformationen. URL https://www.cb.hs-mittweida.de/fileadmin/verzeichnisfreigaben/haenselm/dokumente/dbv_2016ss_transform.pdf.

[4] Eric W Weisstein. Periodic function. URL https://mathworld.wolfram.com/PeriodicFunction.html.