JPEG Bildkompression mit Schwerpunkt auf die Diskrete Cosinus Transformation

Martina Tscheckl

14. März 2015

Inhaltsverzeichnis

1 Einleitung

Kompression von Bildern ist heutzutage nicht mehr wegzudenken. Das gilt besonders für unsere digitale Welt und das Streben möglichst viele Informationen in möglichst kurzer Zeit zu erhalten. Will man sich zum Beispiel ein Bild aus dem Internet ansehen, möchte man nicht Minuten darauf warten, dass das Bild geladen wird. Bereits mehrere Sekunden zu warten, erscheint wie eine Ewigkeit und verringert das Interesse das Bild überhaupt sehen zu wollen.

Diese Arbeit beschäftigt sich mit der Kompression solcher digitalen Bilder. Der Fokus liegt auf der JPEG Bildkompression für schwarz-weiß Bilder.

JPEG steht für Joint Photographic Experts Group. Diese Gruppe hat einen Algorithmus zur Kompression von natürlichen Grauton- und Farbbildern entwickelt.
Ziel bei dieser Entwicklung war eine hohe Kompressionsrate und hohe Codierungs-/ Decodierungsgeschwindigkeit, wobei die Bilddaten so reduziert werden, dass für das menschliche Auge wenig bis kein Qualitätsverlust erkennbar ist.

Man unterscheidet grundsätzlich zwischen zwei Kompressionsmethoden, die verlustfreie und die verlustbehaftete Kompression.

Bei der verlustfreien Kompression kommt es zu keinem Verlust der Bildinformation, auch wenn die Bilddaten reduziert werden. Diese Methode wird vor allem in der Medizin und Analyse von Satellitenbildern verwendet, da dort ein Informationsverlust gravierende Auswirkungen auf die Ergebnisse haben kann. Die so komprimierten Bilder haben hohe Qualität, sind aber auch sehr speicheraufwändig.

Bei der verlustbehafteten Kompression gehen zugunsten geringeren Speicherplatzes Bildinformationen tatsächlich verloren. Diese Bilder können nicht mehr eins-zu-eins wiederhergestellt werden und es können Artefakte auftreten. Diese Methode findet vor allem im Internet Anwendung. Man will große Daten - wie zum Beispiel Bilder - laden, aber nicht zu lange auf die Daten warten.

Dieses Dokument befasst sich ausschließlich mit der verlustbehafteten Kompression.

Abbildung 1: Das Bild vor und nach der Kompression

2 Überblick

Der Aufbau der JPEG Codierung und Decodierung sieht folgendermaßen aus:

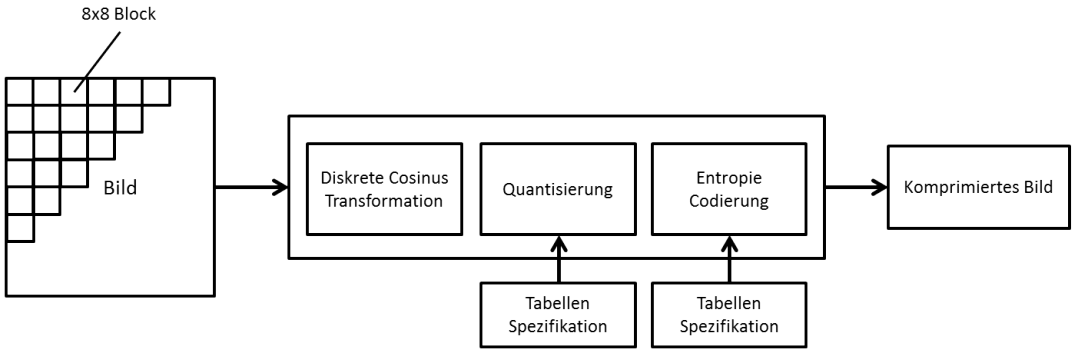


Abbildung 2: Schritte der Codierung

Zuerst wird das Bild in Blöcke von 8×8 Pixeln zerlegt. Anschließend wird auf jeden 8×8 Block die Diskrete Cosinus Transformation angewandt. Diese ist noch verlustfrei. Beim nächsten Schritt - der Quantisierung - gehen dann Bildinformationen verloren. Nach der Quantisierung werden die Wert noch umgeordnet, um bei der Codierung Speicher sparen zu können. Schließlich werden die so gewonnenen Informationen mit einem Entropie-Codierer binär codiert.

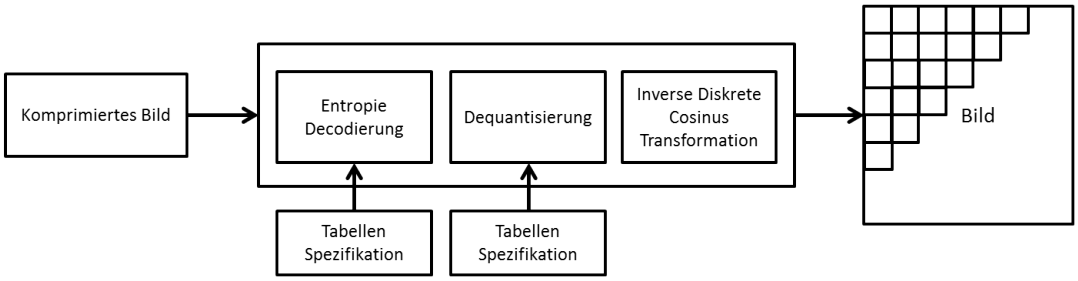


Abbildung 3: Decodierung

Der Schwerpunkt dieser Arbeit liegt auf der Diskreten Cosinus Transformation.

Als Erstes wird gezeigt, dass die Transformationsmatrix der Diskreten Cosinus Transformation unitär ist, woraus folgt, dass ihre Inverse einfach zu bestimmen ist. Außerdem werden die Bildpunkte in Frequenzüberlagerungen umgewandelt, was das Trennen von notwendiger von irrelevanter Information erleichtert, sowie eine günstigere Darstellung zur Kompression liefert. Anschließend wird der Zusammenhang zwischen der Diskreten Cosinus Transformation und der Diskreten Fourier Transformation aufgezeigt. Da die Fourier Transformation viel bekannter als die Cosinus Transformation ist, gibt zu dieser bereits gute Algorithmen um sie effizient berechnen zu können. Aus diesem Grund wird auch ein schneller Algorithmus zur Lösung der Fourier Transformation präsentiert und mit der trivialen Berechnung der Fourier Transformation

verglichen. Danach wird eine Definition für die 2D-DCT angegeben und auf die 1D-DCT reduziert, um die Berechnung der Wert noch effektiver zu gestalten. Als nächstes werden die Quantisierung und die Aufbereitung der Werte für die Entropie-Codierung beschrieben. Gefolgt von der Entropie-Codierung selbst, wobei im Speziellen auf die Huffman-Codierung mit einem Beispiel eingegangen wird.

Schließlich wird die Implementierung des Projekts in MATLAB erklärt.

3 Diskrete Cosinus Transformation (DCT)

Die Diskrete Cosinus Transformation stellt Signale - zum Beispiel Bildpunkte - durch Überlagerungen von Cosinuswellen mit verschiedenen Frequenzen und Amplituden dar. Dabei werden unterschiedliche Frequenzen des Signals sichtbar. Dazu wird zunächst die DCT für 1D-Signale definiert.

Bemerkung. *Es gibt vier Diskrete Cosinus Transformationen, die in der Datenkompression eine wesentliche Rolle spielen. Sie werden mit DCT-I bis DCT-IV bezeichnet. Für die hier beschriebene JPEG-Bildkompression werden die DCT-II und DCT-III verwendet.*

Definition (Diskrete Cosinus Transformation)**.** *Sei x_m der Zeilenvektor der zu transformierenden Signale und N die Länge dieses Vektors. Mit $x_m(n)$ wird das n-te Element des Zeilenvektors bezeichnet, wobei die Elemente des Vektors von 0 bis $N-1$ positioniert sind.*

DCT-II:
$$X(m) = \left(\frac{2}{N}\right)^{\frac{1}{2}} k_m \sum_{n=0}^{N-1} x_m(n) \cos\left(\frac{m\left(n+\frac{1}{2}\right)\pi}{N}\right)$$

DCT-III:
$$X(m) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \sum_{n=0}^{N-1} k_n x_m(n) \cos\left(\frac{(m+\frac{1}{2})n\pi}{N}\right)$$

mit $k_m = k_n = \begin{cases} \frac{1}{\sqrt{2}} & n = 0 \\ 1 & sonst \end{cases}$

Folgerung (DCT Transformationsmatrix)**.** *Aus der Definition der DCT können die Einträge der Transformationsmatrix c_N hergeleitet werden. Hierbei handelt es sich um eine quadratische $N \times N$-Matrix. Mit $[c_N]_{mn}$ wird der Eintrag der m-ten Zeile und n-ten Spalte bezeichnet, wobei m und n die Indizes von 0 bis $N-1$ der Matrix sind.*

DCT-II:
$$\left[c_N^{II}\right]_{mn} = \left(\frac{2}{N}\right)^{1/2} \left(k_m \cos\left(\frac{m\left(n+\frac{1}{2}\right)\pi}{N}\right)\right)$$

DCT-III:
$$\left[c_N^{III}\right]_{mn} = \left(\frac{2}{N}\right)^{1/2} \left(k_n \cos\left(\frac{\left(m+\frac{1}{2}\right)n\pi}{N}\right)\right)$$

mit $k_m = k_n = \begin{cases} \frac{1}{\sqrt{2}} & n = 0 \\ 1 & sonst \end{cases}$

Definition. *Seien g_k und g_m zwei N-dimensionale reelle Vektoren. Dann ist das innere Produkt definiert durch*

$$\langle g_k, g_m \rangle = \sum_{n=0}^{N-1} g_{nk} g_{nm} \text{ mit } g_k = [g_{0k}, g_{2k}, \ldots, g_{(N-1)k}]^T$$

Bemerkung. *Eine Matrix heißt orthogonal, wenn $\langle g_k, g_m \rangle = 0$ für $k \neq m$, mit k und m als Indizes der Matrix (0 bis $N-1$).*

Gilt des weiteren $\langle g_k, g_k \rangle = 1$, so nennt man die Matrix unitär.

Im Folgenden soll gezeigt werden, dass die DCT-II unitär ist. Dazu ein paar Bemerkungen, die die Beweisführung erleichtern.

Bemerkung. *Mit den Additionstheoremen lässt sich folgende Formel herleiten:*

$$\cos(x) \cdot \cos(y) = \frac{1}{2}\left(\cos(x+y) + \cos(x-y)\right)$$

Bemerkung.

$$\sum_{n=1}^{N-1} z^n = z + z^2 + \cdots + z^{N-1} \cdot \frac{1-z}{1-z} = \frac{z - z^N}{1-z}$$

Bemerkung. *Aufgrund der Symmetrie des Cosinus gilt*

$$\cos\left(p\pi - \frac{p\pi}{N}\right) = \cos\left(p\pi\right) \cdot \cos\left(\frac{p\pi}{N}\right)$$

Satz. *Die Diskrete Cosinus Transformation ist unitär.*

Beweis. Sei g_k der k-te Spaltenvektor der NxN-Transformationsmatrix $\left[c_N^{II}\right]$.

Und weiters sei $a_n = \begin{cases} \frac{1}{\sqrt{2}} & n = 0 \\ 1 & sonst \end{cases}$

Dann gilt:

$$\begin{aligned}
\langle g_k, g_m \rangle &= \sum_{n=0}^{N-1} \left(\frac{2}{N}\right)^{\frac{1}{2}} a_n \cos\left(\frac{n(k+\frac{1}{2})\pi}{N}\right) \left(\frac{2}{N}\right)^{\frac{1}{2}} a_n \cos\left(\frac{n(m+\frac{1}{2})\pi}{N}\right) \\
&= \frac{2}{N} \sum_{n=0}^{N-1} a_n^2 \cos\left(\frac{n(k+\frac{1}{2})\pi}{N}\right) \cos\left(\frac{n(m+\frac{1}{2})\pi}{N}\right)
\end{aligned}$$

[Werte für a_n einsetzen]

$$\begin{aligned}
&= \frac{2}{N}\left(\frac{1}{2} + \sum_{n=1}^{N-1} \cos\left(\frac{n(k+\frac{1}{2})\pi}{N}\right) \cos\left(\frac{n(m+\frac{1}{2})\pi}{N}\right)\right) \\
&= \frac{2}{N}\left(\frac{1}{2} + \sum_{n=1}^{N-1} \frac{1}{2}\left(\cos\left(\frac{n(k+m+1)\pi}{N}\right) + \cos\left(\frac{n(k-m)\pi}{N}\right)\right)\right) \\
&= \frac{1}{N}\left(1 + \sum_{n=1}^{N-1} \cos\left(\frac{n(k+m+1)\pi}{N}\right) + \sum_{n=1}^{N-1} \cos\left(\frac{n(k-m)\pi}{N}\right)\right)
\end{aligned}$$

Wähle W_{2N} als $2N$-te Primitivwurzel von 1 mit $W_{2N} = \exp(\frac{-i\pi}{N}) = \cos(\frac{\pi}{N}) - i\sin(\frac{\pi}{N})$.

Betrachte nun für $p = k + m + 1$ bzw $p = k - m$ mit $k \neq m$

$$\begin{aligned}
\sum_{n=1}^{N-1} \left(W_{2N}^{-p}\right)^n &= \frac{W_{2N}^{-p} - (W_{2N}^{-p})^N}{1 - W_{2N}^{-p}} \cdot \frac{1 - W_{2N}^{p}}{1 - W_{2N}^{p}} \\
&= \frac{W_{2N}^{-p} - W_{2N}^{-Np} - 1 + W_{2N}^{-(N-1)p}}{2\left(1 - \cos\left(\frac{p\pi}{N}\right)\right)}
\end{aligned}$$

$$\begin{aligned}
\Re\left[\sum_{n=1}^{N-1}\left(W_{2N}^{-p}\right)^n\right] &= \frac{\cos\left(\frac{p\pi}{N}\right)-\cos\left(p\pi\right)-1+\cos\left(p\pi-\frac{p\pi}{N}\right)}{2\left(1-\cos\left(\frac{p\pi}{N}\right)\right)} \\
&= \frac{\cos\left(\frac{p\pi}{N}\right)-1+\cos\left(p\pi\right)\cos\left(\frac{p\pi}{N}\right)-\cos\left(p\pi\right)}{2\left(1-\cos\left(\frac{p\pi}{N}\right)\right)} \\
&= \frac{\cos\left(\frac{p\pi}{N}\right)-1+\cos\left(p\pi\right)\left(\cos\left(\frac{p\pi}{N}\right)-1\right)}{2\left(1-\cos\left(\frac{p\pi}{N}\right)\right)} \\
&= \frac{\left(\cos\left(\frac{p\pi}{N}\right)-1\right)\left(1+\cos\left(p\pi\right)\right)}{2\left(1-\cos\left(\frac{p\pi}{N}\right)\right)} \\
&= -\frac{1+(-1)^p}{2}
\end{aligned}$$

Daraus folgt:

$$\begin{aligned}
\langle g_k, g_m\rangle &= \frac{1}{N}\left(1+\Re\left(\sum_{n=1}^{N-1}W_{2N}^{n(k+m+1)}+\sum_{n=1}^{N-1}W_{2N}^{n(k-m)}\right)\right) \\
&= \frac{1}{N}\left(1-\frac{1+(-1)^{k+m+1}}{2}-\frac{1+(-1)^{k-m}}{2}\right) \\
&= \frac{1}{N}\left(1-\frac{2+(-1)^{k+m+1}+(-1)^{k-m}}{2}\right)=0
\end{aligned}$$

Betrachte dies nun für den Fall $k=m$:

$$\begin{aligned}
\langle g_k, g_k\rangle &= \frac{1}{N}\left(1+\Re\left(\sum_{n=1}^{N-1}W_{N}^{n(2k+1)}+\sum_{n=1}^{N-1}W_{N}^{n\cdot 0}\right)\right) \\
&= \frac{1}{N}\left(1-\frac{1+(-1)^{2k+1}}{2}+N-1\right)=1
\end{aligned}$$

Das bedeutet, dass DCT-II unitär ist.

□

Bemerkung. *Ist eine Matrix M reell und unitär, so existiert ihre Inverse M^{-1} und diese ist gleich ihrer Transponierten M^T.*

Folgerung. *Aus dem Satz und den Definitionen der zweiten und dritten Diskreten Cosinus Transformation folgt, dass die DCT-III die Inverse zur DCT-II ist.*

Man kann Signale der Länge N mit der DCT-II in ihre Frequenzdarstellung transformieren. Mit der DCT-III, die explizite Inversionsvorschrift der DCT-II, kann die Frequenzdarstellung auch wieder eindeutig in die Signale aufgeschlüsselt werden. Hierbei gehen keine Bildinformationen verloren.

4 DCT via Fourier Transformation

Es gibt bereits Algorithmen, um die Diskrete Fourier Transformation schnell zu berechnen. Deshalb wird in diesem Kapitel gezeigt, dass man die DCT mit Hilfe der Fourier Transformation einfach berechnen kann. Dazu wird die Definition der Diskreten Fourier Transformation benötigt.

Definition (Diskrete Fourier Transformation)**.** *Sei $y \in \mathbb{C}^{2N}$ ein Vektor. Die Diskrete Fourier Transformation (DFT) der Länge $2N$ sei definiert durch*

$$Y(m) = \sum_{n=0}^{2N-1} y(n) W_{2N}^{nm} = \sum_{n=0}^{2N-1} y(n) \exp\left(-\frac{2nmi\pi}{2N}\right)$$

für $m = 0, \ldots, 2N-1$.

Gegeben sei nun der Vektor $\{x(n)\}$, $n = 0, \ldots, N-1$. Der Vektor $\{y(n)\}$ wird symmetrisch um den Punkt $\frac{2N-1}{2}$ konstruiert wie folgt.

$$y(n) = \begin{cases} x(n) & \text{for } n = 0, \ldots, N-1 \\ x(2N-n-1) & \text{for } n = N, \ldots, 2N-1 \end{cases}$$

Setze nun $y(n)$ in die DFT für $m = 0, \ldots, 2N-1$ ein.

$$\begin{aligned} Y(m) = \sum_{n=0}^{N-1} y(n) W_{2N}^{nm} &= \sum_{n=0}^{N-1} x(n) W_{2N}^{nm} + \sum_{n=N}^{2N-1} x(2N-n-1) W_{2N}^{nm} \\ &= \sum_{n=0}^{N-1} x(n) W_{2N}^{nm} + \sum_{n=0}^{N-1} x(n) W_{2N}^{(2N-n-1)m} \\ &= \sum_{n=0}^{N-1} x(n) \left[W_{2N}^{nm} + W_{2N}^{-(n+1)m} \right] \end{aligned}$$

Diese Gleichung wird nun mit $\frac{1}{2} W_{2N}^{\frac{m}{2}}$ multipliziert. Wieder für $m = 0, \ldots, N-1$.

$$\begin{aligned} \frac{1}{2} W_{2N}^{\frac{m}{2}} Y(m) &= \frac{1}{2} \sum_{n=0}^{N-1} x(n) \left[W_{2N}^{(n+\frac{1}{2})m} + W_{2N}^{-(n+\frac{1}{2})m} \right] \\ &= \sum_{n=0}^{N-1} x(n) \cos\left(\frac{m(n+\frac{1}{2})\pi}{N}\right) \end{aligned}$$

Vergleicht man dieses Ergebnis nun mit der DCT-II Darstellung erkennt man, dass die DCT aus der DFT wie folgt hergeleitet werden kann.

$$\begin{aligned} X(m) &= \left(\frac{2}{N}\right)^{\frac{1}{2}} k_m \frac{1}{2} W_{2N}^{\frac{m}{2}} Y(m) \\ &= \left(\frac{1}{2N}\right)^{\frac{1}{2}} k_m \exp\left(-\frac{mi\pi}{2N} Y(m)\right) \end{aligned}$$

5 Schnelle Fourier Transformation

5.1 Algorithmus

Um die Fourier Transformation schnell zu berechnen, verwendet man ein Divide-and-Conquer-Verfahren. Sei f der Input-Vektor und n seine Länge.

Rekursiver Algorithmus der FFT: Dieser Algorithmus funktioniert nur, wenn die Länge f eine

```
1: function FFT(n, f)
2:     if n = 1 then
3:         return f
4:     else
5:         g = fft(n/2, (f_0, f_2, ..., f_{n-2}))
6:         h = fft(n/2, (f_1, f_3, ..., f_{n-1}))
7:         for k = 0 TO n/2 - 1 do
8:             c_k = g_k + h_k · exp(-2πi/n)
9:             c_{k+n/2} = g_k - h_k · exp(-2πi/n)
10:        end for
11:        return c
12:    end if
13: end function
```

Zweierpotenz ist!

Durch Umformen sieht man, dass die DFT mit dem FFT Algorithmus schnell berechnet werden kann.

Für $m = 0, \dots, 2N-1$:

$$\begin{aligned} f_m &= \sum_{n=0}^{2N-1} x_n \exp\left(-\frac{2\pi i}{2N} mn\right) \\ &= \sum_{k=0}^{N-1} x_{2k} \exp\left(-\frac{2\pi i}{N} mk\right) + \sum_{k=0}^{N-1} x_{2k+1} \exp\left(-\frac{2\pi i}{2N} m(2k+1)\right) \\ &= \sum_{k=0}^{N-1} \tilde{x_k} \exp\left(-\frac{2\pi i}{N} mk\right) + \exp\left(-\frac{\pi i}{N} m\right) \sum_{k=0}^{N-1} \hat{x_k} \exp\left(-\frac{2\pi i}{N} mk\right) \end{aligned}$$

mit $\tilde{x_k} = x_{2k}$ und $\hat{x_k} = x_{2k+1}$ für $k = 0, \dots, N-1$.

Für $m < n$ gilt

$$f_m = \sum_{k=0}^{N-1} \tilde{x_k} \exp\left(-\frac{2\pi i}{N} mk\right) + \exp\left(-\frac{\pi i}{N} m\right) \sum_{k=0}^{N-1} \hat{x_k} \exp\left(-\frac{2\pi i}{N} mk\right)$$

und für $m \geq n$ gilt

$$f_m = \sum_{k=0}^{N-1} \tilde{x_k} \exp\left(-\frac{2\pi i}{N} (m-n)k\right) + \exp\left(-\frac{\pi i}{N} (m-n)\right) \sum_{k=0}^{N-1} \hat{x_k} \exp\left(-\frac{2\pi i}{N} (m-n)k\right)$$

5.2 Laufzeit

Um zu zeigen, dass die FFT schneller ist als die trivial berechnete DFT (TDFT) wird die maximale Anzahl von Multiplikationen der beiden verglichen. Damit der Vergleich unabhängig von verschiedenen Systemen ist, wird die $\mathcal{O}$-Notation verwendet.

Bei der DFT eines Vektors mit N Elementen wird ein Element auf triviale Weise mit folgenden Schritten berechnet.

$$\begin{aligned} Y(m) &= \sum_{n=0}^{N-1} y(n) \exp\left(-\frac{2i\pi}{N} nm\right) \\ &\leq \sum_{n=0}^{N-1} y(n) \exp\left(-\frac{2i\pi}{N} N^2\right) \\ &= \sum_{n=0}^{N-1} y(n) \exp\left(-2i\pi N\right) \end{aligned}$$

Hier werden für $y(n) \cdot \exp(\dots)$ eine Multiplikation und für $\exp(-2i\pi N) = \exp(-2i\pi)^N$ $N-1$ Multiplikationen gezählt, da hier die Konstante $\exp(-2i\pi)$ $N-1$-mal zu sich selbst hinzu multipliziert wird. Gesamt sind das N Multiplikationen.

Diese Rechnung muss für jedes m gemacht werden, also noch weitere N-mal. Das führt zu einer Gesamtlaufzeit von $\mathcal{O}(N^2)$.

Die Laufzeit des oben beschriebenen Algorithmus mit Inputgröße N wird folgendermaßen bestimmt. Die *if-else*-Abfrage, die *for*-Schleifen-Abfrage, die Rechnungen in der *for*-Schleife und die *Return*-Befehle benötigen konstante Zeit $\mathcal{O}(1)$. Entscheidend für die Laufzeit sind die zwei rekursiven Aufrufe der FFT mit jeweils $\frac{N}{2}$ Elementen und die maximale Durchlaufzahl $\frac{N}{2}$ der *for*-Schleife.

Mit folgender rekursiven Zeitgleichung und der Abbruchbedingung $k = \log_2(n)$ kann die Laufzeit hergeleitet werden.

$$\begin{aligned} \mathcal{T}(N) &= 2\mathcal{T}\left(\frac{N}{2}\right) + \frac{N}{2} \\ &= \dots \\ &= 2^k \mathcal{T}\left(\frac{N}{2^k}\right) + k \cdot \frac{N}{2} \\ &= N \cdot \mathcal{O}(1) + \log_2(N) \cdot \frac{N}{2} \cdot \mathcal{O}(1) \\ &= \mathcal{O}(N) + \mathcal{O}\left(\frac{N}{2} \cdot \log_2(N)\right) \\ &= \mathcal{O}(N \cdot \log(N)) \end{aligned}$$

Mit einer Laufzeit von $\mathcal{O}(N \cdot \log(N))$ berechnet die FFT die DFT eindeutig schneller, als die triviale Berechnungsvariante mit $\mathcal{O}(N^2)$ Schritten.

6 Reduktion der 2D-DCT auf die 1D-DCT

Definition. *Die 2D-DCT ist für* $u = 0, \ldots, M-1$ *und* $v = 0, \ldots, N-1$ *definiert durch:*

$$[G]_{uv} = \frac{2 \cdot c(u) \cdot c(v)}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} [g]_{mn} \cos\left(\frac{\left(m + \frac{1}{2}\right) u\pi}{M}\right) \cos\left(\frac{\left(n + \frac{1}{2}\right) v\pi}{N}\right)$$

wobei

$$c(k) = \begin{cases} \frac{1}{\sqrt{2}} & k = 0 \\ 1 & \textit{sonst} \end{cases}$$

$[G]_{uv}$ kann man jetzt umformen:

$$[G]_{uv} = \underbrace{\sqrt{\frac{2}{M}} c(u) \sum_{m=0}^{M-1} \underbrace{\left[\sqrt{\frac{2}{N}} c(v) \sum_{n=0}^{N-1} [g]_{mn} \cos\left(\frac{\left(n + \frac{1}{2}\right) v\pi}{N}\right)\right]}_{\text{N-Punkt 1D-DCT der Zeilen}} \cos\left(\frac{\left(m + \frac{1}{2}\right) u\pi}{M}\right)}_{\text{M-Punkt 1D-DCT der (bereits Zeilen-transformierten) Spalten}}$$

Folglich benötigt man für die 2D-DCT einer $M \times N$ Matrix M N-Punkt 1D-DCTs, gefolgt von N M-Punkt 1D-DCTs. Wie im vorigen Kapitel gezeigt, kann die N-Punkt 1D-DCT mit $\mathcal{O}(N \log(N))$ und die M-Punkt 1D-DCT in $\mathcal{O}(M \log(M))$ Schritten berechnet werden.
Die Gesamtlaufzeit für die 2D-DCT ist folglich $M \cdot \mathcal{O}(N \log(N)) + N \cdot \mathcal{O}(M \log(M)) = \mathcal{O}(MN \cdot \max(\log(M), \log(N)))$.

Nun stellt sich die Frage, ob die 2D-Transformation wirklich schneller sein kann als die 1D-Transformation. Dazu ist wieder eine $M \times N$ Matrix gegeben. Diese wird ganz naiv in einen Spaltenvektor umgewandelt, indem man die Spalten der Matrix aneinander reiht. Der so entstehende Vektor ist also ein $MN \times 1$-Vektor. Auf diesen Vektor wird die MN-Punkt 1D-DCT angewandt. Diese benötigt laut vorigem Kapitel $\mathcal{O}(MN \cdot \log(MN))$ Rechenschritte.

Aus dem Verglich der benötigten Rechenschritte geht hervor, dass die 2D-orthogonal Transformation schneller ist als die 1D Transformation. Noch deutlicher ist dies ersichtlich, wenn die 1D-DCT nicht mit der FFT sondern auf triviale Weise berechnet wird. In diesem Fall kommt man für eine $M \times N$ Matrix mit der 2D-DCT auf $\mathcal{O}(MN \cdot \max(M, N))$ Rechenschritte, während für die 1D-DCT $\mathcal{O}(M^2 N^2)$ Schritte benötigt werden.

7 Kompression

7.1 Quantisierung

Das menschliche Auge ist unterschiedlich empfindlich bezüglich Variationen verschiedene Frequenzen. Auf grobe Strukturen reagiert das Auge sensibler, daher werden die niederfrequenten Werte nicht so stark skaliert wie die hochfrequenten. Die Skalierungsfaktoren werden in einer Matrix dargestellt, der so genannten Quantisierungsmatrix Q.

Die DCT ordnet die 64 Bildpunkte nach Frequenzanteilen, dies ist in Abbildung 4 zu sehen. In der Ecke links oben steht der Mittelwert der Helligkeit aller Bildpunkte. Von links nach rechts verlaufen aufsteigend die horizontalen Frequenzanteile. Von oben nach unten verlaufen - ebenfalls aufsteigend - die vertikalen Frequenzanteile.

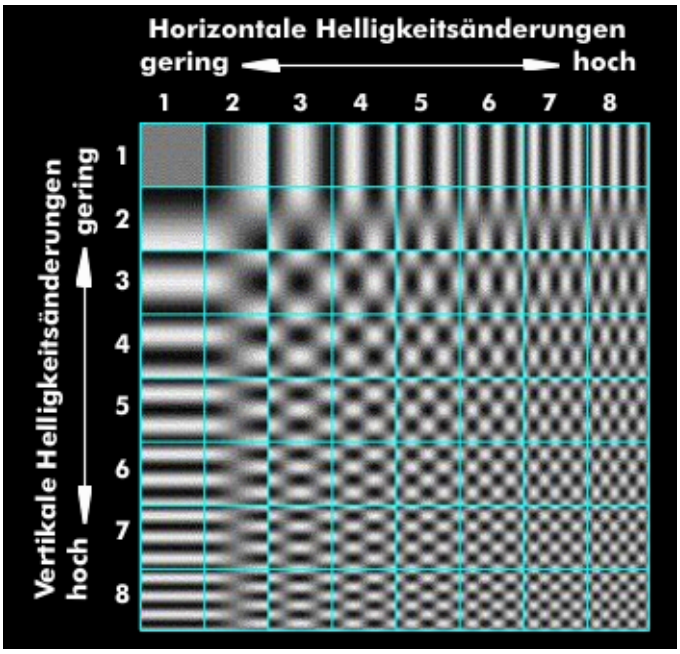


Abbildung 4: Frequenzen der DCT

Aus diesen Gründen werden die Werte der Quantisierungsmatrix ausgehend von einem kleinen Wert in der Ecke links oben nach rechts unten immer größer. Die Quantisierung der Matrix berechnet man, indem ihre Einträge elementweise durch die Quantisierunsmatrix dividiert und anschließend auf die nächstliegende Ganzzahl gerundet werden.

$$X^Q(m,n) = \left\lfloor \frac{X(m,n)}{Q(m,n)} \right\rfloor$$

Durch das Runden gehen bestimmte Informationen im Bild verloren, was einen Verlust der Genauigkeit zur Folge hat.

Die hier beschriebene Quantisierung ist besonders gut für natürliche Bilder geeignet, da bei diesen keine starken Kontraste auftreten. Hingegen ist sie nicht für Bilder mit scharfen Kontrasten geeignet. Ein Beispiel für ein solches Bild wäre das Schachbrett.

7.2 DC- und AC-Koeffizienten und Zick-Zack-Folge

Wie bereits bei der Quantisierung erwähnt, ordnet die DCT die 64 Bildpunkte nach Frequenzanteilen und in der linken oberen Ecke steht der Mittelwert der Helligkeit des 8×8 Blocks. Dieser Mittelwert wird auch als DC - Direct Current - Koeffizient bezeichnet. Die anderen Werte, welche die verschiedenen Frequenzanteile darstellen, werden als AC - Alternating Current - Koeffizienten bezeichnet. Diese Begriffsbezeichnungen stammen aus der Elektrotechnik.

Da die AC-Koeffizienten recht kleine Werte annehmen und durch die Quantisierung noch einmal vermindert werden, benötigen sie bereits wenig Speicherplatz. Hingegen kann der DC Wert recht groß werden. Um auch den DC mit geringem Platz zu speichern, wird erneut auf die Eigenschaft von geringen Farbschwankungen bei natürlichen Bildern zurückgegriffen. Auf Grund dieser Eigenschaft weisen benachbarte 8×8 Blöcke eine starke Korrelation auf und die DC Werte benachbarter Blöcke liegen nahe beieinander. Deswegen kann ein DC Wert als Differenz zu dem DC Wert des Vorgängerblocks dargestellt werden und benötigt somit weniger Speicherplatz.

Durch die Quantisierung gehen irrelevante Informationen verloren. Das betrifft vor allem die hochfrequenten Anteile der DCT, die im unteren und rechten Bereich der Matrix zu finden sind. Folglich haben viele Einträge in diesem Bereich den Wert 0.

Die 8×8 Blöcke werden linear in der JPEG-Datei gespeichert. Die 8×8 Blöcke können also wieder trivial Spalte für Spalte aneinander gereiht werden. Um jedoch effizienter codieren zu können, sollen möglichst viele 0 Werte zusammengefasst werden.

Dafür bietet sich die Umsortierung der DCT-Koeffizienten nach dem Zick-Zack-Muster an (vgl. Abbildung 5) Diese Umsortierung liefert wie gewünscht möglichst viele 0 Werte am Ende der Anordnung. Die 0 Wert können folglich zu einem gesonderten Steuerzeichen zusammengefasst werden, wodurch deutlich weniger Zeichen zu kodieren sind.

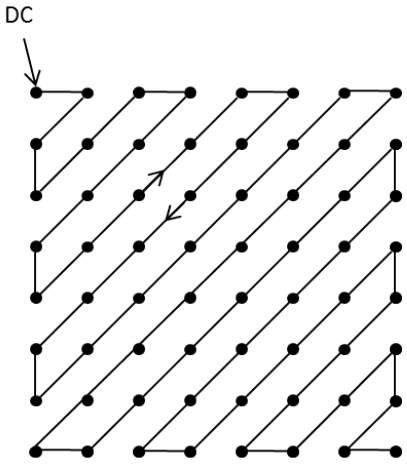


Abbildung 5: Zick-Zack-Sortierung

8 Entropie-Codierung

Die Idee der Entropie-Codierung ist, dass, je nachdem wie häufig ein Wert vorkommt, er mit einer längeren oder kürzeren Zeichenfolge gespeichert wird. Werte, die häufig vorkommen, werden mit einer kurzen Zeichenfolge gespeichert und umgekehrt werden Werte, die selten vorkommen, mit einer längeren Zeichenfolge gespeichert.

Definition (Entropie)**.** *Für ein Zufallsexperiment in einem endlichen Wahrscheinlichkeitsraum* (Ω, p) *ist die wahre Entropie* $H_0(p)$ *definiert als der Erwartungswert der Anzahl benötigter Ereignisse bei optimaler Ereigniswahl, um den Ausgang des Zufallsexperiments zu erfahren. Es gilt:*

$$H(p) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

Einfacher ausgedrückt ist die Entropie die minimale Länge, um einen Sachverhalt vollständig zu beschreiben beziehungsweise Symbole eindeutig binär zu speichern. Eine große Entropie bedeutet, dass die Information auch viel Speicher benötigt, um beschrieben zu werden. Eine kleine Entropie besagt, dass zum Beschreiben einer Information wenig Speicher nötig ist. Bei der Entropie-Codierung soll die Gesamtentropie, Summe der Entropie aller Symbole, minimal sein.

Damit Werte nach der Entropie-Codierung eindeutig wieder rekonstruiert werden können, muss die Codierungs-Abbildung injektiv sein und ein Codewort darf nicht als Präfix eines anderen Codeworts vorkommen.

Was passiert, wenn ein Codewort als Präfix eines anderen Codeworts vorkommt, ist in Abbildung 6 zu sehen.

a = 10
b = 101 abc = 101010 = aaa
c = 0

Abbildung 6: Präfix als eigenes Codewort

Ein Verfahren, das diese Bedingungen erfüllt und auch von JPEG zur Codierung verwendet wird, ist die Huffman-Codierung.

8.1 Huffman-Codierung

Idee der Huffman-Codierung:
Man stellt die Codierung als k-nären Wurzelbaum dar, wobei die Blätter den zu codierenden Symbolen entsprechen. Der Baum wird von den Blättern zur Wurzel erstellt. Das Codesymbol wird durch den Pfad von der Wurzel zum Blatt bestimmt.

Sei X die Zufallsvariable, dass ein gewisses Ereignis eintritt und $\bar{l}$ die mittlere Länge eines Codeworts. Der Huffman-Code ist optimal und es gilt $H(X) \leq \bar{l} \leq H(X) + 1$.

Es folgt ein Beispiel zur Veranschaulichung, dass die Huffman-Codierung weniger Speicherplatz benötigt, als wenn eine einfache Codierung - jedes Symbol gleich lang - verwendet wird.

Codiere das Wort MATHEMATIK.

MATHEMATIK hat sieben verschiedene Buchstaben.

Man kann dieses Wort nun auf einfache Weise codieren, indem man - bei sieben Buchstaben - jedem Buchstaben einen drei-Bit Wert zuweist. Diese Darstellung ist eindeutig, weil jedes Symbol drei Bit hat und somit rekonstruiert werden kann.

M	A	T	H	E	I	K
000	001	010	011	100	101	110

$$\text{MATHEMATIK} = 000001010001100000001010101110$$

MATHEMATIK hat gesamt zehn Buchstaben. Pro Buchstabe werden drei Bit gespeichert. Gesamt werden also 30 Bit benötigt.

Oder man verwendet die Entropie Codierung von Huffman. Im Fall von dem hier genannten Beispiel geht man wie folgt vor.

Die relative Häufigkeit der einzelnen Symbole bestimmen und absteigend ordnen:

$\frac{2}{10}$	$\frac{2}{10}$	$\frac{2}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$
M	A	T	H	E	I	K

Die zwei Symbole mit kleinster Häufigkeit werden zusammengefasst und ihnen werden die Werte 1 und 0 zugewiesen. Wieder absteigend nach Häufigkeit ordnen:

$\frac{2}{10}$	$\frac{2}{10}$	$\frac{2}{10}$	$\frac{2}{10}$	$\frac{1}{10}$	$\frac{1}{10}$
M	A	T	I K	H	E
			1 0		

Der vorige Schritt wird wiederholt.

$\frac{2}{10}$	$\frac{2}{10}$	$\frac{2}{10}$	$\frac{2}{10}$	$\frac{2}{10}$
M	A	T	I K	H E
			1 0	1 0

Erneut wird dieser Schritt wiederholt, doch diesmal haben die zusammengefassten Symbole bereits eine Binärdarstellung, weshalb hier VOR den bereits vorhandenen Werten 1 und 0 hinzugefügt werden.

$\frac{4}{10}$	$\frac{2}{10}$	$\frac{2}{10}$	$\frac{2}{10}$
I K H E	M	A	T
11 10 01 00			

Hier werden wieder zwei Symbole zusammengefügt und ihnen ein Wert zugewiesen.

$\frac{4}{10}$	$\frac{4}{10}$	$\frac{2}{10}$
I K H E	A T	M
11 10 01 00	1 0	

Der Schritt wird noch einmal wiederholt, wobei hier ein Symbol noch keinen Wert hat. Dieses Symbol bekommt den Wert 0 zugewiesen. Den anderen Symbolen wird 1 an der vordersten Stelle hinzugefügt.

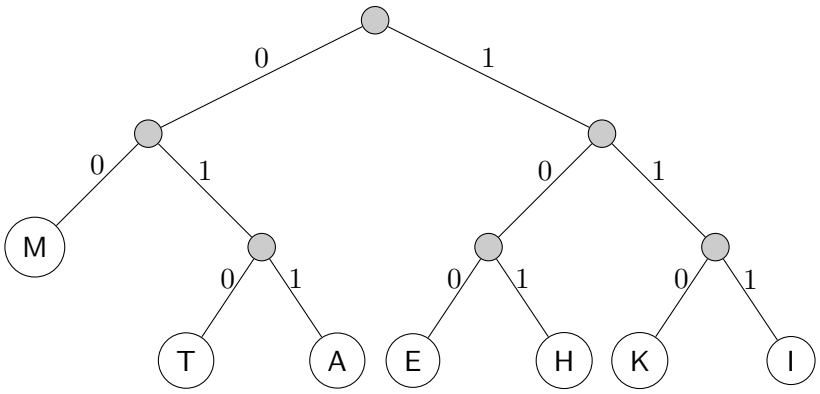
$\frac{6}{10}$	$\frac{4}{10}$
M A T	I K H E
0 11 10	11 10 01 00

Und am Ergebnis ist ersichtlich, dass Werte mit größerer Häufigkeit weniger Bit und somit Speicher verwenden.

M	A	T	I	K	H	E
00	011	010	111	110	101	100

MATHEMATIK hat also die Codierung 00011010101100000110101111110, das sind gesamt 28 Bit.

An diesem Beispiel kann man erkennen, dass man mit Entropie-Codierung Platz sparen kann. Für die einfache Codierung werden 30 Bit benötigt, mit der Huffman-Codierung hingegen nur 28 Bit.



9 Implementierung in MATLAB

Ich habe einen Teil der JPEG Bildkompression in MATLAB implementiert. In einem Skript habe ich die Schritte bis einschließlich der Quantisierung, sowie deren Umkehrung verfasst.

Als Erstes wird das Bild eingelesen und in eine passende Form gebracht, damit es gut in 8×8 Blöcke zerlegt werden kann. Ein schwarz-weiß Bild besteht aus Bildpunkten im Bereich von 0 bis 255. Diese Punkt müssen auf den Bereich -128 bis 127 verschoben werden. Das ist der nächste Schritt. Anschließend wird das Bild in 8×8 Blöcke aufgeteilt. Diese Blöcke werden hintereinander der Funktion `my_dct` übergeben, damit die DCT-II auf sie angewandt wird. Schließlich werden die transformierten Blöcke noch quantisiert.

Als Nächstes folgt die Dekompression im Skript. Dazu werden die 8×8 Blöcke dequantisiert, indem sie elementweise mit der Quantisierungsmatrix multipliziert werden. Gefolgt von dem Aufruf der Funktion `my_idct`, die Implementierung der DCT-III, mit diesen Werten. Danach müssen die 8×8 Blöcke noch wie beim ursprünglichen Bild angeordnet werden und das Bild ist fertig dekomprimiert.

Die Funktion `my_dct` wendet die 1D-DCT zuerst auf die Zeilen und anschließend auf die Spalten an. Die DCT wird dabei mit Hilfe der Funktion `my_fft` berechnet.

Die Funktion `my_idct` ist gleich wie `my_dct` aufgebaut, nur dass zuerst die Spalten und anschließend die Zeilen der Matrix transformiert werden. Auch hier wird auf eine Hilfsfunktion zurück gegriffen und zwar die Umkehrfunktion der FFT, die `my_ifft`.

Die Funktion `my_fft` ist die Implementierung des Algorithmus der FFT, der in diesem Dokument beschrieben ist. Die Funktion `my_ifft` ist die Umkehrung der FFT. Hier wird `my_fft` auf die komplex konjugierten Werte der übergebenen Matrix angewandt. Das Ergebnis sind die komplex konjugierten Werte der Rückgabewerte von der FFT.

Wie man in Abbildung 7 erkennen kann, wird das Bild nach der Kompression bis auf kaum sichtbare Unterschiede wieder rekonstruiert.

Abbildung 7: Das Bild vor und nach der Kompression

10 Quellenangabe

1. G.K. Wallace, The JPEG still picture compression standard, *IEEE Transactions on Consumer Electronics*, Vol. 38, No. 1, February 1992.
2. K.R. Rao and P. Yip, *Discrete Cosine Transform*, Academic Press Inc.,San Diego, 1990.
3. E. Feig and Sh. Winograd, Fast Algorithms for the Discrete Cosine Transform, *IEEE Transactions on Signal Processiong*, Vol. 40, No. 9, September 1992.
4. `http://www.mathematik.de/spudema/spudema_beitraege/beitraege/rooch/`
   a) `einleit.html`
   b) `kap02.html`
   c) `kap04.html`
   d) `kap06.html`
   e) `kap06b.html`
   f) `kap07.html`
   g) `kap08.html`
   h) `kap09.html`
   i) `resumee.html`

   Stand der Homepage am 03.10.2014
5. `http://de.wikipedia.org/wiki/Diskrete_Kosinustransformation`
   Stand der Homepage am 03.10.2014
6. `http://de.wikipedia.org/wiki/JPEG`
   Stand der Homepage am 08.03.2015
7. `http://www.itwissen.info/definition/lexikon/discrete-cosine-transformation-DCT-DCT-Transformation.html`
   Stand der Homepage am 08.03.2015
8. H. Hinrichsen, Entropie entmystifiziert, *Physik in unserer Zeit*, 2012