Digitale Bildverarbeitung natürliche Bilder

Eigenes Skript

Studiengang Angewandte Informatik

Duale Hochschule Baden-Württemberg Karlsruhe

von

Andre Meyering

Kurs:	TINF16B2
Dozent:	Ralph Lausen
Semester:	5. Semester (10.10.2018 - 28.11.2018)
letzte Änderung:	*5. November 2018*

Dies ist das eigene Skript für „Digitale Bildverarbeitung“ bei Herrn Ralph Lausen für das 5. Semester im Jahr 2018. Es enthält fast alles, was im Unterricht an die Tafel geschrieben oder besprochen wurde. Die LaTeX-Dateien sollten sich im gleichen Share befinden, in dem du diese PDF-Datei gefunden hast.

Bei Fragen, Fehlern oder Ergänzungen – oder sollten die LaTeX-Dateien fehlen – wende dich bitte an dhbw@andremeyering.de. Ich hoffe, diese PDF hilft dir beim Lernen.

©2018 - keine Garantie auf Korrektheit

Inhaltsverzeichnis

1 Vorwort

2 Einstieg in die Vorlesung

Dozent:	Ralph Lausen
Mobil:	0176-24652134
Modul:	Computergraphik und Bildverarbeitung (T2INF4303)
Fach:	Digitale Bildverarbeitung
Moodle	`https://moodle.dhbw.de/course/view.php?id=61`
Klausur:	90min

2.1 Buchempfehlungen

- Grundlagen der Bildverarbeitung (Klaus D. Tönnies)
- Digitale Bildverarbeitung (Bernd Jähne)

3 Einführung

3.1 Einsatzgebiete

Bildverarbeitung wird mittlerweile in nahezu allen Wissenschafts- und Ingenieursdisziplinen eingesetzt, wie beispielsweise in der modernen Mikroskopie, medizinischen Diagnostik, Astronomie, Maschinenbau und in der Fernerkundung (Umweltbeobachtung, Spionage). Mit Methoden der Bildverarbeitung werden in Maschinen Objekte gezählt, vermessen, Objekte inspiziert oder codierte Informationen gelesen. Röntgen- und Ultraschallgeräte liefern mit der Bildverarbeitung Bilder, die der Arzt einfacher deuten kann. Röntgengeräte in Sicherheitszonen untersuchen Gepäck und Kleidung automatisch nach gefährlichen Objekten (Waffen etc.).

Im Gegensatz zur Bildbearbeitung, welche sich mit der Manipulation von Bildern zur anschließenden Darstellung beschäftigt, umfasst die Bildverarbeitung weitergehende Bearbeitungsprozeduren zur Extraktion von Information aus den Ursprungsdaten: z. B. Bewegungsbestimmung, Bildsegmentierung, Bilderkennung und Mustererkennung.

– Ralph Lausen, Moodle Raum

Womit beschäftigt sich die Bildverarbeitung?

Mit *natürlichen* Bildern.

Dies wurde von Herrn Lausen mehrmals erwähnt, da viele Studenten in der Klausur vergessen „natürlich“ zu erwähnen. Deshalb an dieser Stelle der Hinweis, dass wir uns in der Vorlesung „Bildverarbeitung“ mit *natürlichen* Bildern beschäftigen.

3.2 Ablauf der Bildverarbeitung

1. Bilderfassung
2. Vorverarbeitung
3. Segmentierung
4. Merkmalsextraktion
5. Klassifikation

4 Bilderfassung

4.1 Natürliche Bilder

Eine Beleuchtungsfunktion sit analog, kontinuierlich/stetig, sodass Intensität $I(x,y) \rightarrow$ 2D, kontinuierlich. Ein Sensor erfasst ein Bild und digitalisiert es, wodurch wir eine diskrete Funktion $g(m,n)$ erhalten.

4.1.1 Sensoren

Die Auflösung eines Sensors wird in dpi (Punkte pro Länge) angegeben. Wir unterscheiden zwischen:

- **Durchleuchtung** (z.B. Röntgen, Abbildung 4.1)
  - Absorption
  - Streuung
  - Rückstreuung
  - Rauschen
- **Beleuchtung** (Abbildung 4.2)
  - Reflektion
  - Schatten
  - Oberflächen

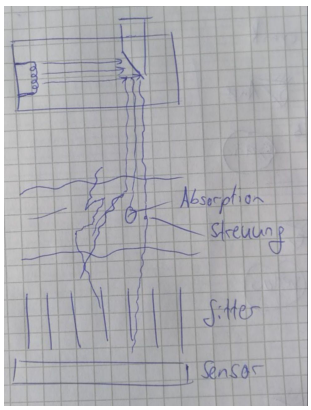


Abbildung 4.1: Sensor – Durchleuchtung (Röntgen)

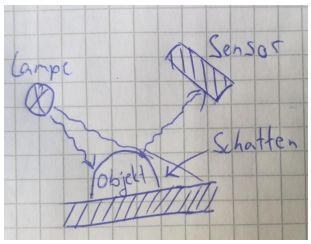


Abbildung 4.2: Sensor – Beleuchtung

4.1.2 Sensor: CCD (Charge Coupled Device)

Ein Charge Coupled Device (CCD) Sensor ist ein Halbleiter Bildsensor bestehend aus eine großen Anzahl an fotosensitiven Elementen (Zellen/Pixeln). Meist wird Silizium als Halbleiter verwendet. Abbildung 4.1 zeigt die Physik hinter diesem Sensor.

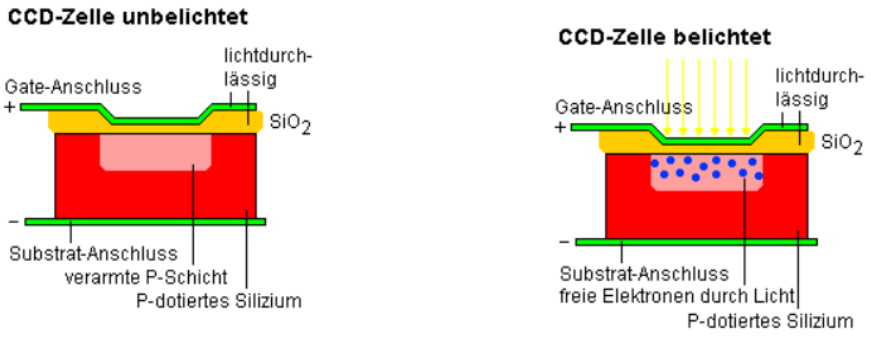


Abbildung 4.1: CCD – Physik

Jedes Element (Pixel) sammelt elektrische Ladungen, die durch das Absorbieren von Photonen (also des einfallenden Lichts) erzeugt wird. Dies ist möglich, da die Photonen Elektronen des P-Dotierten Halbleiters vom Valenz- ins Leitungsband heben. Hierdurch entstehen Elektronen-Loch-Paare. Diese Ladungen werden nacheinander von Sensorelement zu Sensorelement über den Chip in einen mit schwarzer Folien abgedeckten Bereich transportiert (Stichwort Interline-CCD-Sensor). Dieser dient dazu, die Informationen zu schützen während sie ausgelesen werden.

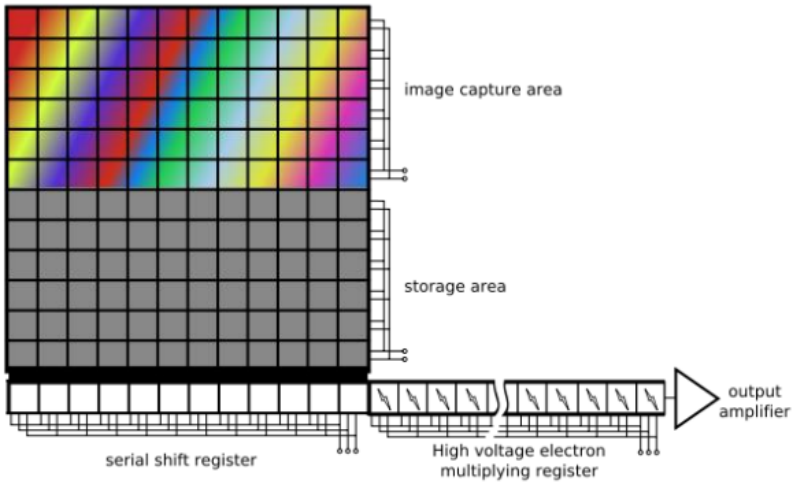


Abbildung 4.2: CCD – Gitter

Es ist anzumerken, dass CCD Sensoren anfällig sind für verschiedene Arten von Rauschen:

- Photonenrauschen $n(\gamma) \sim n(e) \pm \sqrt{n(e)}$
  (poissonverteilt; Verbesserung durch gute Beleuchtung)
- CCD Rauschen / Dunkelstromrauschen: Elektronen lösen sich ungewollt durch Wärmeeinfluss (Verbesserung durch Kühlen)
- Verstärker-Rauschen (Verbesserung durch geringere Verstärkung)
- Quantisierungs-Rauschen (Analog/Digital)

Wie können nun Farben auf dem CCD unterschieden werden? Bei 1-CCD kann hierfür eine sogenannte Bayer-Maske verwendet werden, wie in Abbildung 4.3 dargestellt. Bei dieser Maske

ist jedes Pixel in 4 Subpixel eingeteilt, zwei davon grün und eins jeweils blau und rot. Hierdurch wird das einfallende Licht gefiltert und es ist möglich, Farben zu messen und zu speichern (siehe Kapitel 5).

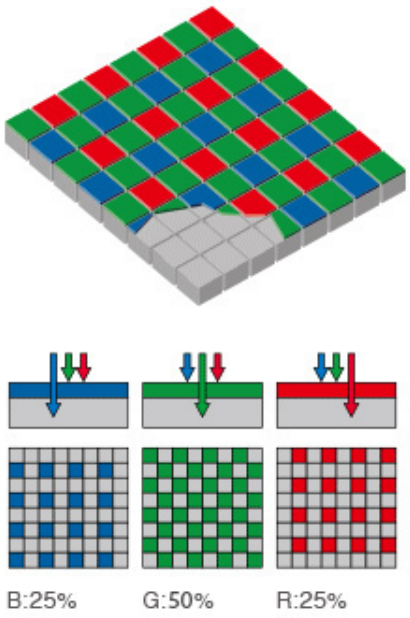


Abbildung 4.3: CCD – Bayer Maske

Dies führt uns zur Pixellüge: Bei 4 Mio. Pixel sind 2 Mio. grün, 1 Mio. rot und 1 Mio. blau. Es gibt somit doppelt so viele grüne wie rote und blaue Pixel.

Die andere Variante ist 3-CCD, wie in Abbildung 4.4 gezeigt wird. Hierbei gibt es drei Sensoren, die jeweils für eine der Farben rot, blau und grün zuständig sind.

Was sollten wir Lernen? TODO CCD Sensor, PhotoEffekt, Photonenrauschen (Poisson Statistik)„ Phononen/CCD Rauschen, Verstärker/Quantisierungsrauschen, Bayer Maske,

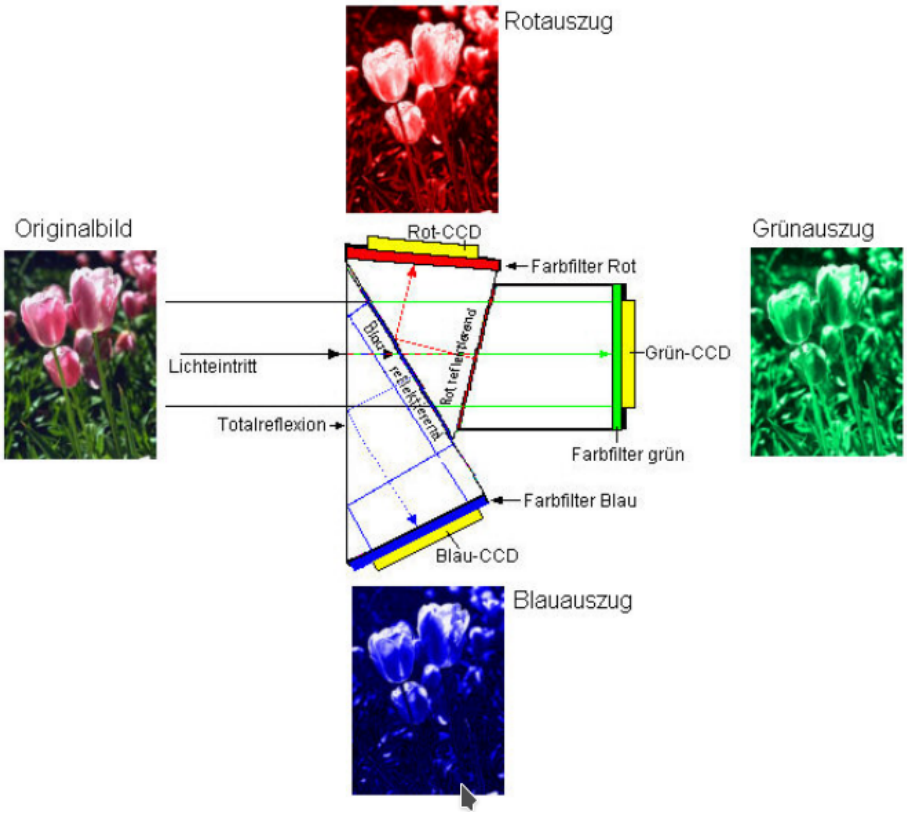


Abbildung 4.4: 3-CCD

5 Farbsysteme

Farben

TODO: Andre Grafik 5

rgba <–> cmyk

6 Bildvorverarbeitung

6.1 Codierung und Kompression

6.1.1 title

LZW / LZ78 Der Lempel-Ziv-Welch (LZW)-Algorithmus ist die bekannteste Form der LZ78-Kompression. Listing 6.1 zeigt die Funktionsweise dieses Algorithmus.

```
1  initialisiere Mustertabelle mit (<leeres Muster>+zeichen) für alle Zeichen
2  muster := <leeres Muster>
3  solange noch Zeichen verfügbar
4      zeichen := lies nächstes Zeichen
5      wenn (muster+zeichen) in Mustertabelle dann
6          muster := (muster+zeichen)
7      sonst
8          füge (muster+zeichen) zur Mustertabelle hinzu
9          Ausgabe muster
10         muster := zeichen
11 wenn muster nicht <leeres Muster> dann
12     Ausgabe muster
```

Listing 6.1: Funktionsweise LZW-Algorithmus

Die nachfolgende Tabelle komprimiert die Zeichenkette „LZWLZ78LZ77LZCLZMWLZAP“ mithilfe des LZW-Algorithmus.

Zeichenkette	gefundener Eintrag	Ausgabe	neuer Eintrag
LZWLZ78LZ77LZCLZMWLZAP	L	L	LZ (wird zu <256>)
ZWLZ78LZ77LZCLZMWLZAP	Z	Z	ZW (wird zu <257>)
WLZ78LZ77LZCLZMWLZAP	W	W	WL (wird zu <258>)
LZ78LZ77LZCLZMWLZAP	LZ (= <256>)	<256>	LZ7 (wird zu <259>)
78LZ77LZCLZMWLZAP	7	7	78 (wird zu <260>)
8LZ77LZCLZMWLZAP	8	8	8L (wird zu <261>)
LZ77LZCLZMWLZAP	LZ7 (= <259>)	<259>	LZ77 (wird zu <262>)
7LZCLZMWLZAP	7	7	7L (wird zu <263>)
LZCLZMWLZAP	LZ (= <256>)	<256>	LZC (wird zu <264>)
CLZMWLZAP	C	C	CL (wird zu <265>)
LZMWLZAP	LZ (= <256>)	<256>	LZM (wird zu <266>)
MWLZAP	M	M	MW (wird zu <267>)
WLZAP	WL (= <258>)	<258>	WLZ (wird zu <268>)
ZAP	Z	Z	ZA (wird zu <269>)
AP	A	A	AP (wird zu <270>)
P	P	P	-

Die Ausgabe wäre somit L Z W <256> 7 8 <259> 7 <256> C <256> M <258> Z A P.

6.1.2 Huffman (Entropiekodierung)

Die Huffman-Kodierung ist eine verlustfreie Entropiekodierung, die eine Kompression um den Faktor ~ 3 ermöglicht. Dabei ist er ein präfixfreier Code. In seiner einfachsten Implementierung werden die vorkommenden Zeichen nach ihrer Häufigkeit sortiert. Das am häufigsten vorkommende Zeichen erhält die Kodierung `1`, das zweithäufigste `01`, das dritthäufigste `001`, usw. fig:huffman zeigt eine Huffman-Tabelle, wie sie zur Kodierung verwendet werden kann.

Codierung	Zeichen	Häufigkeit
`1`	E	10%
`01`	A	9%
`001`	M	6%
`0001`	...	

Abbildung 6.1: Huffmantabelle – Einfachste Implementierung

Hinweis

Herr Lausen hat nach obigem Muster die Huffman Kodierung vorgenommen. Hierbei handelt es sich um eine sehr ineffiziente Art, Huffman zu verwenden. An dieser Stelle sei auf andere Vorlesungen[a] verwiesen, die dieses Thema ausführlicher behandeln.

[a] `http://www.ziegenbalg.ph-karlsruhe.de/materialien-homepage-jzbg/cc-interaktiv/huffman/codierung.htm`

6.1.3 Lauflängenkodierung

Die „Lauflänge“ an gleicher Zeichen, also die Anzahl aufeinanderfolgender gleicher Zeichen wird gespeichert. Verwendet wird dies u. a. beim Fax. Aus `SSSSSWWWWWWWSSS` wird hierdurch `5 7 3`. Bei natürlichen Bilder kann dies auch sinnvoll sein, da der Himmel z. B. eine ähnliche Helligkeit besitzt (siehe Abschnitt 6.2).

6.2 JPEG

Joint Photographic Experts Group (JPEG) ist ein Format zum Speichern *natürlicher* Bilder.

Bei üblichen Bildern werden pro Pixel drei Farben (rot, grün, blau) gespeichert, wodurch sich pro Pixel ein Speicherbedarf von 24 Bit ergibt. Gute Scanner verwenden pro Subpixel sogar 16 Bit und damit 48 Bit pro Pixel.

6.2.1 Konvertierung in ein geeignetes Farbmodell

Anstatt pro Pixel 24 Bit zu speichern, kann der RGB Farbraum auch nach YCbCr umgewandelt werden. Nehmen wir 4×4 Pixel: Es wird die Helligkeit Y (für Gamma) für jedes Pixel abgespeichert, zusätzlich kommt noch 1 Byte für die Farbkomponente Cb (Blue-Yellow Chrominance) und 1 Byte für Cr (Red-Green Chrominance) hinzu. Wie sich aus Abbildung 6.2 ergibt, werden somit anstatt $3 \times 4 \times 8 = 96$ Bit nur $6 \times 8 = 48$ Bit gespeichert, was eine Speicherverbesserung von 50% mit sich bringt. Abbildung 6.3b veranschaulicht die Unterteilung in diese Farben. Anstatt des YCbCr Farbraums wird oft auch YUV verwendet. Y ist dabei wieder die Luminanz und U und V bilden den eigentlichen Farbwert. Abbildung 6.3c veranschaulicht dies.

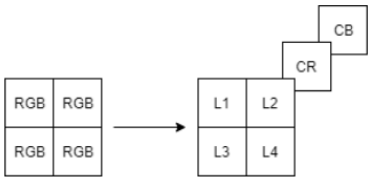
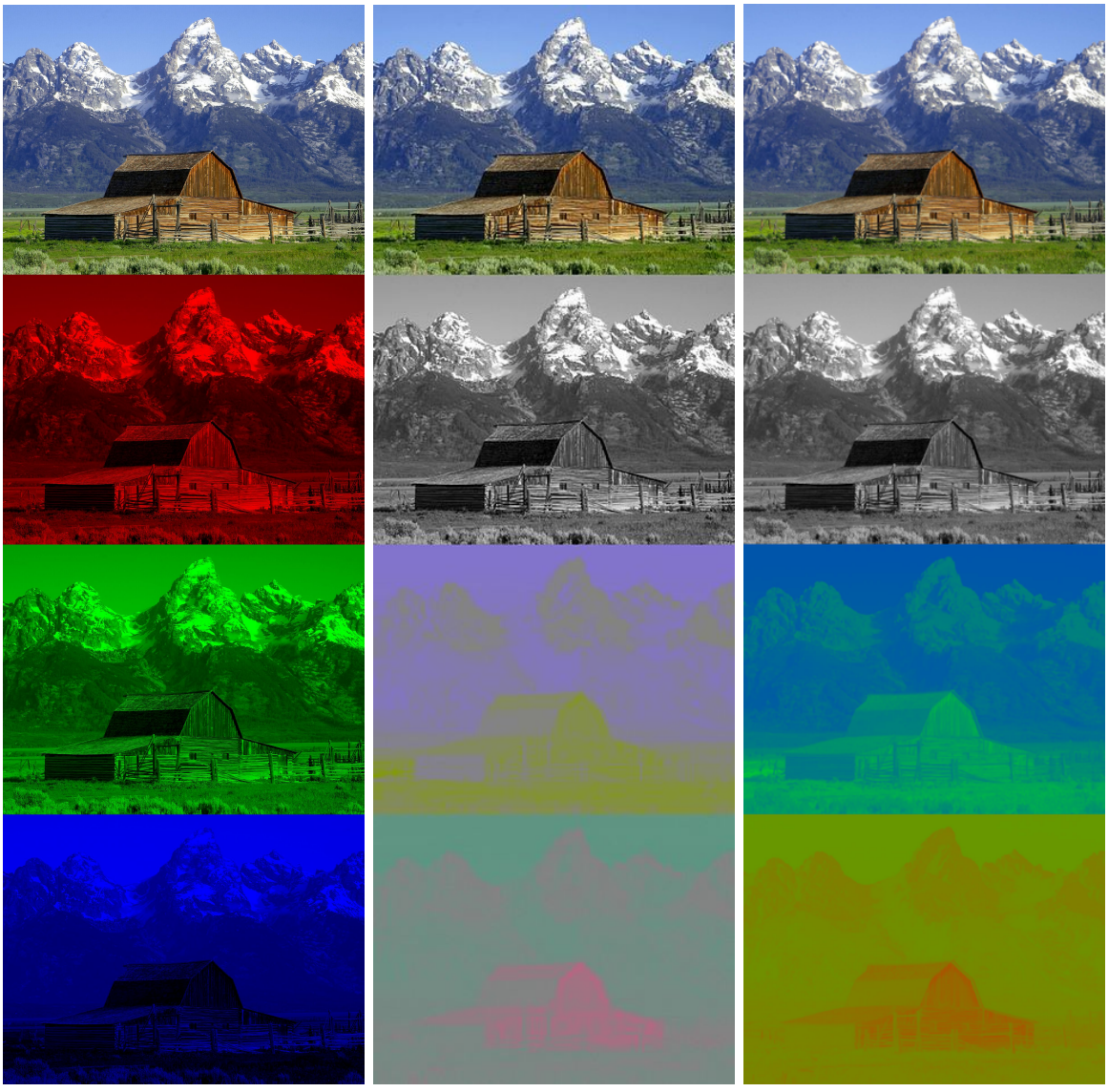


Abbildung 6.2: RGB Alternative – YCbCr



(a) RGB - Original, R, G, B (b) YCbCr - Original, Y, Cb, Cr (c) YUV - Original, Y, U, V

Abbildung 6.3: Farbräume visualisiert

Diskrete Kosinustransformation

JPEG unterteilt das Bild in 8×8 Pixel große Subbilder. DCT

DCT, Formeln...

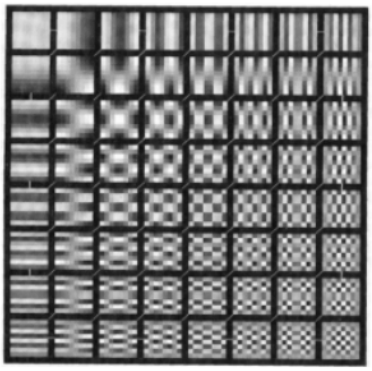


Abbildung 6.4: Diskrete Kosinustransformation (DCT)

Quantisierung

$$\hat{G}_{uv}^{q} = INT[\frac{G_{uv}}{q_{uv}}]$$

$$q_{uv} = \begin{bmatrix} 1 & 2 & 4 & \cdot & \cdot & \cdot \\ 2 & 1 & 4 & \cdot & \cdot & \cdot \\ 2 & 8 & 8 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 0 \end{bmatrix}$$

In den meisten Bildern sind viele Koeffizienten nahezu 0. Setzt man diese zu 0, dann ist dies nach der Dekompression kaum bemerkbar! Weiterhin sind einige Koeffizienten wichtiger als andere! So ist z. B. die Grundhelligkeit und der Helligkeitsverlauf am wichtigsten (niedrige Frequenzen). Weniger wichtig ist die Textur des Bildes (mittlere Frequenzen). Sehr feine hochfrequente Details sind nahezu nicht beobachtbar und unwichtig. Die Koeffizienten werden also durch unterschiedlich starke Quantisierung „gleich wichtig“ gemacht. Die Quantisierung erfolgt mit einer Quantisierungsmatrix q (siehe Formel oben). Jeder Koeffizient der DCT-Matrix wird durch einen entsprechenden Wert (aus der Quantisierungsmatrix) geteilt. Es gibt keine vorgeschriebenen Standardtabellen für Quantisierungsmatrizen. Somit lässt sich die Bildqualität über die Quantisierung steuern. Dies hat zur Folge, dass die genutzte Quantisierungsmatrix im Bild mit transportiert werden muss, um es an anderer Stelle wieder dekodieren zu können. Dies erhöht zwar den Speicherbedarf, aber erhöht auch die Flexibilität des Standards und wird somit ohne weiteres geduldet.

Ist q_{uv} überall 1, dann hat das Bild die beste Qualität, da die Amplituden unverändert bleiben.

Beispiel

$$\hat{G}_{uv} = 31$$

$$q_{uv} = 8$$

$$INT[\frac{G_{uv}}{q_{uv}}] = 3 \rightarrow Dekodierung: 3 \times 8 = 24 = G_{uv}$$

MÜsste das Abrunden sein?

Serialisierung der Matrix

Die DC-Koeffizienten (Matrixinhalt an Position 1,1) benachbarter Blöcke unterscheiden sich nur wenig und werden daher als Differenz zum Vorgängerblock übertragen. Die Koeffizienten werden im Zick-Zack angeordnet, was gleichzeitig einer Anordnung nach Ihrer Wichtigkeit entsprechend der visuellen Wahrnehmung entspricht. Abbildung 6.5 stellt dies dar.

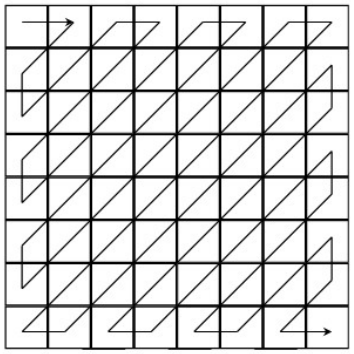


Abbildung 6.5: Serialisierung der Matrix in der Zig-Zag-Ordnung

RLC (Lauflängenkodierung)

Anschließend folgt eine Lauflängenkodierung der serialisierten Matrix. Das funktioniert aufgrund der langen Nullketten sehr gut. Siehe Unterabschnitt 6.1.3 und die vorhergehende Matrix.

Huffman-Kodierung

Es ergibt sich, dass die Huffman-Tabelle sehr klein ist.

Natürliche Bilder haben in jedem 8×8 Block Helligkeit. Schmeißen wir dies raus, dann funktioniert RCL und Huffman gut

Wie meinte er das?

Kantenerkennung bei JPEG

Grafik Diagram

6.3 JPEG 2000

6.4 Moire Effekt bei JPEG

Wenn ein Bild mehrere Male als JPEG mit unterschiedlichen Quantisierungsmatrizen gespeichert wird, so ergeben sich Artefakte, die dem Moire Effekt ähnlich sind.

7 Abkürzungsverzeichnis

dpi dots per inch
CCD Charge Coupled Device 4
DCT Diskrete Kosinustransformation
JPEG Joint Photographic Experts Group 10
LZW Lempel-Ziv-Welch 9

Abbildungsverzeichnis

Tabellenverzeichnis

Listingsverzeichnis

Stichwortverzeichnis