Digitale Bildverarbeitung natürliche Bilder

Zusammenfassung Kapitel 1 + 2

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:	*28. März 2019*

Dies ist die nicht vollständige Zusammenfassung 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.

Inhaltsverzeichnis

1 Vorwort

1.1 Vorlesung -Informationen

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

1.2 Buchempfehlungen

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

1.3 Was ist das?

Das ist meine eigene Zusammenfassung der Vorlesung „Bildverarbeitung“ bei Herrn Lausen. Sie ist leider nicht vollständig und enthält nur Kapitel 1 + 2 (Bilderfassung und Bildvorverarbeitung). Die restlichen Kapitel wurden nicht digitalisiert.

Du findest diese nicht vollständige Zusammenfassung unter: `https://gitea.ameyering.de/DHBW_AI_16/Bildverarbeitung_Lausen`. Ich würde mich über Ergänzungen freuen.

2 Einführung

2.1 Motivation / 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.

2.2 Ablauf der Bildverarbeitung

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

3 Bilderfassung

3.1 Natürliche Bilder

Eine Beleuchtungsfunktion ist 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.

3.1.1 Sensoren

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

- **Durchleuchtung** (z.B. Röntgen, Abbildung 3.1)
  - Absorption
  - Streuung
  - Rückstreuung
  - Rauschen
- **Beleuchtung** (Abbildung 3.2)
  - Reflektion
  - Schatten
  - Oberflächen

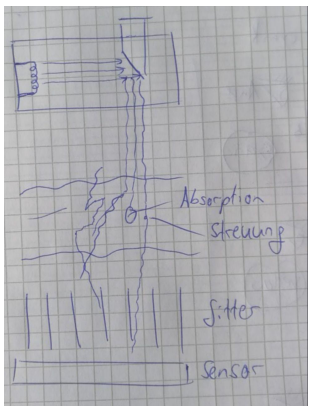


Abbildung 3.1: Sensor – Durchleuchtung (Röntgen)

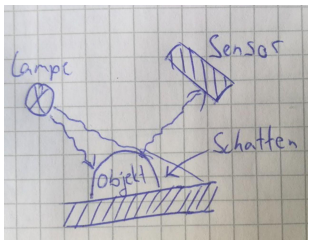


Abbildung 3.2: Sensor – Beleuchtung

3.1.2 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 3.1 zeigt die Physik hinter diesem Sensor.

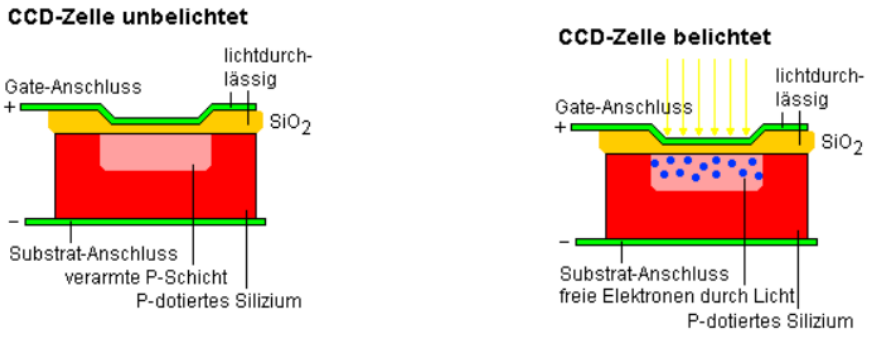


Abbildung 3.1: CCD – Physik

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 Fullframe/Interline-CCD-Sensor). Dieser dient dazu, die Informationen zu schützen während sie ausgelesen werden.

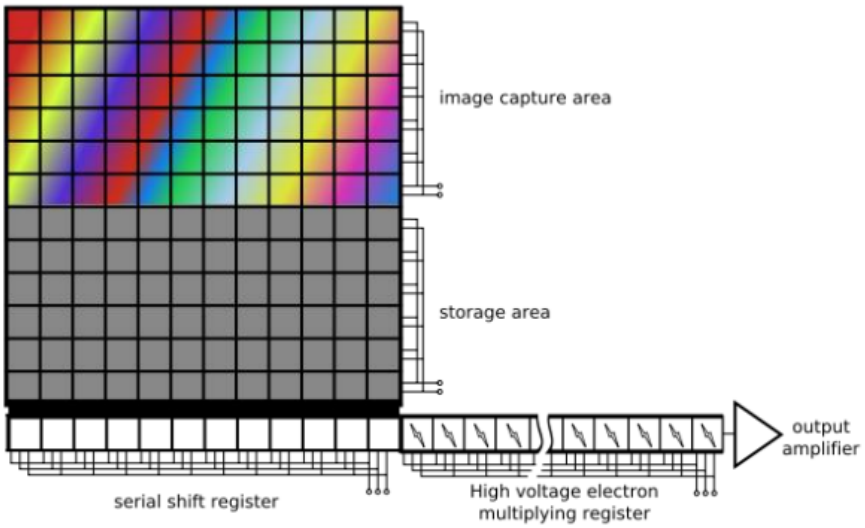


Abbildung 3.2: CCD – Gitter

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 3.3 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 Abschnitt 3.2).

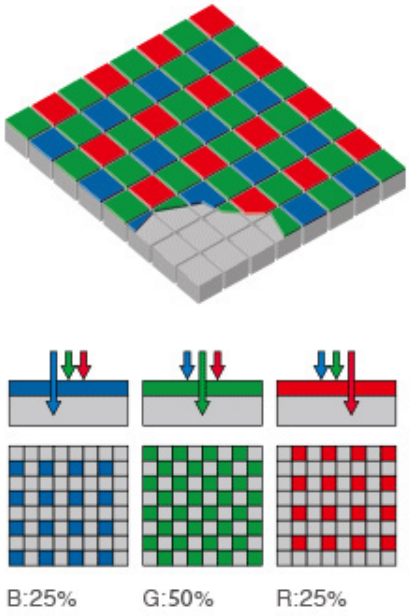


Abbildung 3.3: CCD – Bayer Maske

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 3.4 gezeigt wird. Hierbei gibt es drei Sensoren, die jeweils für eine der Farben rot, blau und grün zuständig sind. Eine Alternative ist CMOS, bei dem die Eindringtiefe für den Farbwert verwendet wird.

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

3.2 Farbsysteme

Wir unterscheiden zwischen zwischen RGB und CMYK. Setzt man rot, grün und blau auf das Maximum, ergibt sich weiß (Emission). Setzt man Cyan, Magenta und Gelb auf das Maximum, ergibt sich schwarz (Absorption).

Für eine Umrechnungsformel zwischen RGB und CMYK siehe
`https://www.rapidtables.com/convert/color/rgb-to-cmyk.html`

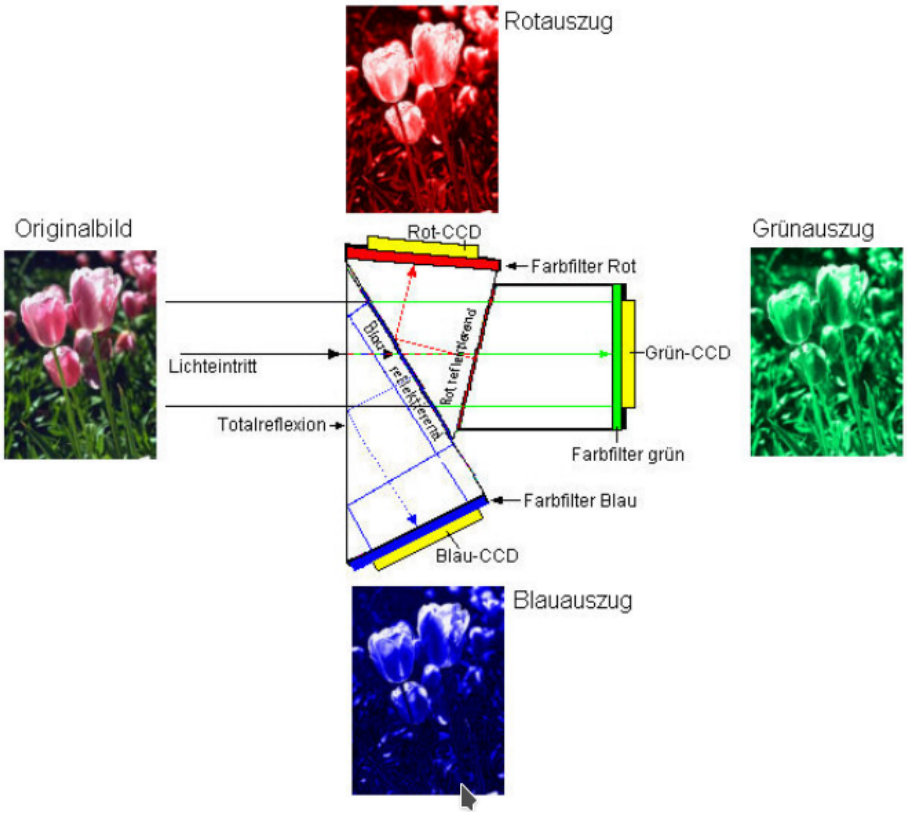


Abbildung 3.4: 3-CCD

4 Bildvorverarbeitung

4.1 Codierung und Kompression

4.1.1 LZW / LZ78

Der Lempel-Ziv-Welch (LZW)-Algorithmus ist die bekannteste Form der LZ78-Kompression. Listing 4.1 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 4.1: 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.

4.1.2 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 4.1: 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`

4.1.3 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 4.2).

4.2 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.

1. Farbmodellumrechnung von RGB -> YCbCr
2. Tiefpassfilterung der Farbdifferenzsignale
3. Blockbildung
4. Diskrete Kosinustransformation
5. Quantifizierung

6. Umsortierung und Differenzkodierung des Gleichanteils
7. Lauflängenkodierung (RLC)
8. Huffmankodierung

4.2.1 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 4.2 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 4.3b 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 4.3c veranschaulicht dies.

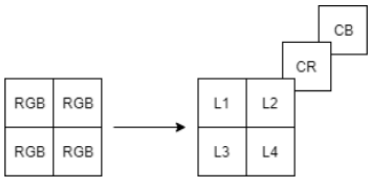


Abbildung 4.2:
RGB Alternative – YCbCr

.....

Berechnung Farbwerte

4.2.2 Diskrete Kosinustransformation

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

DCT, Formeln...

4.2.3 Quantisierung

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

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

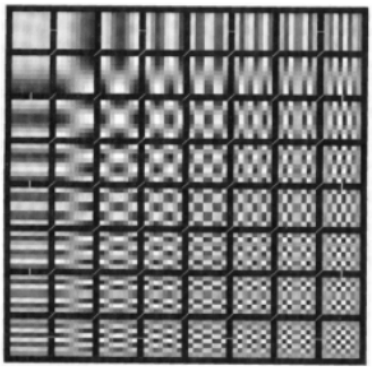
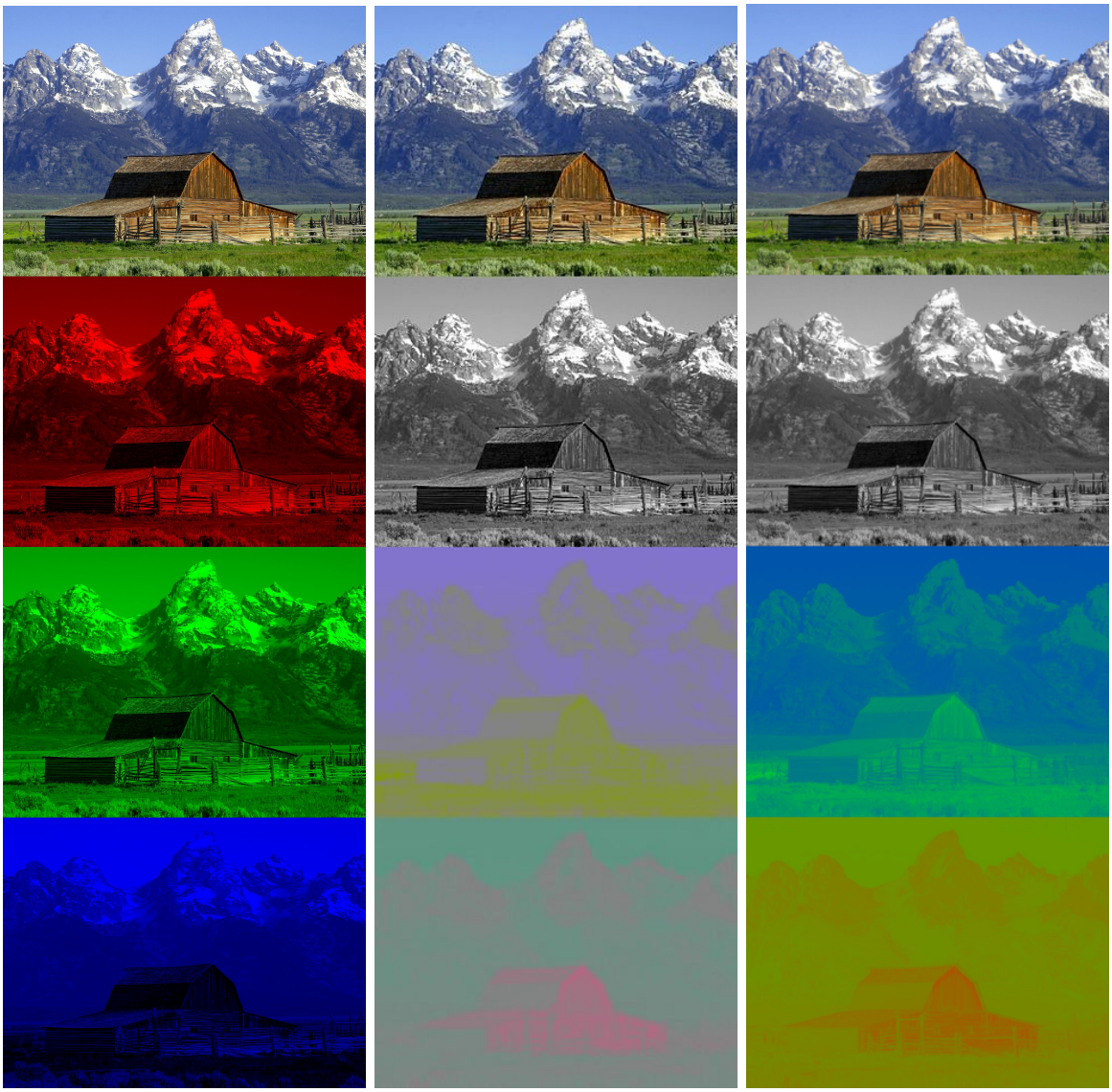


Abbildung 4.4:
Diskrete Kosinustransformation (DCT)

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.



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

Abbildung 4.3: Farbräume visualisiert

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

$$\hat{G}_{uv} = 31$$
$$q_{uv} = 8$$
$$INT\lfloor \frac{G_{uv}}{q_{uv}} \rfloor = 3 \rightarrow Dekodierung : 3 \times 8 = 24 = G_{uv}$$

4.2.4 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 4.5 stellt dies dar.

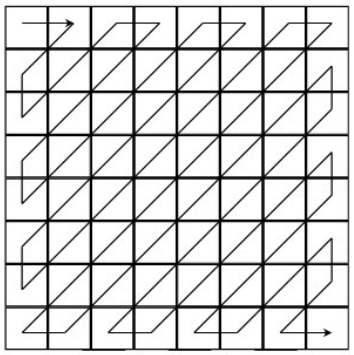


Abbildung 4.5: Serialisierung der Matrix in der Zig-Zag-Ordnung

4.2.5 RLC (Lauflängenkodierung)

Anschließend folgt eine Lauflängenkodierung der serialisierten Matrix. Das funktioniert aufgrund der langen Nullketten sehr gut. Siehe Unterabschnitt 4.1.3 und die vorhergehende Matrix.

4.2.6 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 Running Length Codierung (RLC) und Huffman gut

Wie meinte er das?

4.2.7 Kantenartefakte bei JPEG

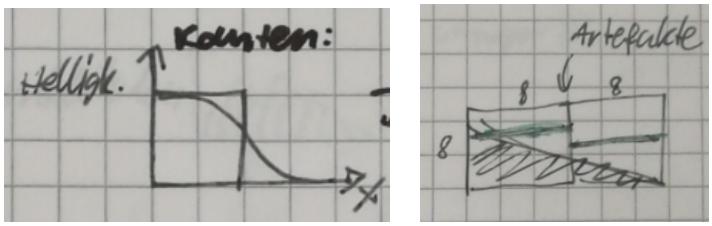


Abbildung 4.6: JPEG Artefakte

4.2.8 JPEG 2000

JPEG 2000 ist eine erweiterte Version von JPEG. Sie verspricht eine bessere Qualität bei geringen Bitraten. Anstatt der DCT werden Wavelets verwendet und anstatt der Huffman Kodierung wird eine arithmetische verwendet. Es gibt keine feste Blockgröße für die Wavelet-Transformation. Trotz dieser Unterschiede ist die Bildqualität nicht zwingend besser als bei JPEG. Sie macht sich nur bei starker Kompression bemerkbar.

4.2.9 Moire Effekt bei JPEG

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

4.3 Analogien eines Bildes

4.3.1 Histogramm

Man kann in einem Histogramm den Mittelwert der Grauwerte, die Anzahl der Einträge, die Streuung, die Anzahl der vorkommenden Helligkeitsklassen und die Spannweite ablesen.

Hinweis

Zu sagen, ein „Histogramm zeige die Häufigkeiten von Grauwerten“ ist falsch! Dies wurde von Herrn Lausen *mehrmals* in aller Deutlichkeit erwähnt. Korrekt ist: „Ein Histogramm ist die Darstellung der Besetzungshäufigkeit von Helligkeitsklassen in einem Bild.“

Histogramm bei RGB

Die einzelnen Farbkanäle können auch dazu verwendet werden, Objekte aufgrund ihrer Farbe zu identifizieren. Somit ergibt sich für jede der drei Farben ein Histogramm, wie in Abbildung 4.7 zu sehen ist.

4.3.2 Gammakorrektur

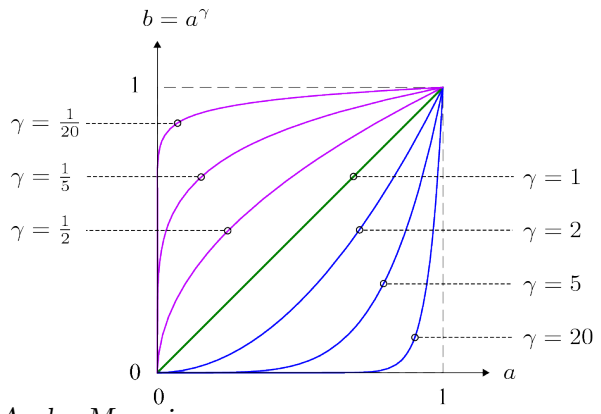


Abbildung 4.8: Gammakorrektur

Das menschliche Auge sieht logarithmisch. Dies bedeutet, dass eine Verdopplung der Helligkeit nicht gleich einer Verdopplung der wahrgenommenen Helligkeit entspricht.

Die Gamma-Korrektur spreizt die Grauwerte in dem oberen oder unteren Wertebereich und staucht sie dafür in dem jeweils anderen. In

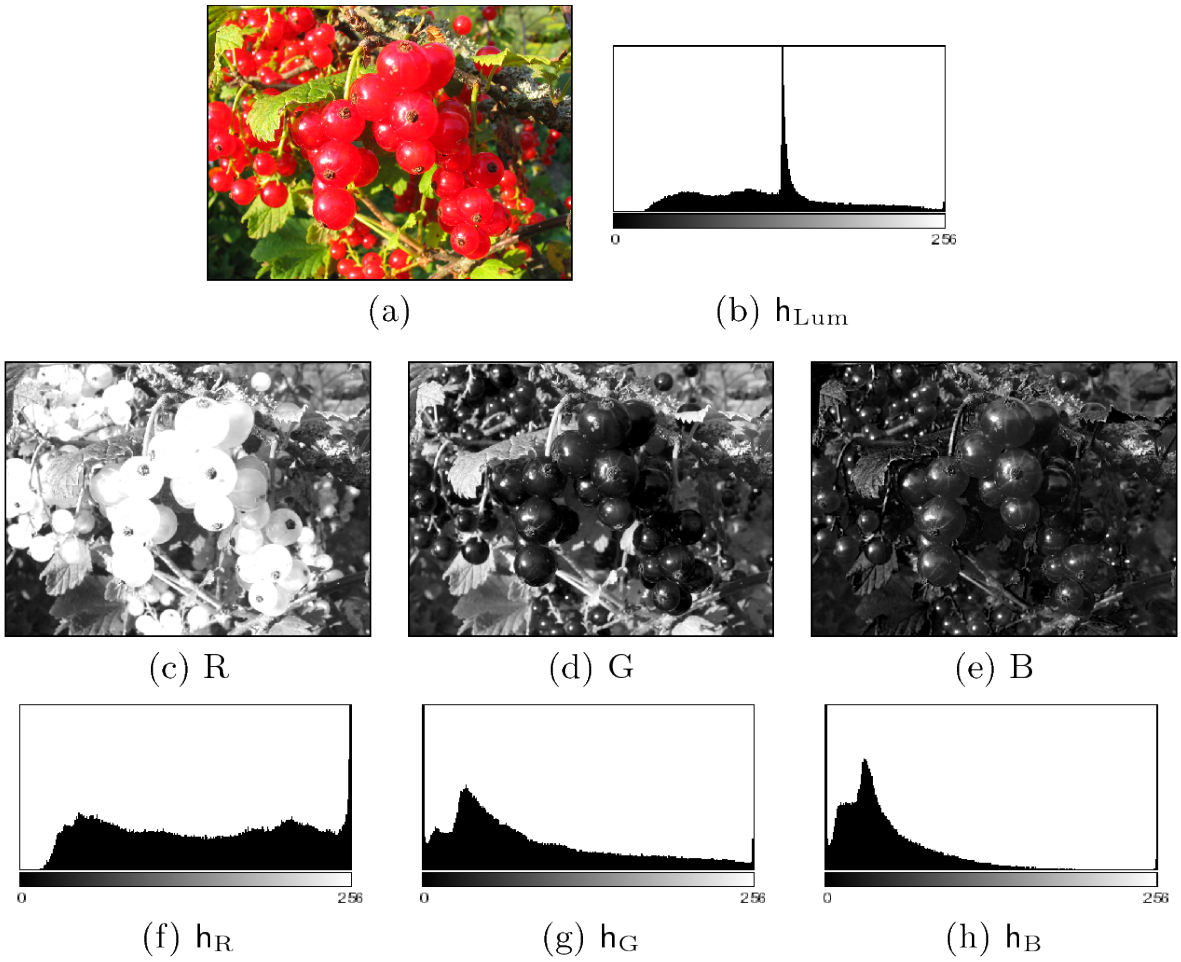


Abbildung 4.7: Histogramm RGB

Abbildung 4.8 ist eine solche Korrektur abgebildet. Die x-Achse repräsentiert das Originalbild I und die y-Achse das veränderte $I' = I^\gamma$. Bei $\gamma < 1$ wird das Bild heller, bei $\gamma > 1$ wird es dunkler. Diese Umrechnung funktioniert nur, wenn die Werte im Bereich $[0, 1]$ liegen. Somit besteht die vollständige Umrechnung aus:

$$I'_\gamma = INT[(\frac{I}{I_{max}})^\gamma \cdot I_{max}]$$

4.3.3 Kontrast

Als Kontrast wird die Spannweite zwischen maximalem und minimalem Helligkeitswert bezeichnet. Kontrast kann man „spreizen“, d. h. die Minimal- und Maximalwerte heraufsetzen. Abbildung 4.9 zeigt ein Bild mit unterschiedlichen Kontrasten (vor und nach Spreizung). Sei $[a_{min}, a_{max}]$ der zu spreizende Be-

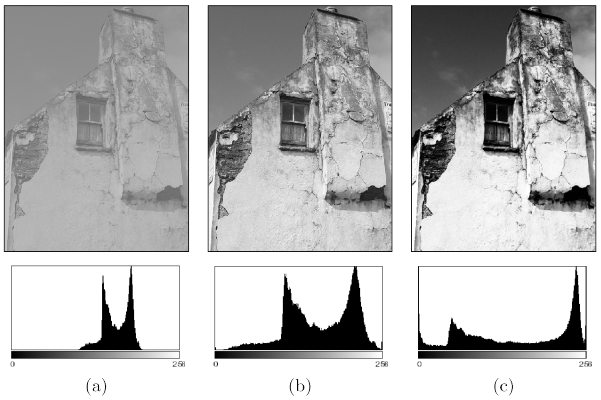


Abbildung 4.9: Kontrastspreizung

reich, so lässt sich das neue Bild berechnen mit:

$$I' = f(I) = INT[\frac{I - a_{min}}{a_{max} - a_{min}} \cdot I_{max}]$$

Diese Operation ist eine homogene Punktoperation (PO) und kann effizient mit einer Look-Up-Table (LUT) berechnet werden.

4.3.4 Dynamik

Die Dynamik eines Bildes gibt an, wie viele Grauwertklassen besetzt sind. Obwohl der Kontrast groß ist ($I_{max} - I_{min}$) kann die Dynamik variieren. In Abbildung 4.10 sind im rechten Bild nur wenige Grauwerte besetzt. Man sagt, das Bild hat wenig Dynamik. Natürliche Bilder haben meist eine hohe Dynamik und keine „Lücken“. An diesen kann man u. a. bearbeitete Bilder erkennen.

4.3.5 Einfluss JPEG-Konvertierung

Eine Kompression kann auf das Histogramm Einfluss haben. Ein Bild mit nur einem Grauwert hat nach der Dekompression plötzlich deutlich mehr Grauwerte im Histogramm. Dies liegt an der Funktionsweise von JPEG mit DCT.

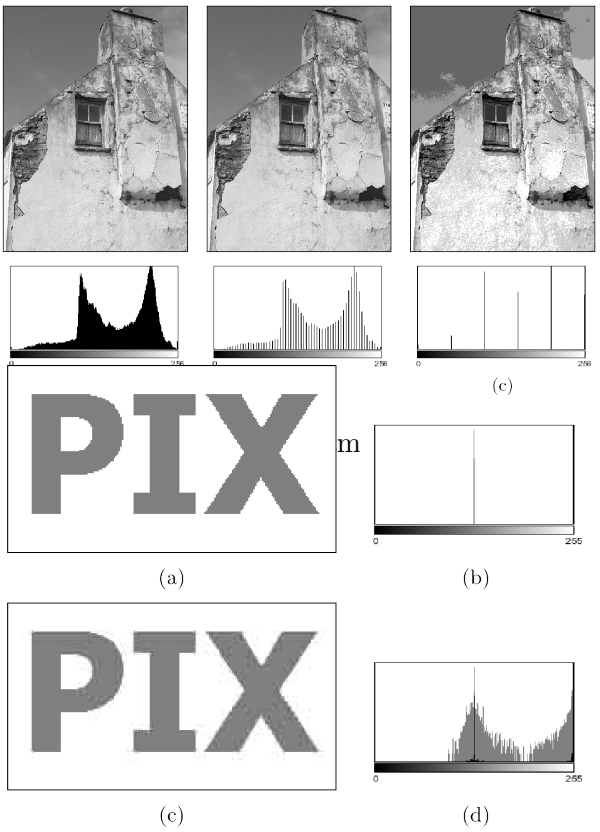


Abbildung 4.11: Kompressionseinfluss

4.4 LSI-Filter

In der Bildverarbeitung werden häufig Linear Shifting Invariant (LSI)-Filter verwendet (zu deutsch „lineare Verschiebungs-/Translationsinvarianz“). Dies bedeutet, dass der Operator auch bei Verschiebung des Systems anwenden kann. Im folgenden bezeichnet I das Bild (Image).

Faltung

2D Faltung - Kontinuierlich

$$I \cdot H(x,y) = \int_{-\infty}^{\infty} I(x - \lambda_1, y - \lambda_2)\, H(\lambda_1, \lambda_2)\, \mathrm{d}\lambda_1 \mathrm{d}\lambda_2$$

2D Faltung - Diskret

$$I \cdot H(m,n) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(m - i, n - j)\, H(i,j)$$

Kommutativität	$\mathcal{H}_1\mathcal{H}_2 = \mathcal{H}_2\mathcal{H}_1$
Assoziativität	$(\mathcal{H}_1\mathcal{H}_2)\mathcal{H}_3 = \mathcal{H}_1(\mathcal{H}_2\mathcal{H}_3)$
Distributivität	$(\mathcal{H}_1 + \mathcal{H}_2)\cdot\mathcal{H}_3 = \mathcal{H}_1\mathcal{H}_3 + \mathcal{H}_2\mathcal{H} * 3$

Sei ${}^{mn}\mathcal{S}$ der Verschiebungsoperator. Die Verschiebungsinvarianz wird dargestellt durch:

$$\mathcal{H}\,{}^{mn}\mathcal{S} = {}^{mn}\mathcal{S}\mathcal{H}$$
$${}^{mn}\mathcal{S}\,g_{m'n'} = g_{m'-m,n'-n}$$

4.4.1 Gaußoperator zum Weichzeichnen

Für die linke, große Matrix ergeben sich 25 Multiplikationen und 24 Additionen. Teilt man diese Matrix jedoch in zwei kleinere auf, so ergeben sich zwei mal 5 Mult. + 4 Add. = 10 Mult. + 8 Add.

$[1\ 4\ 6\ 4\ 1]$ lässt sich weiter unterteilen nach:
$[1\ 1]^4 \cdot (\frac{1}{2})^4$.

$$\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix}$$

Abbildung 4.12:
Gaußoperator zum Weichzeichnen

Erklärung

$$I' = H \cdot I \qquad H = \begin{bmatrix} a & b & c \\ d & e. & f \\ g & h & i \end{bmatrix} \qquad I = \begin{bmatrix} \dots & & \dots \\ & \begin{bmatrix} & \cdot & \\ & & a \end{bmatrix} & \\ \dots & & \dots \end{bmatrix}$$

$$\begin{aligned} I'(m,n) &= H \cdot I(m,n) \\ I'_{mn} &= \sum_{j=-1}^{+1} \sum_{+1}^{i=-1} H(i,j) \cdot I(m-i, n-j) \\ &= (a \cdot I_{m+1,n+1} + b \cdot I_{m,n+1} + c \cdot I_{m-1,n+1} + d \cdot I_{m+1,n} + e \cdot I_{m,n} + f \cdot I_{m-1,n} + \\ &\quad g \cdot I_{m+1,n-1} + h \cdot I_{m,n-1} + i \cdot I_{m-1,n-1}) \end{aligned}$$

Beispiel: 1D-Faltung

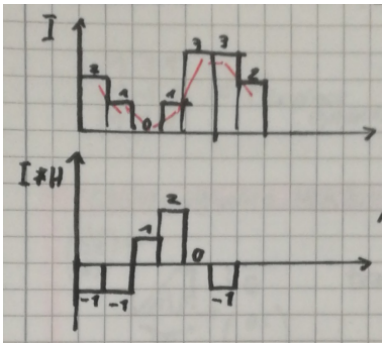


Abbildung 4.13: 1D Faltung

Sei H der Faltungskern $[1 \ -1]$ (gedacht: $[1 \ -1 \ 0]$). Abbildung 4.13 zeigt diesen Kern angewendet auf ein Bild I. Anhand der resultierenden Grafik wird deutlich, dass es sich bei dem Faltungskern um den Ableitungsoperator handelt. Leiten wir das Bild an einem Punkt ab, so ergibt sich aus der Berechnung eben dieser Faltungskern.

$$\begin{aligned}\frac{\mathrm{d}f}{\mathrm{d}x} &= \lim_{h\to 0}\frac{f(x+h)-f(x)}{h}\\ \frac{\mathrm{d}I}{\mathrm{d}m} &= \frac{f(m+1)-f(m)}{1}\\ &= I(m+1)-I(m)\\ &= \mathbf{1}\cdot I_{m+1}+(\mathbf{-1})\cdot I_m\end{aligned}$$

Dieser Ableitungsoperator kommt jedoch „von oben“. Andersherum ergibt sich $[0 \ 1 \ -1]$. Besser wäre jedoch ein Operator, der beide Seiten betrachtet.

Mittelt man beide Operator, ergibt sich:

$$\begin{array}{|c|c|c|}\hline 1 & -1 & 0 \\ \hline 0 & 1 & -1 \\ \hline \end{array} = [1 \ 0 \ -1]\cdot\tfrac{1}{2}.$$

Da nach der Transformation jedoch sowieso kein echtes Bild herauskommt, kann man das Teilen durch zwei weglassen.

5 Abkürzungsverzeichnis

dpi dots per inch
CCD Charge Coupled Device . . . 3
DCT Diskrete Kosinustransformation . . . 9
JPEG Joint Photographic Experts Group . . . 8
LSI Linear Shifting Invariant . . . 14
LZW Lempel-Ziv-Welch . . . 7
LUT Look-Up-Table . . . 14
PO Punktoperation . . . 14
RLC Running Length Codierung . . . 11

Abbildungsverzeichnis

Tabellenverzeichnis

Listingsverzeichnis

Stichwortverzeichnis