Punktoperationen II und Einführung in Filterung

Industrielle Bildverarbeitung, Vorlesung No. 4[1]

M. O. Franz

[1] falls nicht anders vermerkt, sind die Abbildungen entnommen aus Burger & Burge, 2005.

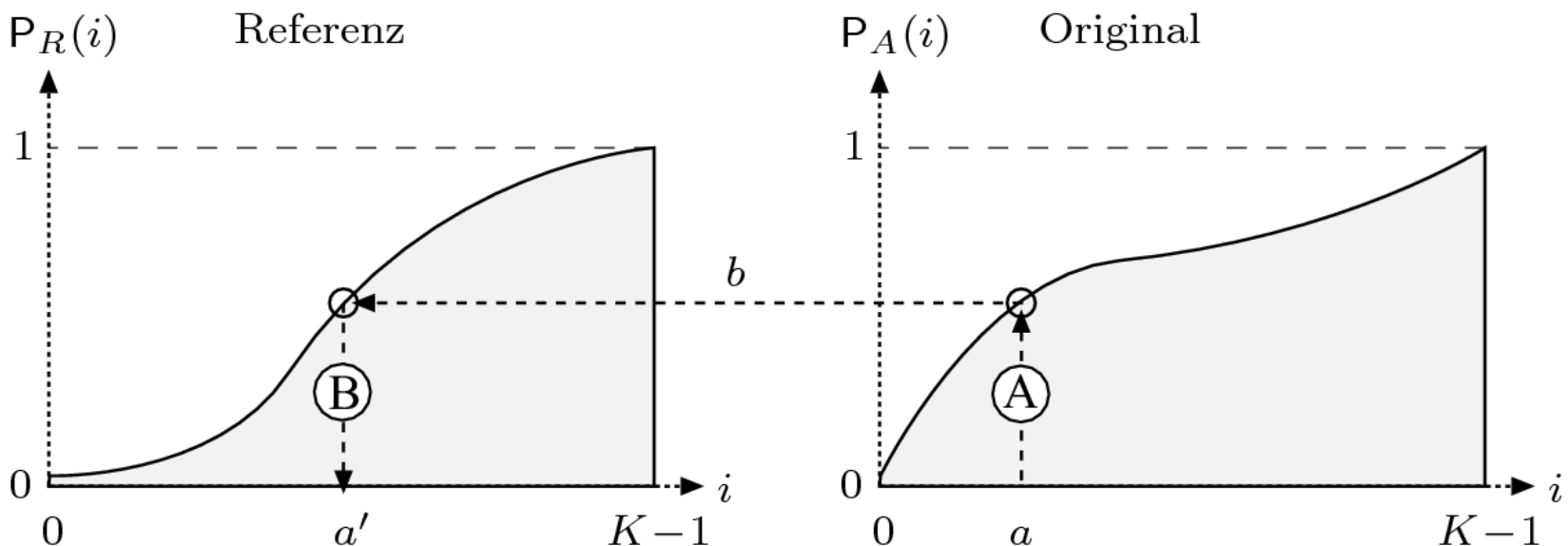
Überblick

Übersicht

Histogrammanpassung

- Histogrammausgleich führt zu einem gleichverteilten Histogramm.
- Gleichverteilung maximiert die in einem Grauwertintervall $[0, K-1]$ darstellbare Information.
- Aber: Bilder sehen unnatürlich aus, da die meisten natürlichen Bilder eher gaußverteilte Histogramme haben.

Daher Histogrammanpassung: Anpassung des Histogramms an eine Referenzverteilung.



Prinzip der Histogrammanpassung

Ziel: Modifiziere Ausgangsbild I_A durch eine homogene Punktoperation so, daß seine Verteilungsfunktion P_A möglichst gut mit P_R eines Referenzbildes I_R übereinstimmt.

2 Schritte:

1. Histogramm wird durch linearen Histogrammausgleich in eine Gleichverteilung überführt:

$$I_A \rightarrow I_{A'} : \quad a' = P_A(a)$$

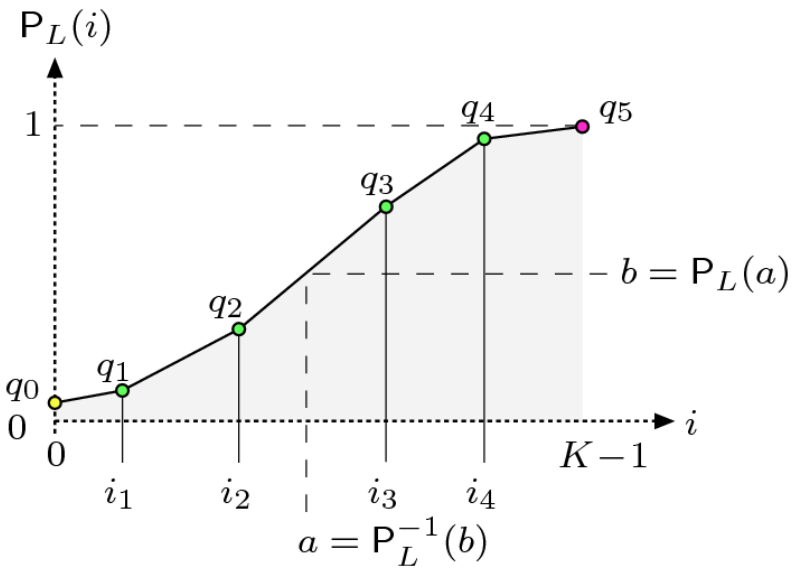
2. Das Resultat wird über die Inverse $P_R(a)^{-1}$ der Referenzverteilung transformiert:

$$I_{A'} \rightarrow I_{A''} : \quad a'' = P_R(a')^{-1}$$

Insgesamt also:

$$I_A \rightarrow I_{A''} : \quad a'' = P_R^{-1}(P_A(a))$$

Stückweise lineare Referenzverteilung



Zwischen N vorgegebene Stützstellen (i_j, q_j) wird linear interpoliert:

$$\mathsf{P}_L(i) = \begin{cases} q_m + (i - i_m) \cdot \frac{(q_{m+1} - q_m)}{(i_{m+1} - i_m)} & \text{für } 0 \le i < K-1 \\ 1 & \text{für } i = K-1 \end{cases}$$

Homogene Punkttransformation:

$$\mathsf{P}_L^{-1}(b) = \begin{cases} 0 & \text{für } 0 \le b < \mathsf{P}_L(0) \\ i_n + (b - q_n) \cdot \frac{(i_{n+1} - i_n)}{(q_{n+1} - q_n)} & \text{für } \mathsf{P}_L(0) \le b < 1 \\ K-1 & \text{für } b \ge 1 \end{cases}$$

Gesamttransformation:

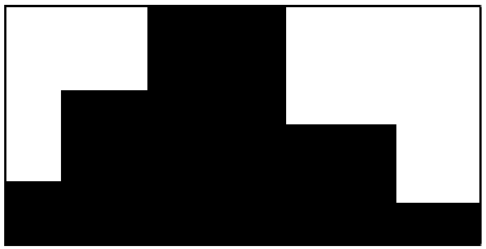
$$I_A \rightarrow I_{A''} : \quad a'' = P_L^{-1}(P_A(a))$$

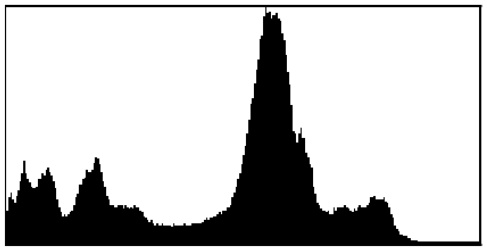
Beispiel: Histogrammanpassung an eine stückweise lineare Verteilung

Originalbild | Modifiziertes Bild

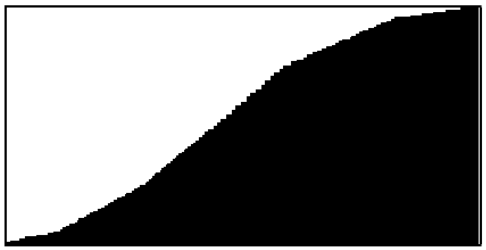
Referenzverteilung (stückweise linear)

(a) I_A (b) $I_{A'}$





(c) h_R (d) h_A (e) $\mathsf{h}_{A'}$

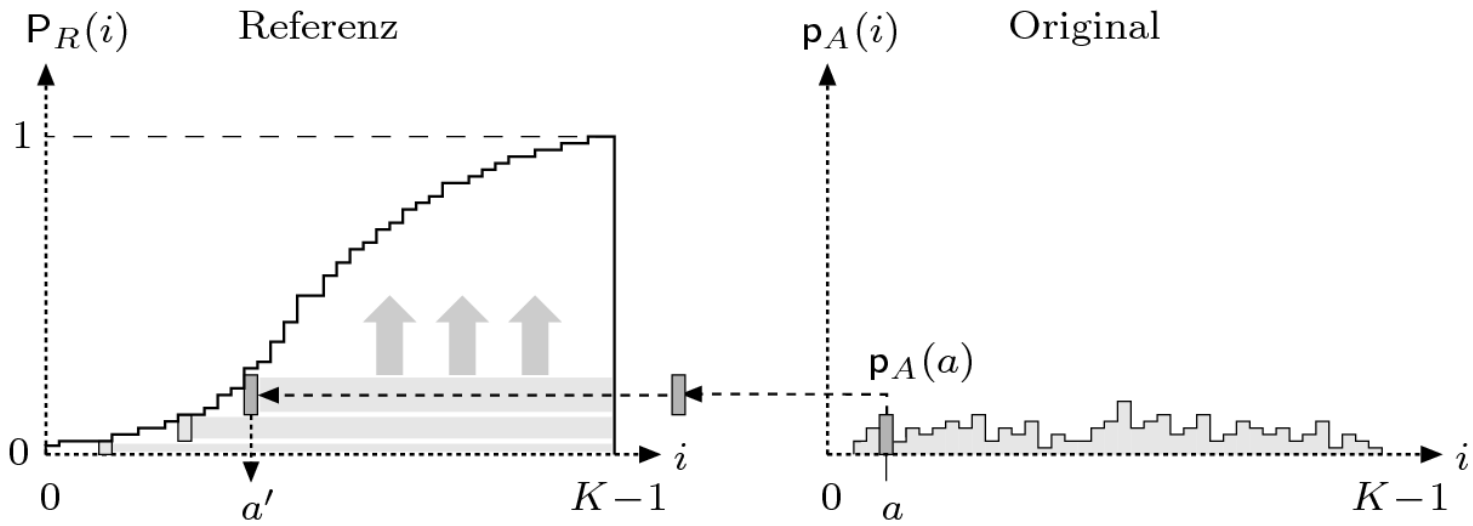


(f) P_R (g) P_A (h) $\mathsf{P}_{A'}$

Anpassung an Histogramm eines anderen Bildes

Problem: Natürliche Verteilungsfunktionen sind oft nicht invertierbar.

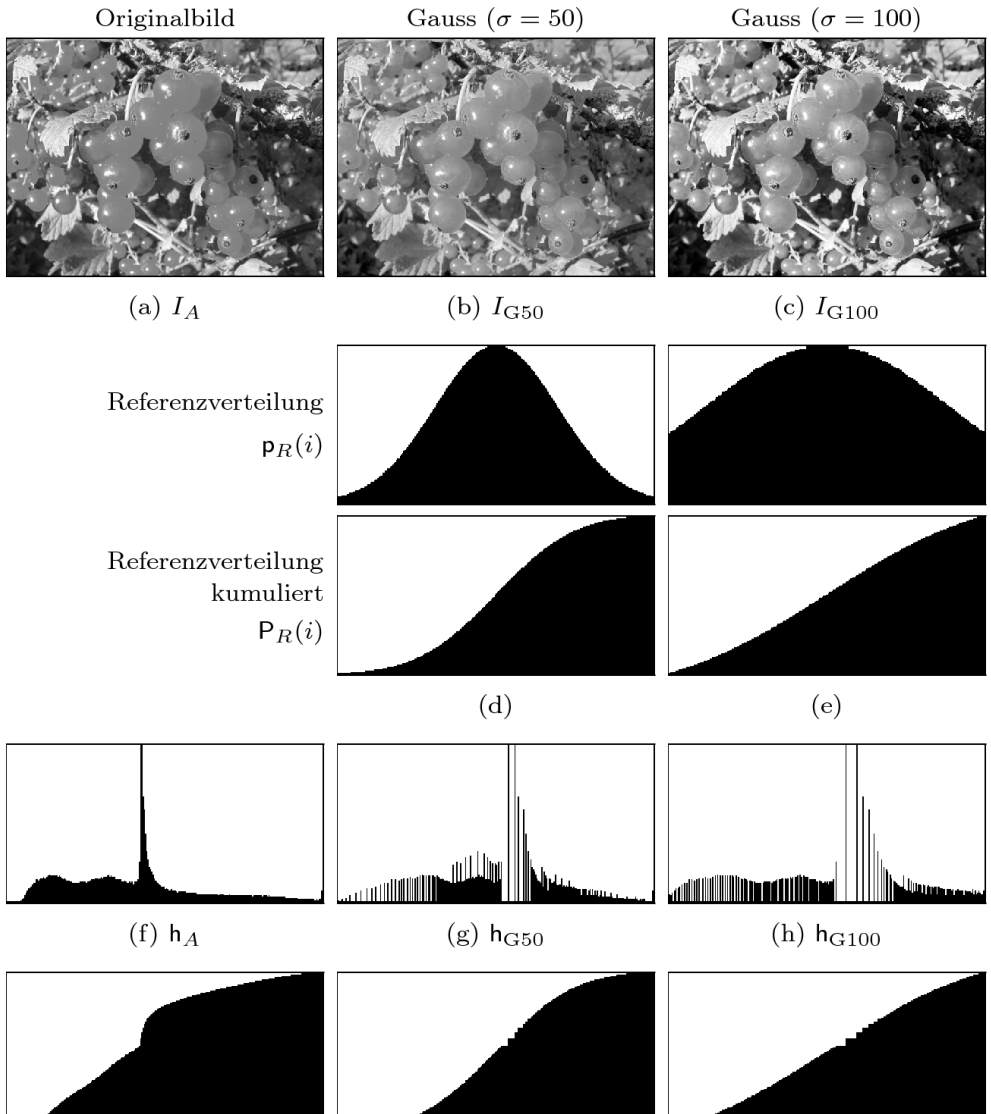
Ansatz: Schrittweises ”Ausfüllen” der Referenzverteilung $P_R(a)$



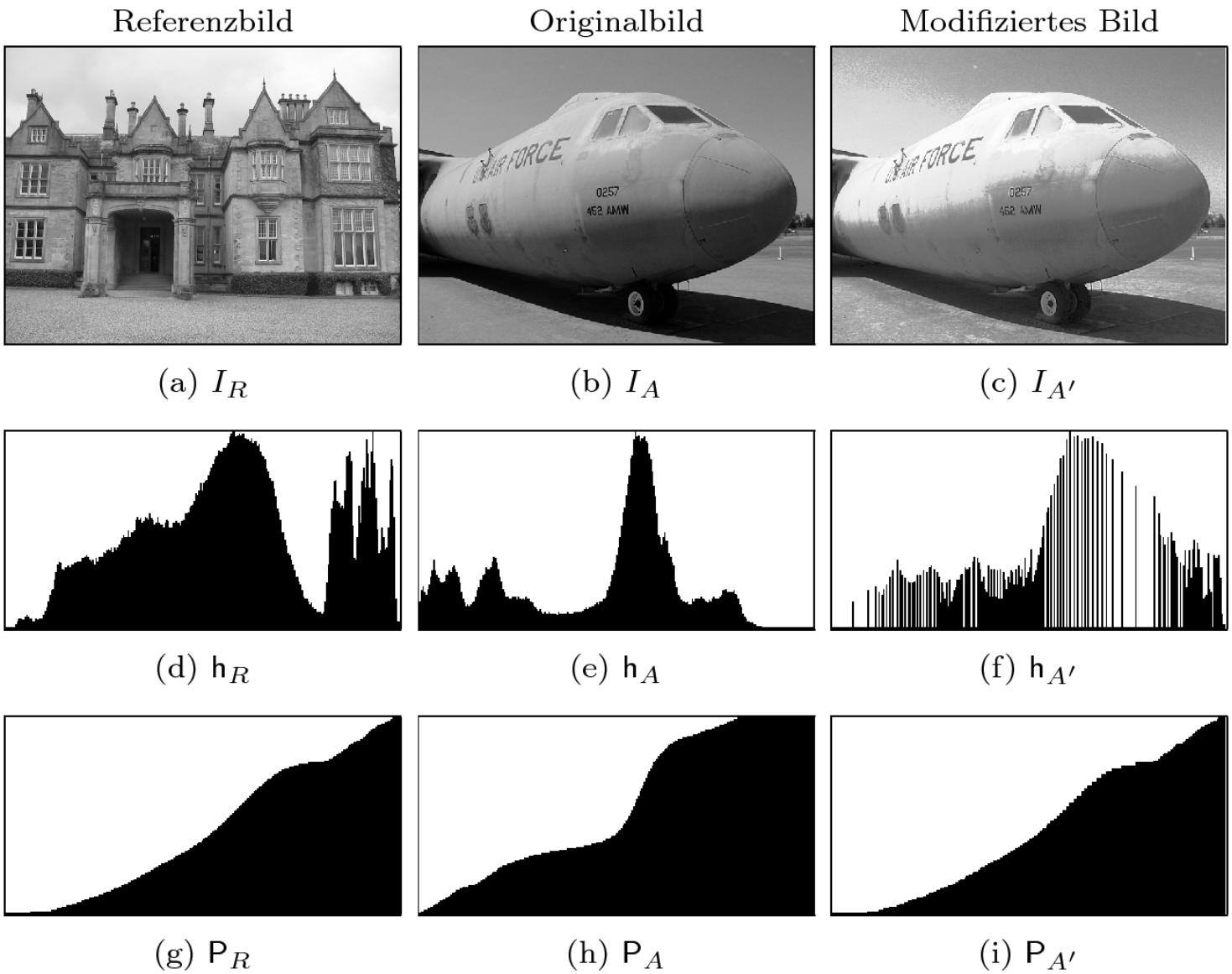
D.h.: für einen geg. Pixelwert a wird der minimale Wert a' in $P_R(a')$ gesucht, bei dem $P_A(a) \leq P_R(a')$ ist:

$$f_{hs}(a) = \min\{a' \in [0, K-1] | P_A(a) \leq P_R(a')\}$$

Beispiel: Histogrammanpassung an eine Gaußverteilung



Beispiel: Histogrammanpassung an Referenzbild

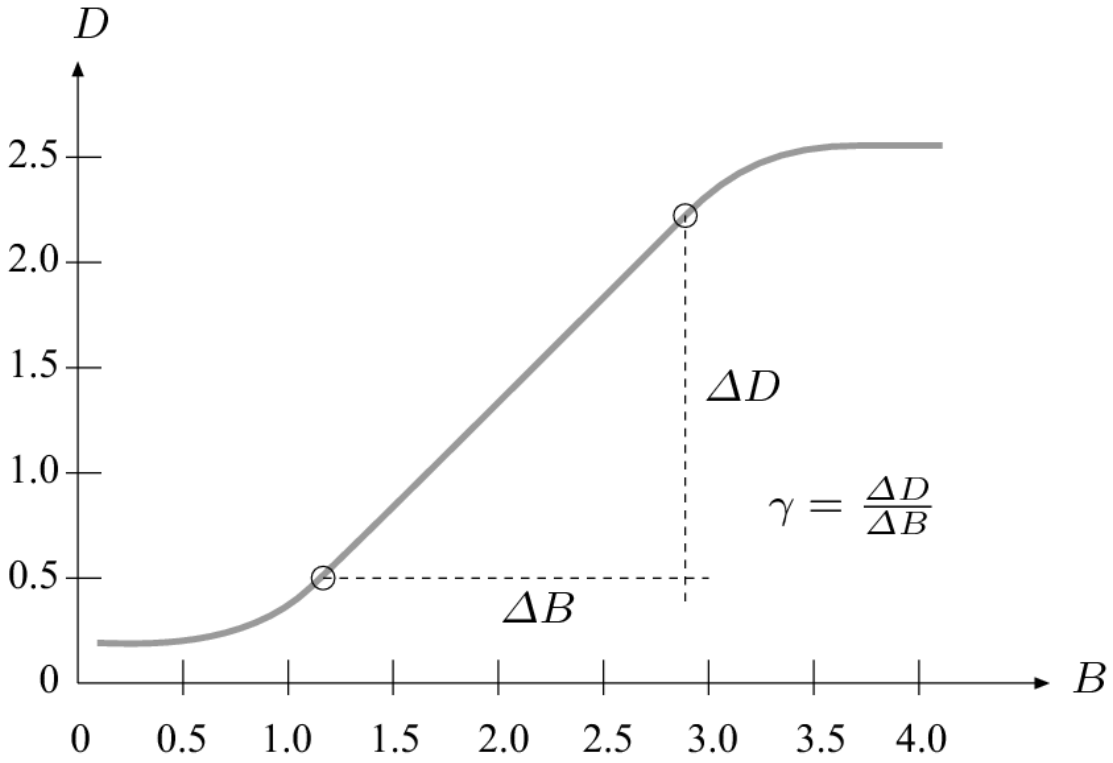


Übersicht

Gammakorrektur

- Reale Aufnahmesysteme (Kameras, Scanner,..) setzen Intensitäten nicht 1:1 in Grauwerte um. Die Abbildung von Intensitäten Φ in Grauwerte ist meist eine nichtlineare Funktion $a = F(\Phi)$.
- Ebenso setzen Ausgabegeräte (z.B. Bildschirme) Grauwerte nicht 1:1 in Helligkeiten um. Auch hier gibt es Nichtlinearitäten.
- Grundidee der Gammakorrektur: Bilder werden durch eine homogene Punktoperation so transfomiert, daß die geräteabhängige Nichtlinearität kompensiert wird.
- Nach der Korrektur entsprechen die Grauwerte nicht den absoluten Intensitäten, aber ihr relatives Verhältnis ist (idealerweise) gleich wie in der Wirklichkeit.

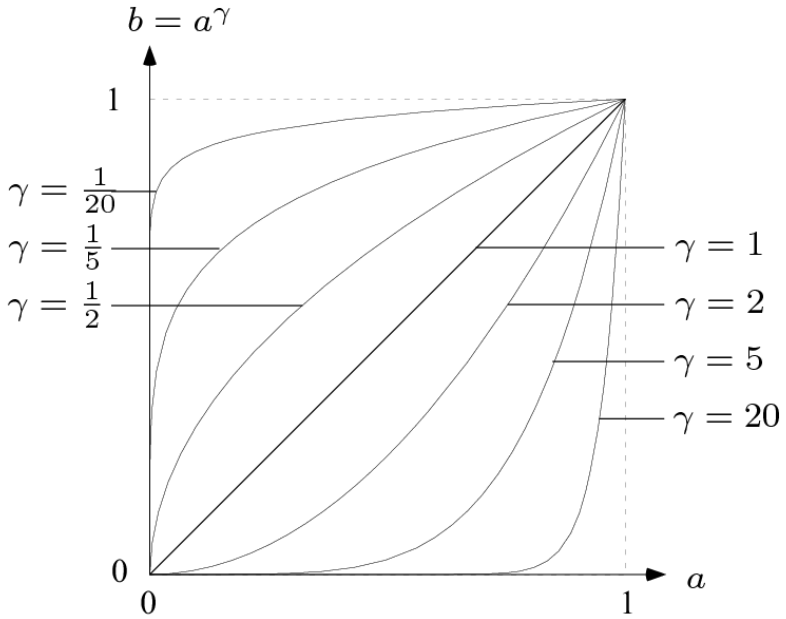
Belichtungskurve von fotografischem Film



Beleuchtung B ist in logarithmischen Einheiten, d.h. die Schwärzung des Films folgt annähernd einer logarithmischen Kurve.

Die Steigung in logarithmischen Koordinaten wird das ”Gamma” des Films genannt.

Die Gammafunktion



Gammafunktion:

$$b = f_\gamma(a) = a^\gamma \quad \gamma > 0$$

γ heißt Gammawert.

- wird nur im Bereich $[0, 1]$ eingesetzt,
- Funktionswert a^γ bleibt damit ebenfalls in $[0, 1]$.
- Für $\gamma > 1$ verläuft Kurve unterhalb der identität, für $\gamma < 1$ oberhalb.

Inverse Gammafunktion:

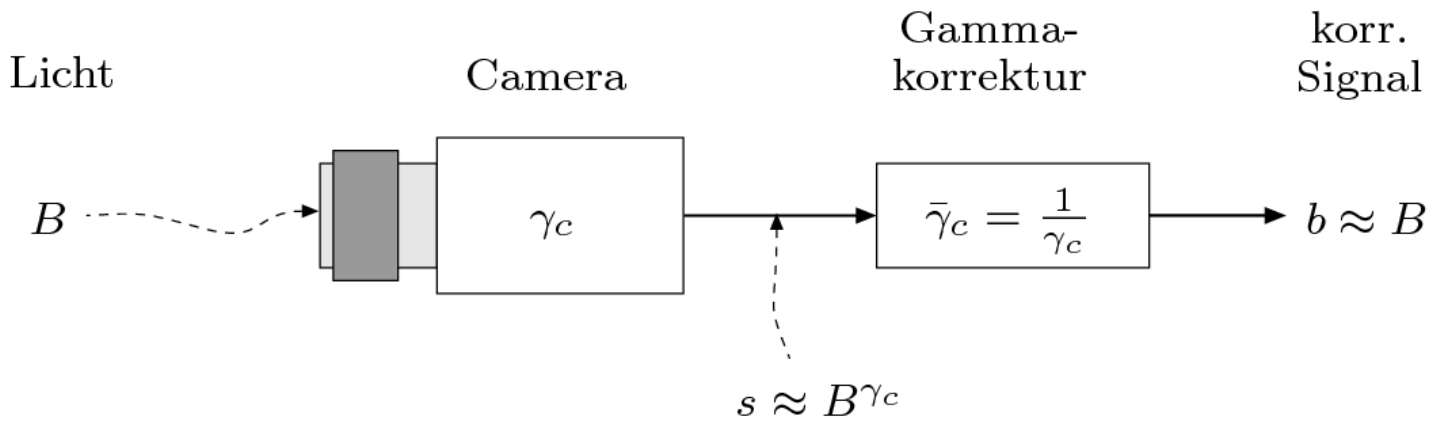
$$a = f_\gamma^{-1}(b) = b^{1/\gamma} = f_{1/\gamma}(b),$$

d.h. die Inverse der Gammafunktion ist wieder eine Gammafunktion.

Reale Gammawerte

- Konkrete Gammawerte werden von den Herstellern aufgrund von Messungen spezifiziert:
  - Röhrenmonitore: $1.8 \ldots 2.8$, typisch ist 2.4.
  - LCD: ähnlich wie Röhrenmonitore durch Voreinstellung
  - Video- und Digitalkameras: $0.4 \ldots 0.6$, Emulation des Belichtungsverhaltens von analogen Kameras
- Genormte Geräte:
  - Fernsehen: 2.2 (NTSC), 2.8 (PAL)
  - Fernsehkameras: 1/2.2 = 0.45 bei beiden
  - Internationale Norm ITU-R BT.709: 2.5 für Wiedergabegeräte, 1/1.956 = 0.51 für Kameras
- Bei Computermonitoren ist der Gammawert in bestimmten Grenzen einstellbar.
- Achtung: Modellierung über Gammafunktion ist nur eine grobe Näherung für das Transferverhalten eines Geräts. Für größere Genauigkeiten muß das Gerät mit exakt vermessenen Profilen kalibriert werden.

Anwendung der Gammakorrektur



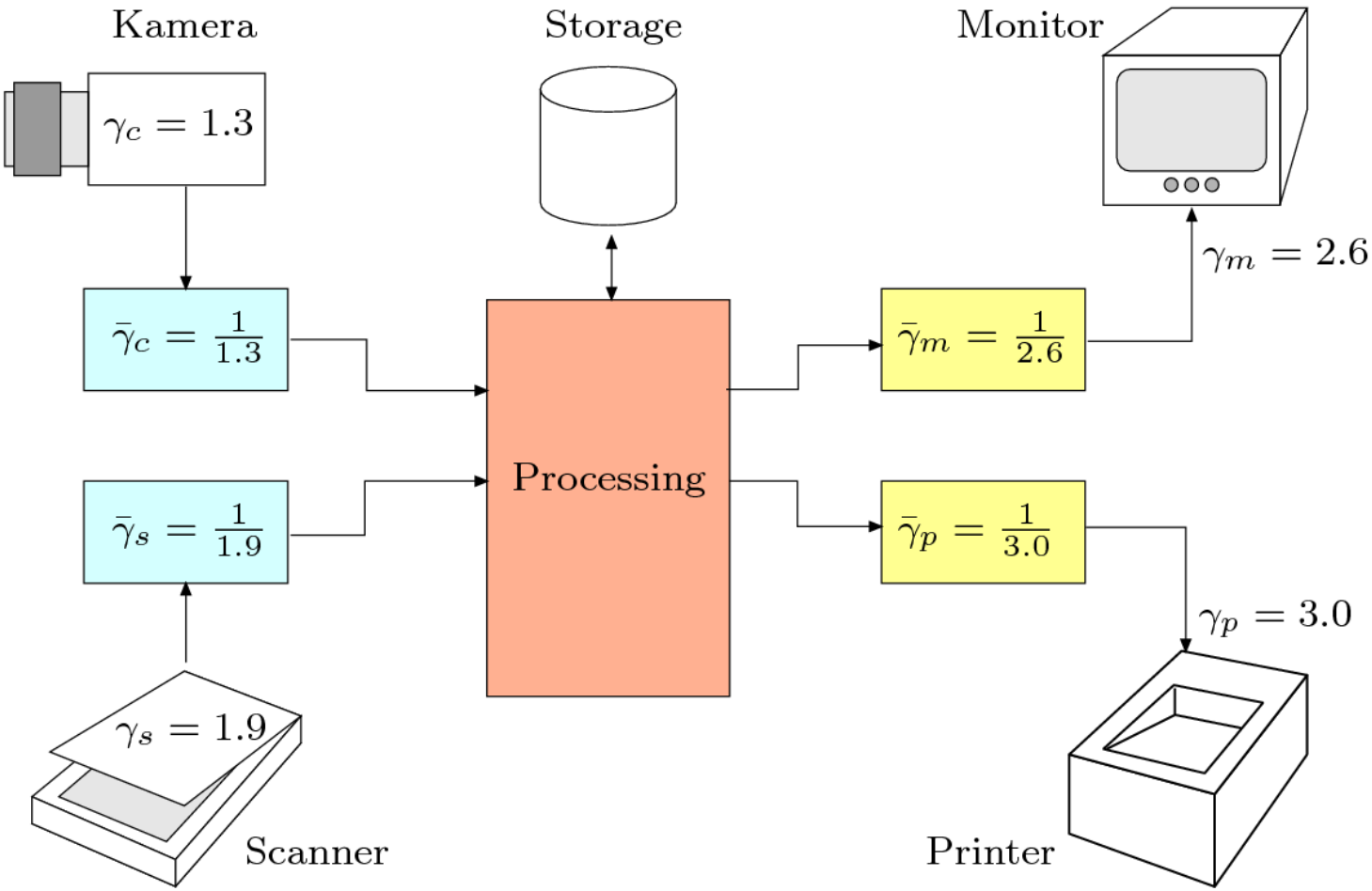
Das (idealisierte) Gerät setzt Intensitäten B in Grauwerte a nach $a = B^{\gamma}$ um. Die Gammakorrektur läuft über eine homogene Punktoperation mit dem *inversen* Gammawert

$$b = f_{1/\gamma}(a) = a^{1/\gamma}.$$

Korrigiertes Signal: $b = a^{1/\gamma} = (B^{\gamma})^{1/\gamma} = B$.
Mit Skalierung von $[0, a_{\max}]$ auf $[0, 1]$ und zurück:

$$b = f_{1/\gamma}(a) = a_{\max} \left(\frac{a}{a_{\max}} \right)^{1/\gamma}.$$

Geräteunabhängige Grauwertbilder



Modifizierte Gammakorrektur

Problem:

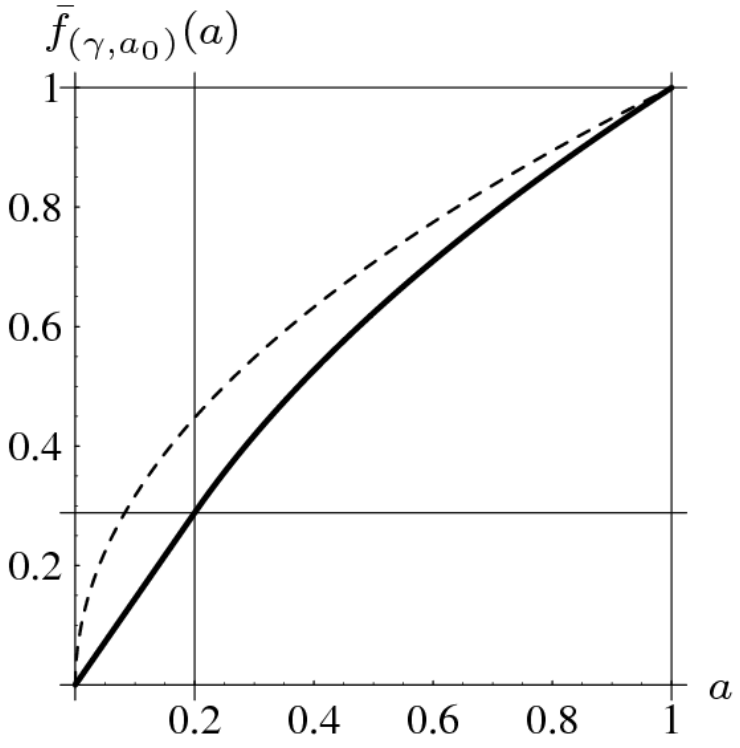
1. Für $\gamma > 1$ ist $f_\gamma(a)$ beinahe konstant, also numerisch schwer invertierbar.
2. Für $\gamma < 1$ geht die Steigung nahe an 0 gegen ∞, d.h. hohe Verstärkung des Rauschens der niedrigen Intensitätswerte

Daher: Linearer Abschnitt nahe 0, Fortsetzung mit Gammafunktion:

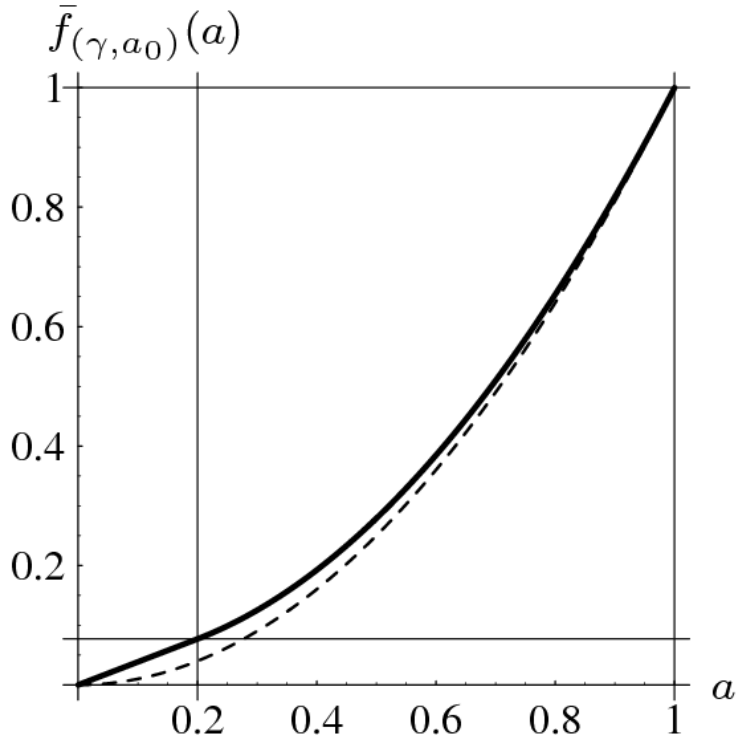
$$\bar{f}_{(\gamma,a_0)}(a) = \begin{cases} s \cdot a & \text{für } 0 \leq a \leq a_0 \\ (1+d) \cdot a^\gamma - d & \text{für } a_0 < a \leq 1 \end{cases}$$

mit $s = \frac{\gamma}{a_0(\gamma-1)+a_0^{1-\gamma}}$ und $d = \frac{1}{a_0^\gamma(\gamma-1)+1} - 1$.

Modifizierte Gammafunktion



(a) $\gamma = 0.5,\ a_0 = 0.2$

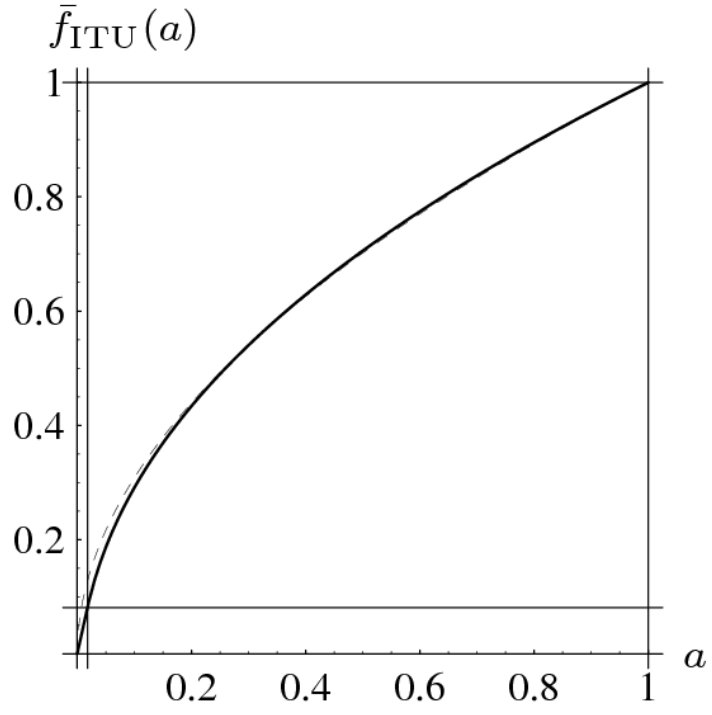


(b) $\gamma = 2.0,\ a_0 = 0.2$

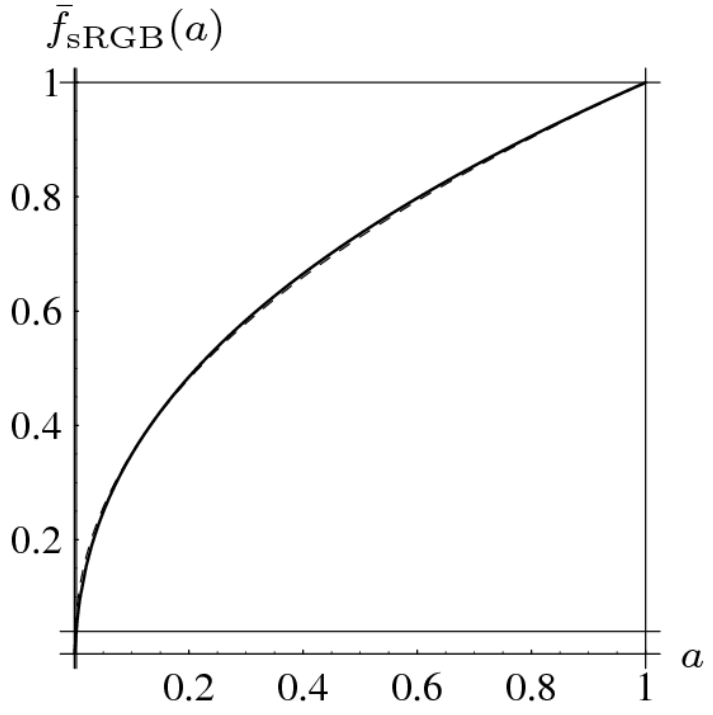
Inverse Korrektur:

$$\bar{f}^{-1}_{(\gamma,a_0)}(b) = \begin{cases} \frac{b}{s} & \text{für } 0 \leq b \leq s \cdot a_0 \\ \left(\frac{b+d}{1+d}\right)^{\frac{1}{\gamma}} & \text{für } s \cdot a_0 < b \leq 1 \end{cases}$$

Beispiel: Modifizierte Gammakorrektur in Standards



(a)



(b)

Standard	nomineller Gammawert γ	a_0	s	d	effektiver Gammawert γ_{eff}
ITU-R BT.709	$1/2.222 \approx 0.450$	0.01800	4.5068	0.09915	$1/1.956 \approx 0.511$
sRGB	$1/2.400 \approx 0.417$	0.00304	12.9231	0.05500	$1/2.200 \approx 0.455$

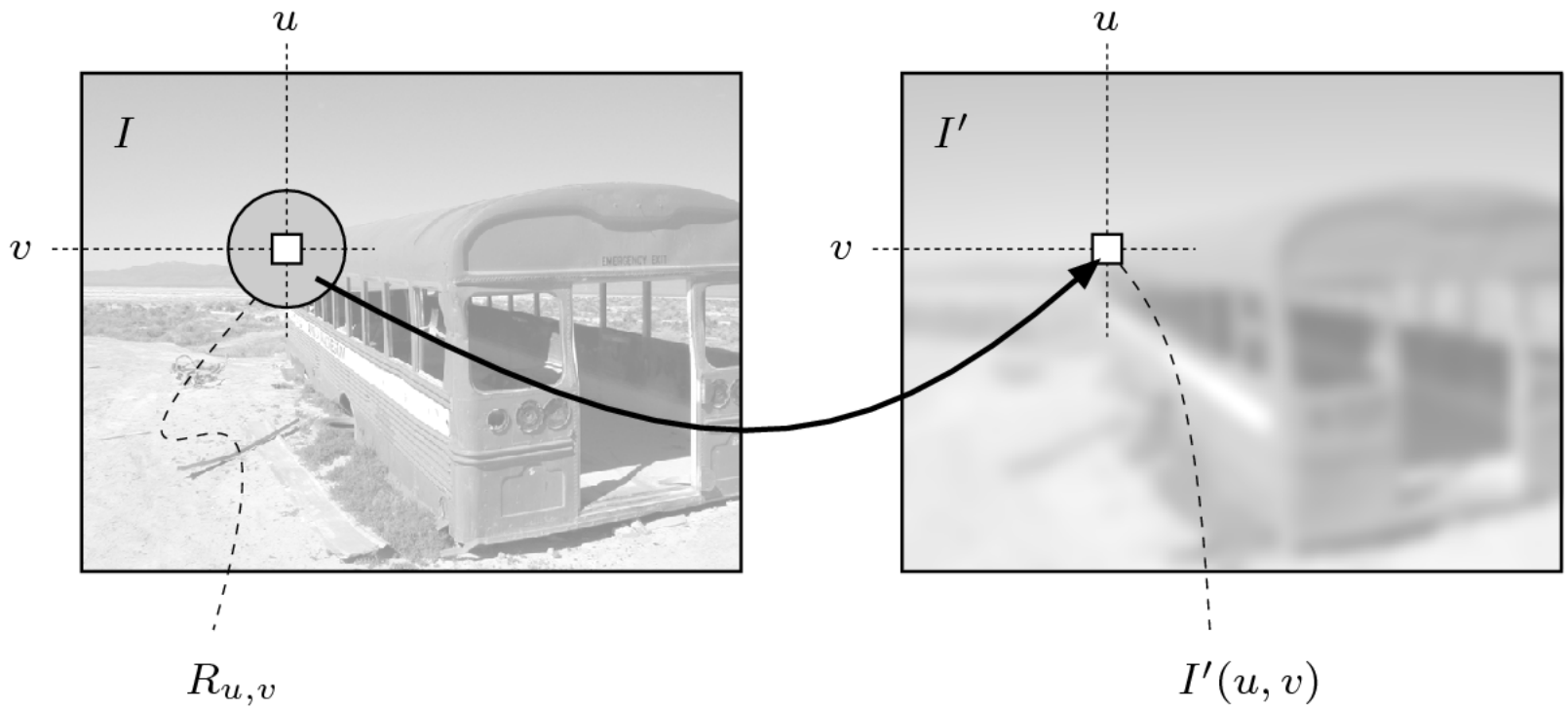
Übersicht

Filterung

Beispiel: Glättung

- Mit Punktoperationen allein läßt sich keine Glättung oder Schärfung eines Bildes erzielen $\Rightarrow$ Filterung notwendig,
- Auch Filterung ändert nicht die Bildgeometrie, d.h. die Position der Pixel bleibt nach Filterung unverändert.

Beispiel: Glättungsfilter (1)



Idee: Ersetze jeden Pixel durch den Durchschnitt seiner Nachbarschaft $p_1, p_2 \ldots p_9$:

$$I'(u,v) \leftarrow \frac{1}{9}\sum_{i=1}^{9} p_i$$

Beispiel: Glättungsfilter (2)

In relativen Bildkoordinaten:

$$I'(u,v) \leftarrow \frac{1}{9} \left[\begin{array}{lll} I(u-1,v-1) & + I(u,v-1) + I(u+1,v-1) + \\ I(u-1,v) & + I(u,v) + I(u+1,v) + \\ I(u-1,v+1) & + I(u,v+1) + I(u+1,v+1) \end{array} \right]$$

Filtermerkmale:

- Ergebnis wird nicht aus einem einzigen Pixel berechnet, sondern aus einer Menge von Pixeln.
- Die Koordinaten der Quellpixel habe eine feste relative Position zum Zielpixel und bilden i.A. eine zusammenhängende Region.

Parameter:

- Größe der Filterregion
- Form der Filterregion
- Gewichtung der Quellpixel (konstant oder ortsabhängig)

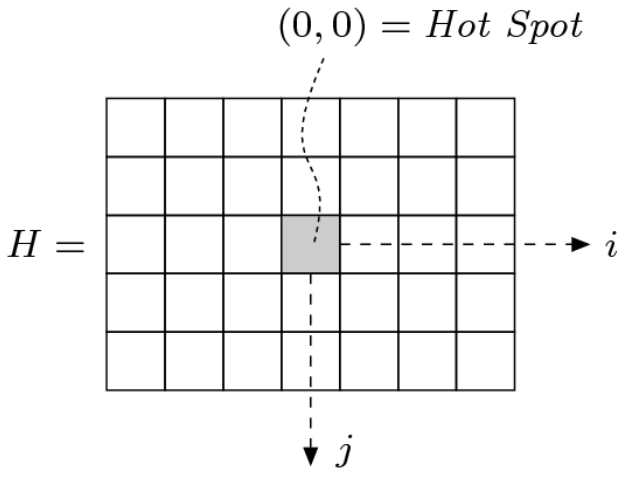
Lineare Filter

Lineare Filter: Wert des Zielpixels wird als gewichtete Summe der Quellpixel berechnet.

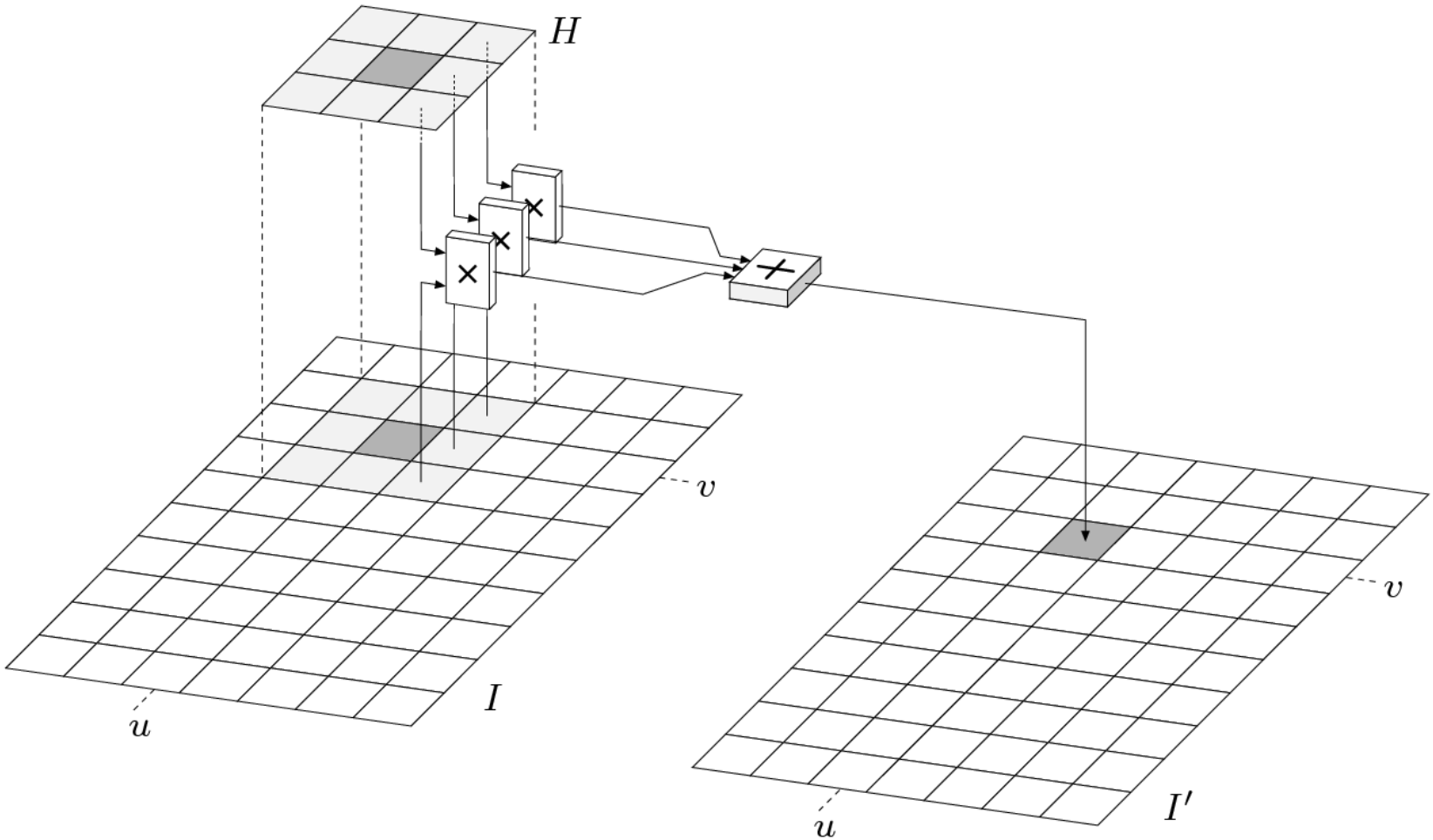
Größe und Form der Filterregion und Gewichte des Filter werden durch eine Matrix von *Filterkoeffizienten* spezifiziert, der Filtermatrix H_{ij} oder Filtermaske, z.B.

$$H(i,j) = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Die Filtermatrix ist - wie ein Bild - eine diskrete zweidimensionale Funktion. Koordinaten werden meist relativ zum Zentrum angegeben (”hot spot”).

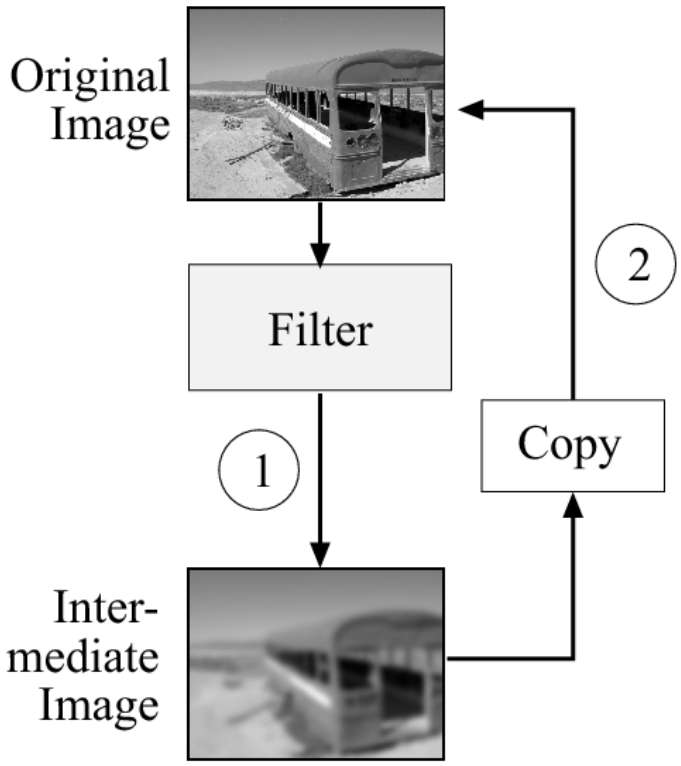


Anwendung eines Filters

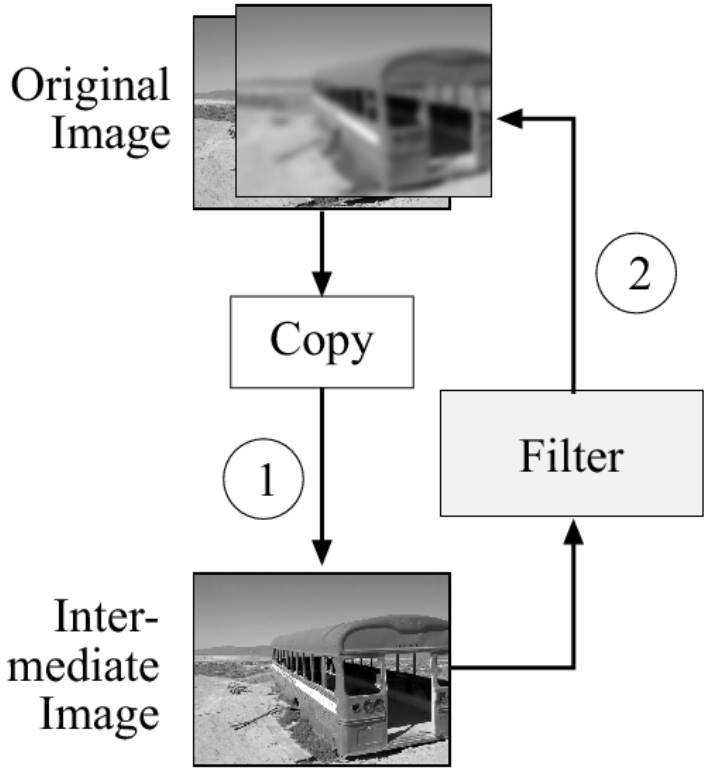


Praktische Implementierung von Filteroperationen

Im Gegensatz zu Punktoperationen ist bei Filtern keine ”in place”-Verarbeitung möglich, da die Quellpixel mehrere Male benötigt werden.



Variante A



Variante B

Beispiel: einfacher 3×3-Glättungsfilter

”Box”-Filter: 4 Schleifen

```
 8        int w = orig.getWidth();
 9        int h = orig.getHeight();
10        ImageProcessor copy = orig.duplicate();
11
12        for (int v=1; v<=h-2; v++) {
13            for (int u=1; u<=w-2; u++) {
14                //compute filter result for position (u,v)
15                int sum = 0;
16                for (int j=-1; j<=1; j++) {
17                    for (int i=-1; i<=1; i++) {
18                        int p = copy.getPixel(u+i,v+j);
19                        sum = sum + p;
20                    }
21                }
22                int q = (int) (sum / 9.0);
23                orig.putPixel(u,v,q);
24            }
25        }
```

Beispiel 2: 3×3-Glättungsfilter mit unterschiedlichen Koeffizienten

Glockenförmiger Glättungsfilter:

$$H(i,j) = \begin{bmatrix} 0.075 & 0.125 & 0.075 \\ 0.125 & \underline{0.200} & 0.125 \\ 0.075 & 0.125 & 0.075 \end{bmatrix}$$

```
2         int w = orig.getWidth();
3         int h = orig.getHeight();
4         // 3 × 3 filter matrix
5         double[][] filter = {
6             {0.075, 0.125, 0.075},
7             {0.125, 0.200, 0.125},
8             {0.075, 0.125, 0.075}
9         };
10        ImageProcessor copy = orig.duplicate();
11
12        for (int v=1; v<=h-2; v++) {
13            for (int u=1; u<=w-2; u++) {
14                // compute filter result for position (u, v)
15                double sum = 0;
16                for (int j=-1; j<=1; j++) {
17                    for (int i=-1; i<=1; i++) {
18                        int p = copy.getPixel(u+i,v+j);
19                        // get the corresponding filter coefficient:
20                        double c = filter[j+1][i+1];
21                        sum = sum + c * p;
22                    }
23                }
24                int q = (int) Math.round(sum);
25                orig.putPixel(u,v,q);
26            }
27        }
```