Bildverarbeitung: Filterung

Allgemeines

Klassische Anwendung: Entrauschung



(Fast) jeder Filter basiert auf einem Modell (Annahme): Signal + Rauschen

Pipeline: Modell → Aufgabe → Lösung → Algorithmus (Programm)

- Mittelwertfilter: Modell → ... → Lösung
- Medianfilter: Aufgabe → Lösung
- Paar andere Filter, Beispiele

- Lineare Filterung: Algorithmen

Mittelwertfilter

$D \subset \mathbb{Z}^2$ – Definitionsbereich (Gitter), r – Pixel $r = (i, j)$, $r \in D$

Bild ist eine Funktion $x : D \to C$ (Farbraum), x_r – Farbe des Pixels r

Sei y das „Ideale“ Signal (nicht verrauscht) und
x das verrauschte Signal (Beobachtung)

Aufgabe: Man sieht x, man berechne y.

Rauschmodell – Gaussche Wahrscheinlichkeitsverteilung für Abweichungen der Farben

$$p(x_r|y_r) = \mathcal{N}(x_r; y_r, \sigma) \sim \exp\left[-\frac{\|x_r - y_r\|^2}{2\sigma^2}\right]$$

Weitere Annahme – Pixel sind von einander unabhängig verrauscht:

$$p(x|y) \sim \prod_r \exp\left[-\frac{\|x_r - y_r\|^2}{2\sigma^2}\right]$$

Aufgabe nach dem Maximum Likelihood Prinzip: $p(y|x) = p(y)p(x|y)/p(x) \to \max_y$
Ohne zusätzlicher Annahmen über $p(y)$ ist die Lösung trivial: $y_r = x_r$ für alle r

Eine Annahme über das Signal y ist notwendig!!!

Annahme:
In einer kleinen Umgebung $W(r) \subset D$ eines Pixels r ist y (fast) konstant – y_r

$$\text{ML:} \qquad \prod_{r' \in W(r)} \exp\left[-\frac{\|x_{r'} - y_r\|^2}{2\sigma^2}\right] \to \max_{y_r}$$

$$\ln \ \ldots : \qquad F(y_r) = \sum_{r'} \|x_{r'} - y_r\|^2 \to \min_{y_r}$$

$$\text{Ableiten:} \qquad \frac{\partial F}{\partial y_r} = \sum_{r'} (x_{r'} - y_r) = \sum_{r'} x_{r'} - |W| \cdot y_r = 0$$

$$\Rightarrow \qquad y_r = \frac{1}{|W|} \sum_{r'} x_{r'} \quad \text{(Mittelwert)}$$

Median Filter

Andere Zielfunktion – entspricht einem anderen Rauschmodell (Einfachheit halber $C = \mathbb{R}$ – Grauwertbild):

$$F(y_r) = \sum_{r'} |x_{r'} - y_r| \to \min_{y_r}$$

Problem: nicht differenzierbar. Gute Neuigkeit – konvex.

(Ableitung an der Tafel).

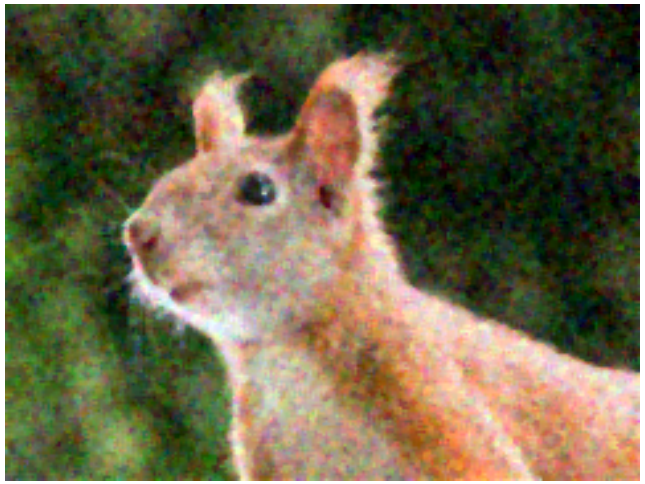
Lösung – Medianfilter.

Vergleich: Gaussches Rauschen

(a) Original

(b) Verrauscht

(c) Gaussche Glättung



(d) Medianfilter

Vergleich: „Salz und Pfeffer“ Rauschen

In manchen Pixeln ist der Farbwert zufällig gesetzt (gleichwahrscheinlich)

(e) Original

(f) Verrauscht

(g) Gaussche Glättung

(h) Medianfilter

Eine „andere“ Aufgabe

Glättung des Hintergrunds verbessert die räumliche Wahrnehmung der Szene

(Xue Bai, Guillermo Sapiro)

Umgekehrte Aufgabe: Rekonstruiere 3D aus Unschärfe.

Konsequenz: Die Filter liefern keine “Antworten“,
sie geben ein lokales Maß für die höheren Stufen der Verarbeitung.

Noch paar andere Aufgaben

Faltung:

$$y_r = \sum_{r'} x_{r'} \cdot g_{r'-r}$$

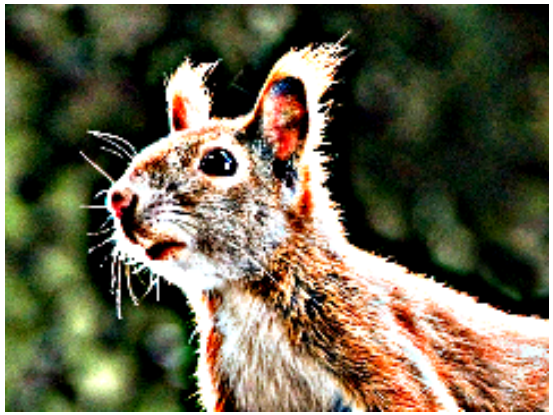
mit der Maske $g : \mathbb{Z}^2 \to \mathbb{R}$
Beispiel – Mittelwert:

$$g_r = \begin{cases} 1/|W| & \text{wenn} \quad r \in W(0) \\ 0 & \text{sonst.} \end{cases}$$

Welche weitere Masken sind denkbar (nützlich, wofür etc.)?

- Etwas „zartere“ Glättungen, z.B. mit dem Gausschen Kern: $g_r \sim \exp\left[-\|r\|^2/2\sigma^2\right]$
- Kontrasterhöhung: $g_r \sim \alpha \cdot \mathbb{1}(0) - \beta \cdot \exp\left[-\|r\|^2/2\sigma^2\right]$ (Unsharp Mask)

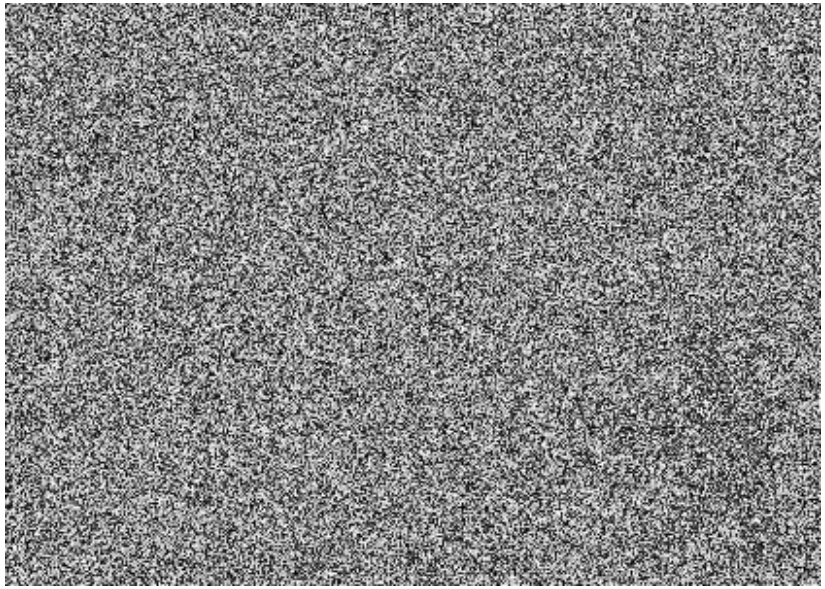




- Kantendetektoren, etc.

Eine ganz andere Aufgabe

„Salz und Pfeffer“ Rauschen mit 90%



Lokale Filterung ist offensichtlich kaum hilfreich

Explizite Modellierung des Signals ist notwendig (in diesem Fall – MRF)

Das Modell entspricht dem Bild „exakt“. Dies macht genaue Rekonstruktion möglich.

Fazit

Explizite Annahmen über das Signal (lokale Eigenschaften) – lokale Filterung

Signal wird durch seine Autokorrelationen beschrieben – Wiener Filter

Signal ist die Überlagerung der Signale bestimmter Frequenzen – Fourier Analyse

Signal wird durch eine Differentialgleichung beschrieben – Variationelle Ansätze

Explizite Modellierung statistischer Abhängigkeiten zwischen den Pixelwerten – MRF

u.s.w.

Die Eigenschaften des Signals
(das Modell, die Annahmen, die Art der zu lösenden Aufgabe etc.)
sind ausschlaggebend für die Wahl des Filters.

Lineare Filterung – Algorithmen

Eindimensional: $r \in \mathbb{N}$

Beispiel: Mittelwertfilter $y_r = \frac{1}{|W|} \sum_{r'=r-w}^{r+w} x_{r'} \qquad (|W| = 2w+1)$

Naiver Algorithmus (direkt nach der Formel):

for $r = 0$ `bis` n
    $sum = 0$
    for $r' = r - w$ `bis` $r + w$
        $sum = sum + x_{r'}$
    $y_r = sum/|W|$

Zeitkomplexität: $n \cdot |W|$

Idee: (Bildchen)

$$\sum_{r'=r-w}^{r+w} x_{r'} = \sum_{r'=0}^{r+w} x_{r'} - \sum_{r'=0}^{r-w-1} x_{r'} = \tilde{x}_{r+w} - \tilde{x}_{r-w-1}$$

Besserer Algorithmus:

Berechne $\tilde{x}_r$ **für alle** r**:**

for $r = 0$ **bis** n

$\tilde{x}_r = \tilde{x}_{r-1} + x_r$

Berechne y_r**:**

for $r = 0$ **bis** n

$y_r = (\tilde{x}_{r+w} - \tilde{x}_{r-w-1})/|W|$

Zeitkomplexität: n

Lineare Filterung – Algorithmen

Verallgemeinerung auf 2D:
Verwendung des „Integralbildes“ $\tilde{x}$

Berechne $\tilde{x}_r$**:**

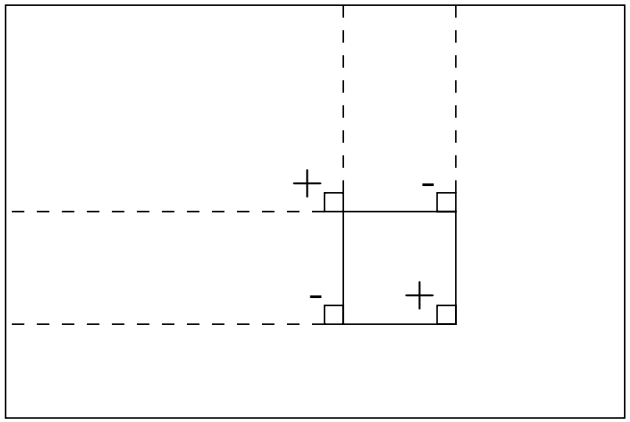
for $(i,j) = (0,0)$ **bis** (m,n)
(zeilenweise)

$$\tilde{x}_r = \tilde{x}_{i,j-1} + \tilde{x}_{i-1,j} - \tilde{x}_{i-1,j-1} + x_r$$

Berechne y_r**:**

for $(i,j) = (0,0)$ **bis** (m,n)

$$y_r = (\tilde{x}_{i+w,j+w} - \tilde{x}_{i-w,j+w} - \tilde{x}_{i+w,j-w} + \tilde{x}_{i-w,j-w})/|W|$$



Ein etwas komplizierteres Beispiel (wieder 1D, Bildchen):

$$y_r = \sum_{r'=r-w}^{r} (w - r + r') \cdot x_{r'}$$

Man überlege, ob sich y_{r+1} aus dem y_r effizient berechnen lässt:

$$\begin{aligned}
y_{r+1} &= \sum_{r'=r+1-w}^{r+1} (w - r - 1 + r') \cdot x_{r'} \\
&= \sum_{r'=r+1-w}^{r+1} (w - r + r') \cdot x_{r'} - \sum_{r'=r+1-w}^{r+1} x_{r'} \\
&= \sum_{r'=r-w}^{r} (w - r + r') \cdot x_{r'} + w \cdot x_{r+1} - \sum_{r'=r-w}^{r} x_{r'} \\
&= y_r + w \cdot x_{r+1} - \bar{x}_r
\end{aligned}$$

$\bar{x}_r$ kann mit linearer Zeitkomplexität berechnet werden (Mittelwertfilter)
$\Rightarrow$ die gesamte Zeitkomplexität ist linear.

Lineare Filterung – Algorithmen

Faltungen allgemein:

$$y = x * g$$

$$y_i = \sum_{j=-\infty}^{\infty} x_{i-j} \cdot g_j$$

Eigenschaften:

- sind kommutativ, d.h. $x * g = g * x$;
- sind assoziativ, d.h. $(x * g^1) * g^2 = x * (g^1 * g^2)$;
- sind distributiv mit „+“, d.h. $x * (g^1 + g^2) = x * g^1 + x * g^2$.

Identische Faltung (ändert das Signal nicht): $g^I_j = \mathbb{1}(j = 0)$

Inverse Faltungen: $g * g^{-1} = g^I$
Beispiel ($j = 0$ ist fett gekennzeichnet):

$$\begin{array}{rcll} g^{diff} & = & [\ldots, 0, 0, 0, \mathbf{1}, -1, 0, \ldots] & \text{Differential Operator} \\ g^{int} & = & [\ldots, 0, 0, 0, \mathbf{1}, \quad 1, 1, \ldots] & \text{Integral Operator} \end{array}$$

Lineare Filterung – Algorithmen

Der Trick zur effizienten Berechnung basiert auf der folgenden Umwandlung:

$$x * g = x * g^I * g = x * g^{int} * g^{diff} * g = (x * g^{int}) * (g * g^{diff})$$

oder sogar mehr

$$x * g = (x * g^{int} * \ldots * g^{int}) * (g * g^{diff} * \ldots * g^{diff})$$

Die Faltung $x * g^{int}$ braucht lineare Zeitkomplexität,
mit $\tilde{g} = g * g^{diff} * \ldots * g^{diff}$ (Vorberechnung) wird erreicht, dass $\tilde{g}$ schwach besetzt ist.