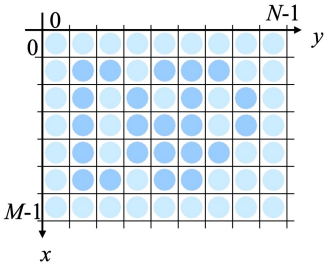
Bildeigenschaften

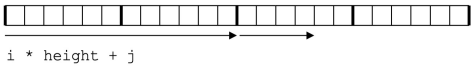
Repräsentation eines Bildes

$$f(x,y) = \begin{bmatrix} f(0,0) & \cdots & f(0,N-1) \\ \vdots & & \vdots \\ f(M-1,0) & \cdots & f(M-1,N-1) \end{bmatrix}$$



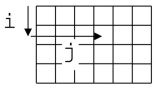
1D-Vektor (auch Unrolling / Reshaping)

- Deklaration: `int img[height * width]`
- Zugriff auf Element (i,j): `img[i*height + j]`



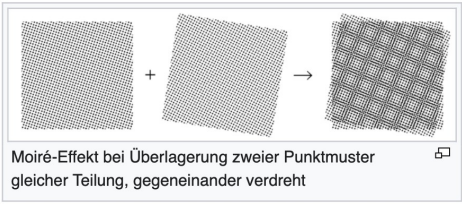
2D-Matrix

- Deklaration: `int img[height][width]`
- Zugriff auf Element (i,j): `img[i][j]`



Spatial Aliasing

irreparable Degradierung durch Artefakte wie zB Moiré-Effekt



Moiré-Effekt bei Überlagerung zweier Punktmuster gleicher Teilung, gegeneinander verdreht

Bits pro Pixel

Grayscale (Intensity Images):

Chan.	*Bits/Pix.*	*Range*	*Use*
1	1	0…1	Binary image: document, illustration, fax
1	8	0…255	Universal: photo, scan, print
1	12	0…4095	High quality: photo, scan, print
1	14	0…16383	Professional: photo, scan, print
1	16	0…65535	Highest quality: medicine, astronomy

Color Images:

Chan.	*Bits/Pix.*	*Range*	*Use*
3	24	$[0…255]^3$	RGB, universal: photo, scan, print
3	36	$[0…4095]^3$	RGB, high quality: photo, scan, print
3	42	$[0…16383]^3$	RGB, professional: photo, scan, print
4	32	$[0…255]^4$	CMYK, digital prepress

Special Images:

Chan.	*Bits/Pix.*	*Range*	*Use*
1	16	$-32768…32767$	Whole numbers pos./neg., increased range
1	32	$\pm 3.4 \cdot 10^{38}$	Floating point: medicine, astronomy
1	64	$\pm 1.8 \cdot 10^{308}$	Floating point: internal processing

Alpha-Channel bestimmt Transparenz eines Pixels

Example of an RGBA image composited over a checkerboard background. alpha is 0% at the top and 100% at the bottom.

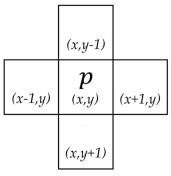
Datentypen

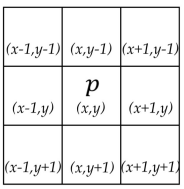
grundsätzlich uint, für Verarbeitung oft in float umgewandelt und am Ende wieder cast

Neighborhood

$N_4(p)$: Pixel und direkte Nachbarn (linke Abbildung)

$N_8(p)$: Pixel, direkte und diagonale Nachbarn (rechte Abbildung)





Adjacency

2 Pixel p und q sind adjacent falls $p \in N_4(q)$ oder $p \in N_8(q)$ UND ein Ähnlichkeits-Kriterium erfüllt ist

Pixeldistanz

zwei Pixel $p_1(x_1,y_1)$, $p_2(x_2,y_2)$ haben

- die euklidische Distanz $D_E = \sqrt{(x_1-x_2)^2 + (y_1-y_2)^2}$
- die Manhatten-Distanz $D_M = |x_1-x_2| + |y_1-y_2|$
- die Chessboard-Distanz $D_C = \max\{|x_1-x_2|, |y_1-y_2|\}$

D_E

$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$
$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$
2	1	**0**	1	2
$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$
$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$

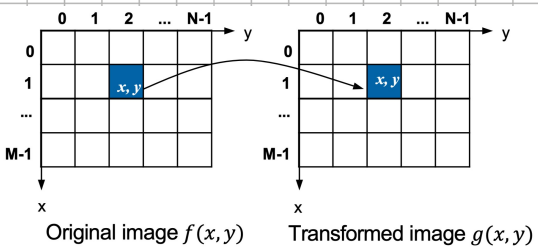
D_M

4	3	2	3	4
3	2	1	2	3
2	1	**0**	1	2
3	2	1	2	3
4	3	2	3	4

D_C

2	2	2	2	2
2	1	1	1	2
2	1	**0**	1	2
2	1	1	1	2
2	2	2	2	2

Intensitätsverteilungen und -transformationen



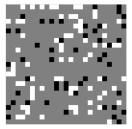
Histogramm

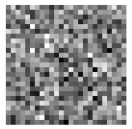
zeigt Anzahl Pixel für jeden Intensitätswert $k \in \{0, \dots, L-1\}$:

$$h(i_k) = n_k$$

kann (wenn normalisiert) als Wkeitsverteilung aufgefasst werden:

$$p(i_k) = \frac{n_k}{M \cdot N}$$

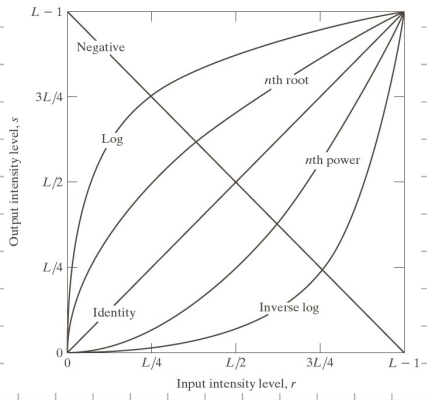




Rauschen

2 wichtige Modelle:
- Salz und Pfeffer Rauschen: kleine Teilmenge der Pixel entweder Schwarz oder Weiss
- Gausssches Rauschen: $g(x,y) = f(x,y) + \eta(x,y)$

Punkttransformationen



allgemein: $s = g(x,y) = T(f(x,y)) = T(r)$

Negativ: $s = L-1-r$ wobei $r \in \{0, \dots, L-1\}$

Log-T.: $s = c \log(1+r)$

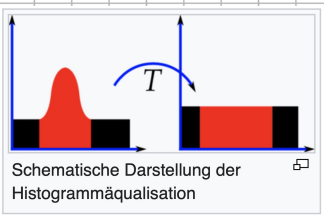
Gamma-T.: $s = c r^{\gamma}$

Lin. Kontr.-Streckung: $s = s_{min} + (r - r_{min}) \frac{s_{max} - s_{min}}{r_{max} - r_{min}}$

↳ um weniger sensibel von Ausreissern zu sein, kann man für r_{min} und r_{max} auch das 5te resp. 95te Percentil des Histogramms benutzen

Histogrammäqualisation

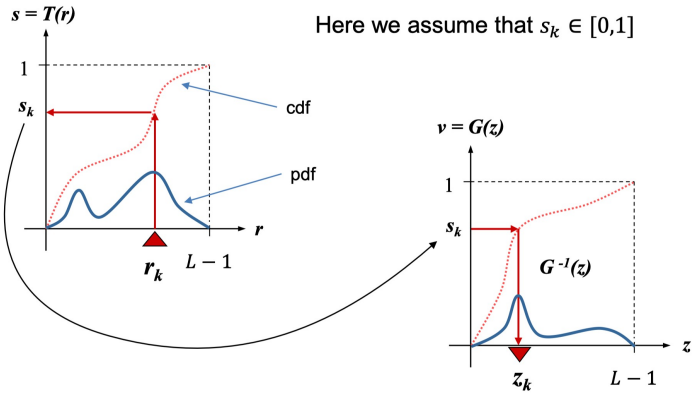
$$s_k = T(r_k) = (L-1)\,CDF(r_k) = (L-1)\sum_{j=0}^{k} \frac{n_i}{M \cdot N} = (L-1)\sum_{j=0}^{k} p_r(r_j)$$



Schematische Darstellung der Histogrammäqualisation

Histogrammanpassung

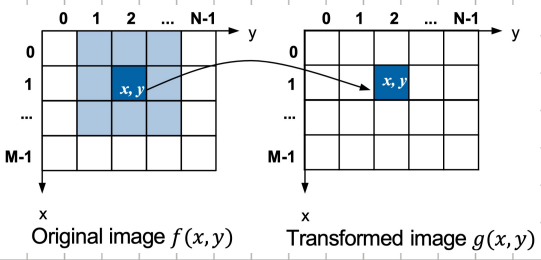
1. PDF (normalisiertes Histogramm) $p_r(r_k)$ des ursprünglichen Bildes berechnen
2. $s_k = T(r_k) = (L-1)\sum_{i=0}^{k} p_r(r_j)$ berechnen (Histogrammäq.)
3. aus Zielhistogramm resp. Ziel-PDF $p_z(z_k)$ eine CDF resp. Look-Up-Table $s_k = G(z_q) = (L-1)\sum_{i=0}^{q} p_z(z_i)$ berechnen
4. $z_q = G^{-1}(s_k) = G^{-1}(T(r_k))$ für alle s_k in Look-Up-Table nachschauen



Filterung im Ortsraum

Ausgang-Pixel ist Linearkombination von den Pixelwerten in der Nachbarschaft des Eingang-Pixels:

$$g(x,y) = \sum_{m=-a}^{a} \sum_{n=-b}^{b} h(m,n) \cdot f(x+m, y+n)$$ Korrelation



Tiefpass-Filterung (Glätten)

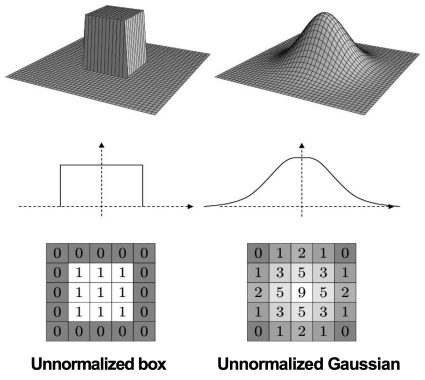
Rauschen wird weniger, Kanten werden aber verwaschen

Boxfilter / Mittelwertfilter: → separabel

$$\underline{H} = \frac{1}{m^2}\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & & \vdots \\ 1 & \cdots & 1 \end{bmatrix} \in \mathbb{R}^{m \times m}$$

Gaussfilter: → separabel

$$\underline{H} \text{ mit } h(x,y) = \frac{1}{2\pi\sigma} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) = K \exp\left(-\frac{r^2}{2\sigma^2}\right)$$



Faltung vs. Korrelation

Faltung: $(w * f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) \cdot f(x-s, y-t)$

Korrelation: $(w \star f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) \cdot f(x+s, y+t)$

Kann um 180° drehen

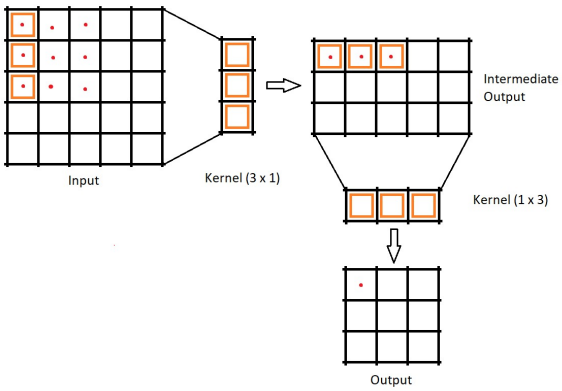
→ kommt bei symmetrischen Filterkernen nicht darauf an

separable Filter

separierbare Kerne $w = w_1 * w_2$ sind rechnerisch vorteilhaft:

$$\begin{aligned} w*f &= (w_1 * w_2) * f = w_1 * (w_2 * f) \\ &= (w_2 * w_1) * f = w_2 * (w_1 * f) \end{aligned}$$

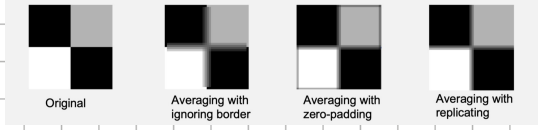
man kann zuerst mit mit dem einen und anschliessend mit dem anderen filtern, was den Aufwand von $M \cdot N \cdot m \cdot n$ auf $M \cdot N \cdot (m+n)$ reduziert



Umgang mit Bildrändern

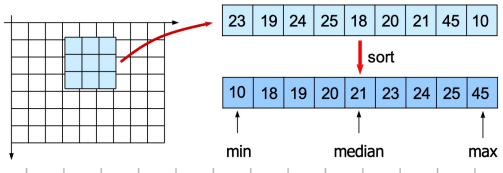
je nach Anwendung, 3 verbreitete Möglichkeiten:

- ignorieren: meist nicht empfohlen, da Ausgangsbild kleiner als Eingangsbild
- Zero-padding: einfach, kann aber zu Artifakten führen
- Rand-Pixel replizieren



Nichtlineare Filter

Ordnungsstatistikfilter:
Filter-Ausgang hängt von Resultat der Sortierung der Pixel ab
→ gut um Ausreisser zu entfernen (e.g. Salz/Pfeffer Rauschen)
→ bewahrt Kanten relativ gut
→ rechnerisch aufwändig (erfordert sort()-Operation)

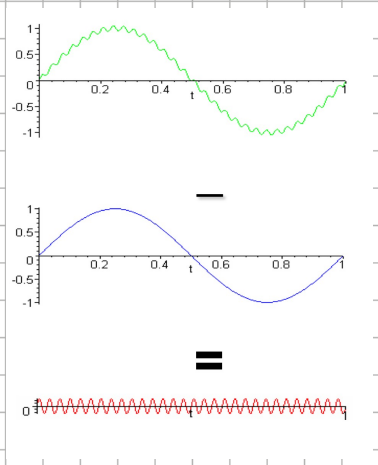


bilateraler Filter:
es werden nur Pixel verwendet, welche eine ähnliche Intensität wie das Pixel in der Mitte haben:

$$g[i,j] = \frac{1}{W_{sb}} \sum_m \sum_n f[m,n]\, n_{\sigma_s}[i-m, j-n]\, n_{\sigma_b}(f[m,n] - f[i,j])$$

mit $n_{\sigma_s}[m,n] = \frac{1}{2\pi\sigma_s^2} \exp\left(-\frac{1}{2}\frac{m^2+n^2}{\sigma_s^2}\right)$, $n_{\sigma_b}(k) = \frac{1}{\sqrt{2\pi}\sigma_b} \exp\left(-\frac{1}{2}\frac{k^2}{\sigma_b^2}\right)$ und $W_{sb} = \sum_m \sum_n n_{\sigma_s}[i-m, j-n]\, n_{\sigma_b}(f[m,n] - f[i,j])$

Hochpass-Filterung (Kantenanhebung)



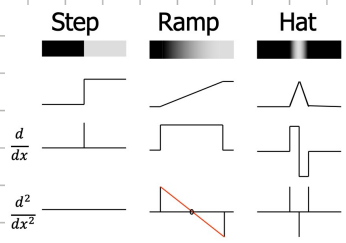
Kanten und kleine Objekte bleiben, während uniforme Strukturen "versteckt" werden

Unschärfemaskierung:

1. unscharfe Version des Bildes vom Original subtrahieren:
$$g_{mask}(x,y) = f(x,y) - \sum_{m=-a}^{a} \sum_{n=-b}^{b} h_{LP}(m,n) \cdot f(x+m, y+n)$$
2. resultierende Maske zum Original addieren:
$$g(x,y) = f(x,y) + k \cdot g_{mask}(x,y)$$

im Falle $k > 1$ spricht man auch von High-Boost-Filterung

Ableitungsbasiertes Schärfen:



- Prewitt-Operator: → separabel

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} * I = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} [1\ 0\ -1] * I, \quad G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * I = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} [1\ 1\ 1] * I$$

Gradientenbetrag: $G = \sqrt{G_x^2 + G_y^2}$

- Sobel-Operator: → separabel

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} [1\ 0\ -1] * I, \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} [1\ 2\ 1] * I$$

Gradientenbetrag: $G = \sqrt{G_x^2 + G_y^2}$

- Laplace-Filter-Kerne:

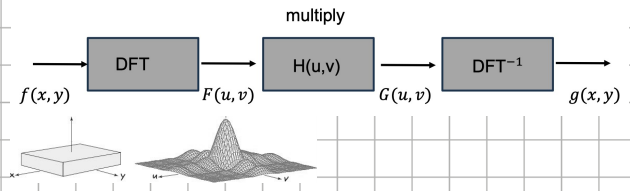
$$\nabla^2 f(x,y) = \frac{\partial^2}{\partial x^2} f(x,y) + \frac{\partial^2}{\partial y^2} f(x,y) \approx f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

$$\Rightarrow \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \text{ (isotrop für 90° Drehungen)}, \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{ (isotrop für 45° Drehungen)}$$

Schärfen: $g(x,y) + c \cdot \nabla^2 f(x,y)$

Filterung im Frequenzraum

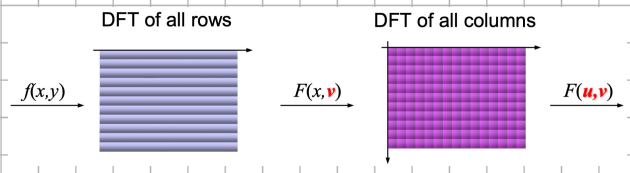
für grosse Filterkerne effizienter als im Ortsraum (dank FFT)



LSI-Systeme

der Ausgang eines linearen verschiebungsinvarianten (Linear Shift Invariant) Systems kann durch Faltung des Eingangsbildes mit dem Filterkern erhalten werden

2D Fourier Transformation



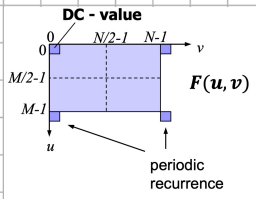
2D-FT: $F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\, e^{-j2\pi(ux+vy)}\, dx\, dy$

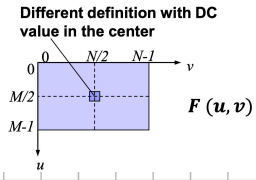
2D-DFT: $F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$

→ separabel: $\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} = \sum_{x=0}^{M-1}\left(\sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi\frac{vy}{N}}\right) e^{-j2\pi\frac{ux}{M}} = \sum_{x=0}^{M-1} F(x,v)\, e^{-j2\pi\frac{ux}{M}} = F(u,v)$

2D-IDFT: $f(x,y) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} F(u,v)\, e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$

Faltung: $(f * h)(x,y) \quad \circ\!\!-\!\!\bullet \quad F \cdot H(u,v)$



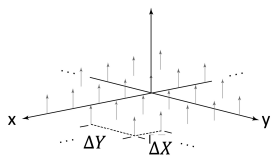


analog zu 1D treffen sich vier Viertelperioden im Punkt $(\frac{M}{2}, \frac{N}{2})$
→ zum Anzeigen und Filtern ist es praktischer den DC-Wert zu zentrieren

um einen höheren Dynamikbereich zu haben logarithmiert man das Betragsspektrum zuvor

Abtasttheorem in 2D

abgetastetes Bild: $S_{\Delta X \Delta Y}(x,y) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} \delta(x - m\Delta X, y - n\Delta Y)$

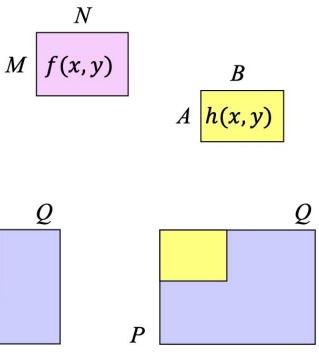


eine stetige, bandbegrenzte Funktion kann fehlerfrei aus ihren Samples rekonstruiert werden, wenn die Abtastintervalle $\Delta X < 1/2u_{max}$ und $\Delta Y < 1/2v_{max}$ betragen

Filterung

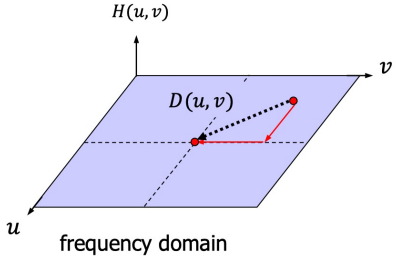
wie in 1D muss man auch in 2D angemessen mit Nullen auffüllen damit die Multiplikation im Frequenzraum der linearen und nicht der zirkularen Faltung im Ortsraum entspricht:

$$P \geq M + A - 1 \quad \text{und} \quad Q \geq N + B - 1$$



wobei: f(x,y) Grösse $M \times N$ und h(x,y) Grösse $A \times B$ hat

Vorgehen:
1. Padding-Parameter P und Q bestimmen
2. DFT: $F_{pad}(u,v) = FFT(f_{pad}(x,y))$, $H_{pad}(u,v) = FFT(h_{pad}(x,y))$
3. Multiplikation: $G_{pad}(u,v) = F_{pad}(u,v) \cdot H_{pad}(u,v)$
4. IDFT: $g_{pad}(x,y) = Re(IFFT(G_{pad}(u,v))$
5. Bereich wählen: $g(x,y) = g_{pad}(?,?)$



Distanzmessung im Frequenzraum:

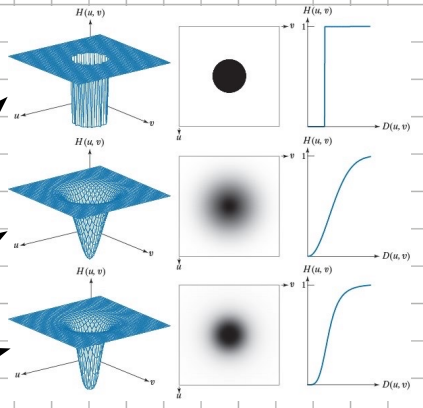
D(u,v) = euklidische Distanz von (u,v) vom Ursprung

Tiefpassfilter

ideales TP: $H(u,v) = \begin{cases} 1 & D(u,v) \leq D_0 \\ 0 & D(u,v) > D_0 \end{cases}$ → tendiert zu Artefakten

Butterworth TP n^{ter} Ordnung: $H(u,v) = \frac{1}{1+(D(u,v)/D_0)^{2n}}$

Gaussscher TP: $H(u,v) = \exp\left(\frac{-D^2(u,v)}{2\sigma^2}\right)$



Hochpassfilter

TP-HP-Transformation: $H_{HP}(u,v) = 1 - H_{TP}(u,v)$

idealer HP: $H(u,v) = \begin{cases} 0 & D(u,v) \leq D_0 \\ 1 & D(u,v) > D_0 \end{cases}$

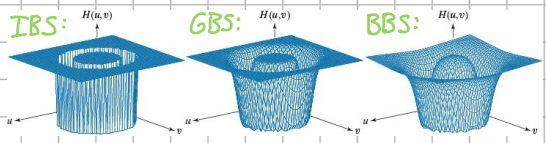
Butterworth HP n^{ter} Ordnung: $H(u,v) \overset{+}{=} \frac{1}{1+(D_0/D(u,v))^{2n}}$

Gaussscher HP: $H(u,v) = 1 - \exp\left(\frac{-D^2(u,v)}{2\sigma^2}\right)$

+: $1 - \frac{1}{1+(D(u,v)/D_0)^{2n}} = \frac{1+(D(u,v)/D_0)^{2n}}{1+(D(u,v)/D_0)^{2n}} - \frac{1}{1+(D(u,v)/D_0)^{2n}} = \frac{(D(u,v)/D_0)^{2n}}{1+(D(u,v)/D_0)^{2n}} = \frac{1}{(D(u,v)/D_0)^{-2n}+1} = \frac{1}{1+(D(u,v)/D_0)^{2n}}$

Bandpass/-sperre

BP-BS-Transformation: $H_{BS}(u,v) = 1 - H_{BP}(u,v)$



Notch-Filter

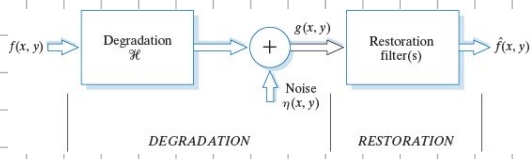
periodischen Rauschen entfernen → Notches am Ort der Impulse im Frequenzraum platzieren

Inverse Filterung

degradiertes Bild: $g(x,y) = f(x,y) * h(x,y)$ Verschlechterungsprozess durch Filter $H(u,v)$ beschreiben

⇒ inverse Filterung: $\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)} \bullet\!\!-\!\!\circ \hat{f}(x,y)$ Annäherung an das Originalbild

Wiener-Filter



Problem: Rauschen

inverse Filterung: $\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)} = \frac{F(u,v)\cdot H(u,v) + \sigma^2}{H(u,v)}$

Annahme: additives Rauschen $N(u,v) = \sigma^2$ habe konstantes Leistungsspektrum

Wo $H(u,v)$ nahe 0 ist, nimmt $\hat{F}(u,v)$ enorme Werte an ⇒ Rauschen wird verstärkt

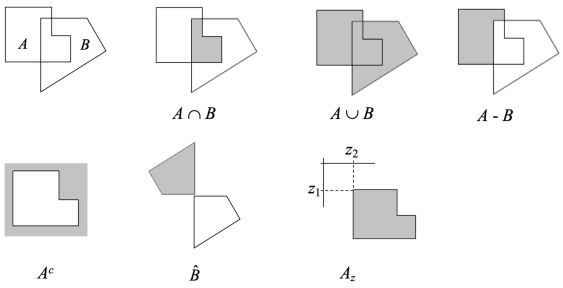
Lösung (Wiener-Filter):

$$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)} \cdot \left(\frac{1}{1+\frac{NSR(u,v)}{|H(u,v)|^2}}\right) = \frac{H^*(u,v)G(u,v)}{|H(u,v)|^2 + NSR(u,v)}$$

mit Noise-to-Signal Ratio (NSR) wobei: $NSR(u,v) = \frac{|N(u,v)|^2}{|H(u,v)|^2} = \frac{\text{Leistung des Rauschens bei } (u,v)}{\text{Leistung des ursprünglichen Bildes bei } (u,v)}$

typischerweise weder $N(u,v)$ noch $F(u,v)$ genau bekannt
⇒ oft wird NSR als konstant angenommen resp $NSR(u,v) = k$ (k als Tuning-Parameter verwenden)

Morphologische Bildverarbeitung



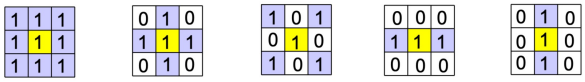
Mengenoperationen

Differenz: $A-B = A \setminus B = A \cap B^c = \{z \mid (z \in A) \wedge (z \notin B)\}$

Reflexion / Punktspiegelung: $\hat{B} = \{w \mid w = -b, \text{ für } b \in B\}$

Translation: $(B)_z = \{c \mid c = b + z, \text{ für } b \in B\}$

Strukturelemente (SE)



definieren Nachbarschaft und identifizieren welche Pixel verwendet werden sollen (können irgendetwas sein)
als Matrix mit definiertem Ursprung dargestellt

Erosion und Dilation

Dilation: $A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\} = \{z \mid ((\hat{B}_z) \cap A) \subseteq A\}$

Erosion: $A \ominus B = \{z \mid (B)_z \subseteq A\} = \{z \mid (B)_z \cap A^c = \emptyset\}$

verhalten sich bezüglich Komplementierung und Reflexion dual zueinander: $(A \ominus B)^c = A^c \oplus \hat{B}$, $(A \oplus B)^c = A^c \ominus \hat{B}$

Erosion und Dilation kombinieren: Öffnen und Schliessen

Öffnen: $A \circ B = (A \ominus B) \oplus B = \bigcup \{(B)_z \mid (B)_z \subseteq A\}$

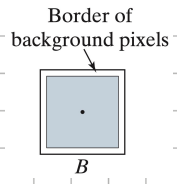
→ geometrisch: alle Punkte p in A, sodass das Strukturelement B, um p platziert, in A enthalten ist
→ entfernt kleine Strukturen (kleiner als Strukturelement), glättet Konturen, öffnet enge Verbindungen, entfernt hervorstehende Teile

Schliessen: $A \bullet B = (A \oplus B) \ominus B = \left(\bigcup \{(B)_z \mid (B)_z \cap A = \emptyset\}\right)^c$

→ geometrisch: Strukturelement rollt auf Grenzen von A
→ glättet Konturen, schliesst schmale Lücken und lange dünne Buchten, beseitigt kleine Löcher, füllt Löcher in der Kontur

verhalten sich bezüglich Komplementierung und Reflexion dual zueinander: $(A \circ B)^c = A^c \bullet \hat{B}$, $(A \bullet B)^c = A^c \circ \hat{B}$

Hit-Or-Miss-Transformation (HMT)



Definition: $A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2) = \{z \mid (B_1)_z \subseteq A \text{ und } (B_2)_z \subseteq A^c\}$

→ B_1 muss im Vordergrund enthalten sein und gleichzeitig B_2 im Hintergrund
→ Formen identifizieren: B_1 gemäss der gesuchten Form wählen und B_2 als einen Rahmen um B_1 wählen

weitere Transformationen

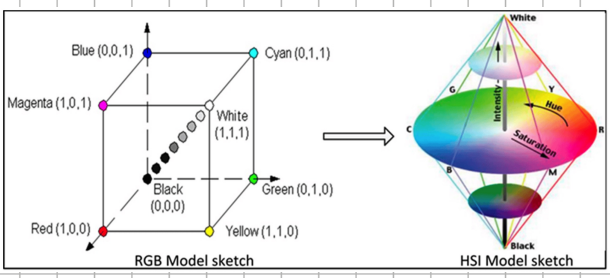
Randextraktion: $\beta(A) = A - (A \ominus B)$

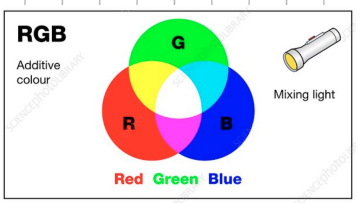
Lochfüllung:
1. X_0 mit Nullen und in jedes Loch eine 1
2. Iteriere $X_k = (X_{k-1} \oplus B) \cap I^c$ bis $X_k = X_{k-1}$

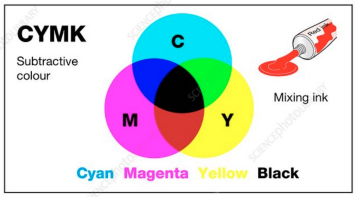
Extraktion verbundener Komponenten:
1. X_0 mit Nullen und in jede zusammenhängende Struktur eine 1
2. Iteriere $X_k = (X_{k-1} \oplus B) \cap I$ bis $X_k = X_{k-1}$

Farben

Farbmodelle







RGB- und CMY(k)-Modell:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

HSI-Farbraum:

$$H = \begin{cases} \theta & \text{falls } B \leq G \\ 360° - \theta & \text{falls } B > G \end{cases} \quad \rightarrow \text{Hue (bestimmt dominante Farbe)}$$

$$\theta = \arccos\left(\frac{\frac{1}{2}((R-G)+(R-B))}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right)$$

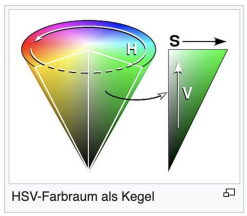
$$S = 1 - \frac{3}{R+G+B}\min(R,G,B) \quad \rightarrow \text{Sättigung (misst wie stark reine Farbe durch Weiss verdünnt wird)}$$

$$I = \frac{1}{3}(R+G+B) \quad \rightarrow \text{Lichtintensität}$$

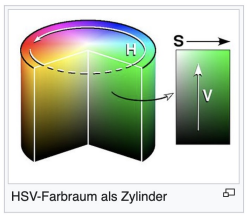
HSV:

$$V = \max(R,G,B)$$

und auch leicht andere Formeln für H und S...



HSV-Farbraum als Kegel



HSV-Farbraum als Zylinder

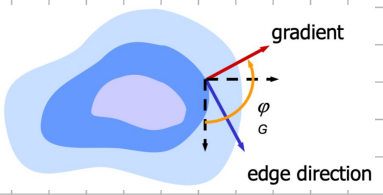
Farbbild in Graubild umwandeln

$$Y = w_R R + w_G G + w_B B \quad (w_R \approx 0.2,\ w_G \approx 0.7,\ w_B \approx 0.1)$$

Bildverarbeitung

2 grundsätzliche Herangehensweisen:
- Bild schichtweise behandeln und Kombination der einzelnen Ergebnisse
- gleichzeitige Behandlung aller Schichten

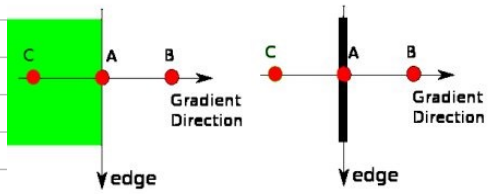
Linien und Kanten



Gradient

$$\nabla f(x,y) = \begin{bmatrix} G_x \\ G_y \end{bmatrix} \text{ resp. } |\nabla f(x,y)| = \sqrt{G_x^2 + G_y^2} \approx |G_x| + |G_y|, \quad \varphi_G = \arctan\left(\frac{G_y}{G_x}\right)$$

Canny Kanten Detektor



1. Bild mit Gaussfilter glätten um Rauschen zu unterdrücken
2. Gradient berechnen (Sobel, Prewitt, etc...)
   → Betrag und Winkel
3. Nicht-Maxima unterdrücken → $r(x,y)$
   - in Gradientenrichtung nach Pixel mit höchstem Gradientenbetrag suchen → Kamm gefunden
   - Kamm folgen (Richtung ist orthogonal zu Gradient)
4. Hysterese - Schwellenwertvorarbeitung

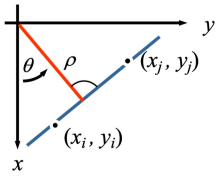
a) Pixel mit $T_1 \leq r(x,y)$ welche grossen Schwellenwert T_1 überschreiten werden definitiv zum Kamm gezählt

b) Zerklüftete Kämme mit tiefem Schwellenwert T_2 verbinden
→ Pixel mit $T_2 \leq r(x,y) < T_1$ werden zu Kamm gezählt, wenn ein Kammpixel in der 8-Nachbarschaft verbunden ist

Hough Transformation für Linien

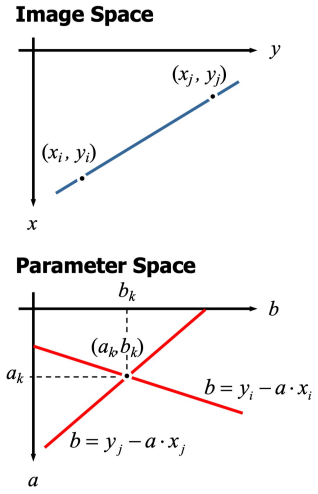
Problem mit Parametrisierung $b = y_k - a x_k$ ist dass für vertikale Linien $a \to \infty$ geht

→ besser mit Hesse-Normal-Form parametrisieren: $x\cos(\theta) + y\sin(\theta) = \rho$



jeder Punkt in $f(x,y)$ wird in eine sinusoidale Kurve im Parameterraum (θ, ρ) transformiert

falls Kantengradienten verlässlich sind, kann man θ effizienter mit $\theta = \theta_G = \text{atan2}(G_y(x,y), G_x(x,y))$ berechnen



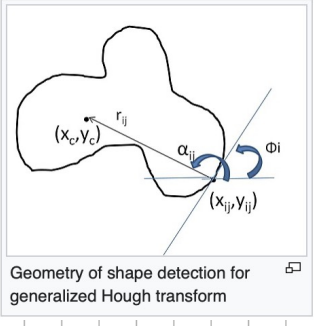
optional kann man die Stimme zusätzlich mit dem Betrag des Gradienten gewichten

analog kann man bei Kreisen und anderen analytisch parametrisierbaren Formen vorgehen

Generalisierte Hough Transformation

Modell für Form: R-Tabelle
$\Phi_k = \frac{2\pi \cdot k}{K}$, $[[r_{k,0}, \alpha_{k,0}], \ldots, [r_{k,l_k}, \alpha_{k,l_k}]]$ wobei $k \in \{0, \ldots, K-1\}$
→ enthält für jeden Winkel Φ_k eine Liste aller Vektoren welche zum Anker (x_c, y_c) zeigen

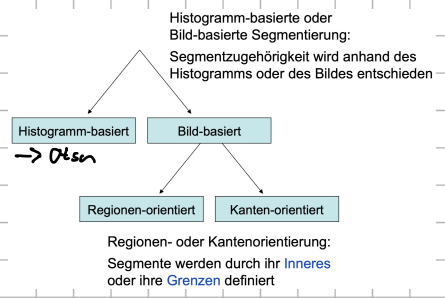
Form im Bild suchen: Akkumulator-Matrix
für jeden Punkt (x,y) im Bild den Winkel des Gradienten berechnen
in R-Tabelle entsprechende Liste suchen
für jeden Vektor (r, α) die Addition mit (x,y) berechnen für das Resultat (x_c, y_c) in der Akkumulator-Matrix stimmen



Geometry of shape detection for generalized Hough transform

um verschiedene Orientierungen und Skalierungen zu berücksichtigen, muss man die Akkumulator-Matrix auf ein 4-dimensionales Array erweitern, da das Resultat für jede Skalierung S und Orientierung berechnet werden muss
→ $(x_c, y_c) = (x + S \cdot r \cdot \cos(\alpha + \theta),\ y + S \cdot r \cdot \sin(\alpha + \theta))$

Segmentation



einfache Methode

1. T schätzen, e.g. Mittelpunkt zwischen Maximum oder durchschnittlicher Grauwert
2. Bild mit T_n in G_1 ($\leq T_n$) und G_2 ($> T_n$) segmentieren
3. durchschnittlichen Wert m_1 und m_2 in G_1 resp G_2 bestimmen
4. $T_{n+1} = \frac{1}{2}(m_1 + m_2)$
5. falls $|T_{n+1} - T_n| < \varepsilon$ Abbruch sonst wiederholen

Otsu Methode

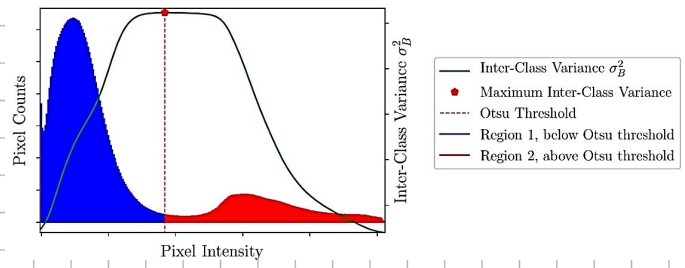
Klassenvarianzen:

$$\sigma_1^2(k) = \frac{1}{P_1(k)} \sum_{i=0}^{k-1} p(i)(i - m_1(k))^2$$
$$\sigma_2^2(k) = \frac{1}{P_2(k)} \sum_{i=k}^{L-1} p(i)(i - m_2(k))^2$$

mit $m_1(k) = \frac{1}{P_1(k)} \sum_{i=0}^{k-1} p(i)\, i$, $m_2(k) = \frac{1}{P_2(k)} \sum_{i=k}^{L-1} p(i)\, i$, $P_1(k) = \sum_{i=0}^{k-1} p(i)$, $P_2(k) = \sum_{i=k}^{L-1} p(i)$

Intra-Klassen-Varianz:

$$\sigma_w^2 = P_1 \sigma_1^2 + P_2 \sigma_2^2$$



Inter-Klassen-Varianz:

$$\sigma_B^2 = \sigma_G^2 - \sigma_w^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2$$

wobei $\sigma_G^2 = \sum_{i=0}^{L-1} p(i)(i - m_G)^2$ wobei $m_G = \sum_{i=0}^{L-1} i \cdot p(i)$

σ_B^2 kann man umschreiben: → effizienter, da $m(k)$ und $P_1(k)$ so iterativ berechnet werden können

$$\sigma_B^2(k) = P_1(k) P_2(k) (m_2(k) - m_1(k))^2 = \frac{(m_G P_1(k) - m(k))^2}{P_1(k)(1 - P_1(k))}$$

→ maximieren

mit $m(k) = \sum_{i=0}^{k-1} i \cdot p(i) = P_1(k) \cdot m_1(k)$

Separabilitätsmass: $\eta = \frac{\sigma_B^2}{\sigma_G^2}$ wobei nicht separierbar $= 0 \leq \eta(k) \leq 1 =$ einfach separierbar

Otsu's Methode kann auch auf mehrere Schwellen erweitert werden

Globale Schwelle und Kanten

Rauschen tendiert dazu die Histogrammspitzen für kleine Objekte zu verschmieren
→ nur Pixel auf oder nahe der Kante berücksichtigen

Algorithmus:
1. Kanten finden
2. Kantenbild mit Schwellenverfahren binarisieren
3. originales Bild mit Binärbild aus Schritt 2 maskieren
4. Histogramm des maskierten Bildes aus Schritt 3 verwenden um Schwellenwert zu finden (e.g. Otsu, etc.)
5. originales Bild mit Schwellenwert aus Schritt 4 segmentieren

lokale Schwellen

→ z.B. basierend auf Mittelwert und Standardabweichung der Nachbarschaft:

$$T_{xy} = a\sigma_{xy} + b m_{xy} \quad \text{oder} \quad T_{xy} = a\sigma_{xy} + b m_G$$

mit $a, b > 0$
→ gibt viele weitere Möglichkeiten für lokale Schwellen...

Farben-Clustering

Fehlermass:

$$J = \sum_{i=1}^{k} \sum_{\vec{x}_j \in S_i} \|\vec{x}_j - \vec{\mu}_i\|^2$$

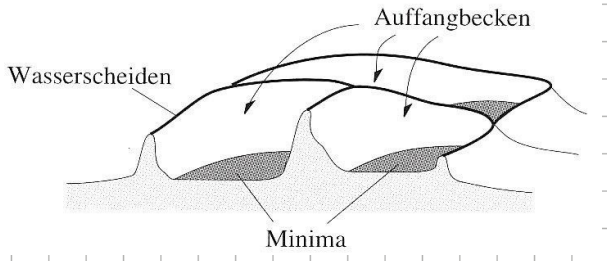
proportional zur durchschnittlichen in-Klass-Varianz (wie bei Otsu)

mit der Menge S_i der Bildpunkte in Klasse i und dem Farbzentrum $\vec{\mu}_i$ der Klasse i

K-Means - Algorithmus:
1. Startfarbe für alle K Klassen
2. Iterative Klassenerfügung von Pixeln:
a) Pixel $f(x,y)$ der Klasse hinzufügen, dessen Klassen-Varianz am wenigsten zunimmt
b) Klassenmittelpunkt neu berechnen
c) a) und b) wiederholen bis sich Klassenzugehörigkeit nicht mehr ändert

Morphologische Wasserscheiden

$g(x,y)$	Bild
M_i	Minimum des Bildes
$C(M_i)$	Einzugsgebiet von Minimum M_i
$T[n]$	$\{(s,t) \mid g(s,t) < n\}$
$C_n(M_i)$	$C(M_i) \cap T[n]$
$C[n]$	$\bigcup_{i=1}^{R} C_n(M_i)$

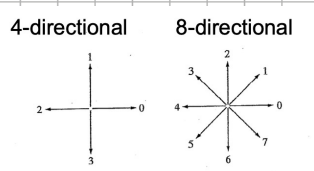


Algorithmus startet mit $C[\min+1] = T[\min+1]$
berechnet rekursiv $C[n]$ aus $C[n-1]$, basierend auf der Beziehung zwischen q (eine verbundene Komponente in $T[n]$) und $C[n-1]$, wobei 3 Fälle auftreten können:
1. $q \cap C[n-1] = \emptyset$, was ein neues Minimum anzeigt und q wird zu $C[n-1]$ hinzugefügt um $C[n]$ zu bilden
2. $q \cap C[n-1]$ enthält eine Komponente von $C[n-1]$, und q wird in $C[n-1]$ integriert um $C[n]$ zu bilden
3. $q \cap C[n-1]$ enthält mehrere Komponenten, was einen Grat darstellt
   => mittels Dilatation von $q \cap C[n-1]$ wird ein Damm errichtet, um Überflutung zu verhindern

Feature - Extraktion

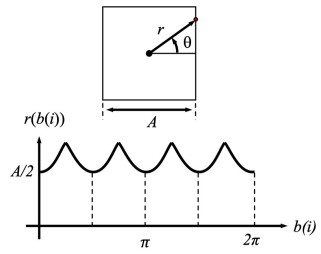
Randbasierte Deskriptoren

Ketten-Code



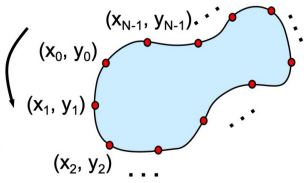
Encoder bewegt sich entlang der Grenze eines Objekts und überträgt bei jedem Schritt ein Symbol, das die Richtung dieser Bewegung darstellt
→ verlustfreie Komprimierung um ein Objekt in einem Binärbild zu repräsentieren

Signaturen



eindimensionale Darstellung der Grenze eines Objekts, meist basierend auf Binärbildern
→ messen bspw. Abstand von einem Referenzpunkt wie dem Schwerpunkt zur Grenze in Abhängigkeit von einem Winkel
→ translationsinvariant, aber grundsätzlich von Skalierung und Rotation abhängig
→ Modifizierung: auf maximal 1 skalieren und beim Maximum Winkel $\doteq 0$ setzen

Fourier-Deskriptoren



$$(x_i, y_i) \rightarrow x_i + j \cdot y_i \Rightarrow \vec{s}(k) = (x_0 + jy_0, \dots, x_{N-1} + jy_{N-1})^T$$

$$\vec{F}(u) = FFT(\vec{s}) = \sum_{k=0}^{N-1} \vec{s}(k) \cdot \exp\left(-2\pi j \frac{uk}{N}\right)$$

Eigenschaften:

- Translation um t: $\vec{s}(k) \rightarrow \vec{s}(k) + t = \vec{s}(k) + t_x + j \cdot t_y$

$$\vec{F}_t(u) = \sum_{k=0}^{N-1} (\vec{s}(k) + t) \cdot \exp\left(-2\pi j \frac{uk}{N}\right)$$
$$= \underbrace{\sum_{k=0}^{N-1} \vec{s}(k) \cdot \exp\left(-2\pi j \frac{uk}{N}\right)}_{\vec{F}(u)} + \underbrace{\sum_{k=0}^{N-1} t \cdot \exp\left(-2\pi j \frac{uk}{N}\right)}_{N \cdot t \text{ für } u=0,\ 0 \text{ sonst}}$$

- Skalierung mit v: $\vec{s}(k) \rightarrow v \cdot \vec{s}(k)$

$$\vec{F}_s(u) = \sum_{k=0}^{N-1} v \cdot \vec{s}(k) \cdot \exp\left(-2\pi j \frac{uk}{N}\right) = v \cdot \sum_{k=0}^{N-1} \vec{s}(k) \cdot \exp\left(-2\pi j \frac{uk}{N}\right) = v \cdot \vec{F}(u)$$

- Rotation um φ: $\vec{s}(k) \rightarrow e^{j\varphi} \cdot \vec{s}(k)$

$$\vec{F}_r(u) = \sum_{k=0}^{N-1} e^{j\varphi} \cdot \vec{s}(k) \cdot \exp\left(-2\pi j \frac{uk}{N}\right) = e^{j\varphi} \cdot \sum_{k=0}^{N-1} \vec{s}(k) \cdot \exp\left(-2\pi j \frac{uk}{N}\right) = e^{j\varphi} \cdot \vec{F}(u)$$

- Reihenfolge schieben: $\vec{s}(k) \rightarrow \vec{s}(k - k_0)$

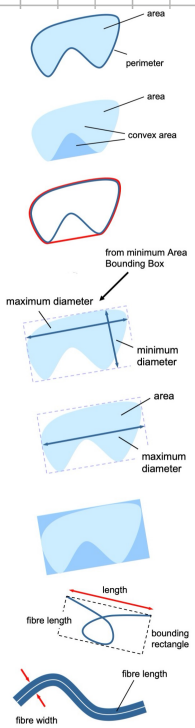
$$\vec{F}_{sp}(u) = \sum_{k=0}^{N-1} \vec{s}(k - k_0) \cdot \exp\left(-2\pi j \frac{uk}{N}\right) = \sum_{k=0}^{N-1} \vec{s}(k) \cdot \exp\left(-2\pi j \frac{u(k+k_0)}{N}\right)$$
$$= \exp\left(-2\pi j \frac{uk_0}{N}\right) \cdot \sum_{k=0}^{N-1} \vec{s}(k) \cdot \exp\left(-2\pi j \frac{uk}{N}\right) = e^{-2\pi j \frac{uk_0}{N}} \vec{F}(u)$$

- Reihenfolge umdrehen: $\vec{s}(k) \rightarrow \vec{s}(N - k)$

$$\vec{F}_{dir}(u) = \sum_{k=0}^{N-1} \vec{s}(N-k) \cdot \exp\left(-2\pi j \frac{uk}{N}\right) = \sum_{k=0}^{N-1} \vec{s}(k) \cdot \exp\left(-2\pi j \frac{u(N-k)}{N}\right)$$
$$= e^{-2\pi j \frac{uN}{N}} \sum_{k=0}^{N-1} \vec{s}(k) \cdot \exp\left(-2\pi j \frac{(-u)k}{N}\right) = 1 \cdot \sum_{k=0}^{N-1} \vec{s}(k) \cdot \exp\left(-2\pi j \frac{(-u)k}{N}\right)$$
$$= \sum_{k=0}^{N-1} \vec{s}(k) \cdot 1 \cdot \exp\left(-2\pi j \frac{(-u)k}{N}\right) = \sum_{k=0}^{N-1} \vec{s}(k) \cdot e^{-2\pi j \frac{uN}{N}} \cdot \exp\left(-2\pi j \frac{(-u)k}{N}\right)$$
$$= \sum_{k=0}^{N-1} \vec{s}(k) \cdot \exp\left(-2\pi j \frac{(N-u)k}{N}\right) = \vec{F}(N-u)$$

Polygonzug → Liste der Eckpunkte in Uhrzeigerrichtung: $[\vec{p}_0, \vec{p}_1, \vec{p}_2, \dots, \vec{p}_{N-1}]$, $\vec{p}_i \in \mathbb{R}^2$

einfache Deskriptoren



basierend auf Fläche und Umfang

- Formfaktor: $F = 4\pi \frac{\text{Fläche}}{\text{Umfang}^2}$
- Solidität: $S = \frac{\text{Fläche}}{\text{konvexe Fläche}}$
- Konvexität: $C = \frac{\text{konvexer Umfang}}{\text{Umfang}}$

basierend auf Fläche und Durchmesser

- Seitenverhältnis: $A = \frac{\text{maximaler Durchmesser}}{\text{minimaler Durchmesser}}$
- Rundheit: $R = \frac{4}{\pi} \frac{\text{Fläche}}{(\text{maximaler Durchmesser})^2}$
- Kompaktheit: $P = \sqrt{R}$
- Ausdehnung: $E = \frac{\text{Fläche}}{\text{Fläche umgebendes Rechteck}}$

Beschreiber für Fasern

- Krümmung: $C_r = \frac{\text{Länge}}{\text{Faserlänge}}$
- Dehnung: $E_L = \frac{\text{Faserlänge}}{\text{Faserbreite}}$

flächenbasierte Deskriptoren

geht auch für Graustufenbild

sei $f(x,y)$ Binärbild der Größe $M \times N$, i.e. $f(x,y) \in \{0,1\}^{M \times N}$

Momente der Ordnung $(p+q)$: $m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p \cdot y^q \cdot f(x,y)$

zentrale Momente: $\mu_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x-\bar{x})^p \cdot (y-\bar{y})^q \cdot f(x,y) \quad \bar{x} = \frac{m_{10}}{m_{00}}, \; \bar{y} = \frac{m_{01}}{m_{00}}$

Fläche: $m_{00} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) = \sum_{\text{objekt}} 1$ → Anzahl Pixel in Objekt

Momente 1ter Ordnung:

$$m_{10} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x \cdot f(x,y) = \sum_{\text{objekt}} x$$
$$m_{01} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} y \cdot f(x,y) = \sum_{\text{objekt}} y$$

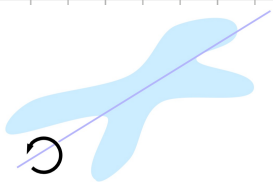
Koordinaten des Schwerpunkts:

$$M_x = \bar{x} = m_{10} / m_{00}$$
$$M_y = \bar{y} = m_{01} / m_{00}$$

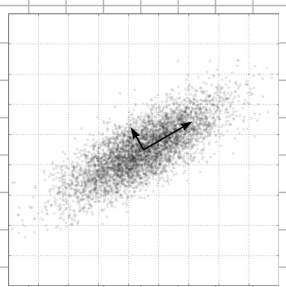
Hauptachsen der Trägheit:

$$\theta = \frac{1}{2} \arctan 2(2 \cdot M_{xy}, M_{xx} - M_{yy})$$

wobei $M_{xy} = m_{11} - \frac{m_{10} m_{01}}{m_{00}}$, $M_{xx} = m_{20} - \frac{m_{10}^2}{m_{00}}$, $M_{yy} = m_{02} - \frac{m_{01}^2}{m_{00}}$



alternativ: PCA



Pixel als Punktwolke: (x_i', y_i')
→ shiften, sodass Schwerpunkt bei $(x,y) = (0,0)$

Matrix mit allen Pixel: $\underline{A} = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix} = \begin{bmatrix} \vec{x}_1^T \\ \vdots \\ \vec{x}_n^T \end{bmatrix}$

Hauptachse $\hat{\vec{x}}$ maximiert $\|\underline{A}\vec{x}\|$, i.e. $\hat{\vec{x}} = \underset{\vec{x} \in \mathbb{R}^2, \|\vec{x}\|=1}{\arg\max} \{\vec{x}^T \underline{A}^T \underline{A} \vec{x}\}$

Lösung ist Eigenvektor der Kovarianzmatrix $\underline{C} := \underline{A}^T \underline{A}$ der mit dem größten Eigenwert assoziiert ist

zentralen Momente sind unabhängig von Translation: $\mu_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x-\bar{x})^p \cdot (y-\bar{y})^q \cdot f(x,y)$

normalisierte zentrale Momente:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \quad \text{wobei} \quad \gamma = \frac{p+q}{2} + 1$$

Hu-Momente:
aus normalisierten zentralen Momenten kann man 7 invariante Momente ableiten →

$$\Phi_1 = \eta_{20} + \eta_{02}$$

$$\Phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$\Phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$\Phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$\Phi_5 = (\eta_{30} - 3\eta_{12}) \cdot (\eta_{30} + \eta_{12}) \cdot \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + (3\eta_{21} - \eta_{03}) \cdot (\eta_{21} + \eta_{03}) \cdot \left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]$$

$$\Phi_6 = (\eta_{20} - \eta_{02}) \cdot \left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] + 4\eta_{11}(\eta_{30} + \eta_{12}) \cdot (\eta_{21} + \eta_{03})$$

$$\Phi_7 = (3\eta_{21} - \eta_{03}) \cdot (\eta_{30} + \eta_{12}) \cdot \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + (3\eta_{12} - \eta_{30}) \cdot (\eta_{21} + \eta_{03}) \cdot \left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]$$

independent of
- translation
- scale
- rotation