Bildverarbeitung

20.4.10

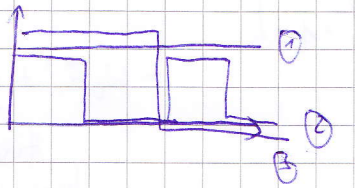
2. Vorlesung

Die Fourier-Matrix und ihre magisch-mystischen Eigenschaften

Letztes Mal:

Hadamard-Transformation $H_N = \begin{pmatrix} H_{N_2} & H_{N_2} \\ H_{N_2} & -H_{N_2} \end{pmatrix}$ $H_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$ Basis-funktionen

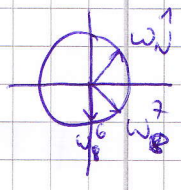
$H B H = \hat{f}$



Fourier-Transformation

$F B F = Y$ $\quad F_N = \left[\omega_n^{ij} \right]$ $\quad i \cdot j$

Einheitswurzel $\omega_n^n = 1 \Leftrightarrow \omega^n = 1$



F_n hat sehr interessante Eigenschaften und Symmetrien.

Konvolutions-theorem

Circular shifts

F_n — symmetrisch $F^T = F$

conjugate flip

shuffle symmetrie

FFT

1. F_n ist symmetrisch

$$F_n = \left[\omega_N^{ij} \right] = \left[\omega_n^{ij} \right]$$

2. conjugate flip

$\vec{f} \in \mathbb{R}^N$

$F_N \vec{f} = \hat{f} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_2^* \\ f_1^* \end{pmatrix}$

Exponenten

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 \\ 0 & 2 & 4 & 6 \\ 0 & 3 & 6 & \\ \vdots & & & \\ 0 & (n-2) & 2(n-2) & 3(n-2) \\ 0 & (n-1) & 2(n-1) & 3(n-1) \end{pmatrix}$$

konjugiert

Wir können uns also die Hälfte der Berechnungen sparen, da der zweite Teil einfach nur konjugiert ist.

Kosinus-Transformation

$$\vec{x} = \begin{pmatrix} x_0 \\ \vdots \\ x_{N-1} \end{pmatrix} \xrightarrow{1.} \vec{x}' = \begin{pmatrix} x_0 \\ \\ x_{N-1} \\ x_{N-1} \\ \\ x_1 \end{pmatrix} \xrightarrow{2.} F_{2N-1}\,\vec{x}' = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_2 \\ w_1^* \end{pmatrix} \vec{x}'$$

$$= w_0 x_0' + \underbrace{(w_1 + w_1^*)}_{\text{Hier fällt überall der Imaginärteil weg! Übrig bleibt quasi nur 2 mal der Realteil in Sinus-Kosinus-Schreibweise der komplexen Zahlen bleibt eine nur Kosinus übrig.}} x_1' + (w_2 + w_2^*) x_2' + \dots$$

$$F\,F^{-1} = I$$

$$F\,F^{*} \overset{?}{=} I$$

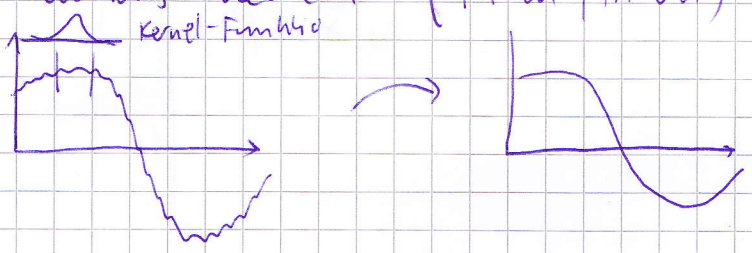
$$= \begin{pmatrix} w_1^{\dagger} \\ w_2^{\dagger} \\ \vdots \\ w_{N-1}^{\dagger} \end{pmatrix} \begin{pmatrix} w_1^* & w_2^* & \dots & w_{N-1}^* \end{pmatrix} = \frac{1}{N} \begin{pmatrix} N & 0 & 0 & \dots \\ 0 & N & & \\ 0 & 0 & N & \\ 0 & & & N \end{pmatrix}$$

~~Schnelle Fourier-Synthese~~

circular shift

$$F_4 \cdot \begin{pmatrix} w_0 & w_1 & w_2 & w_4 \end{pmatrix} = \begin{pmatrix} w^0 & w^0 & w^0 & w^0 \\ w^1 & w^2 & w^4 & w^0 \\ w^2 & w^4 & w^2 & w^0 \\ w^3 & w^6 & w^1 & w^0 \end{pmatrix}$$

Konvolutions-Theorem (Falten, Filtern)



Wichtig für
- Weichzeichnen
- Kantenerkennung

Bildverarbeitung Tutorium

$$\mathcal{F}(\vec{h} * \vec{b}) = \left[\mathcal{F}\begin{pmatrix} w_0 \\ w_{02} \\ \vdots \end{pmatrix} \quad \mathcal{F}\begin{pmatrix} w_{11} \\ w_{12} \\ \vdots \end{pmatrix} \right] \vec{b} = \left[w_0 \circledast \mathcal{F}\vec{h} \quad w_1 \circledast \mathcal{F}\vec{h} \right] \cdot \vec{b}$$

↖ Punktweise Multiplikation

22.4.10

$$\mathcal{F}\vec{h} \circledast \begin{bmatrix} w_0 & w_1 & w_2 & \dots \\ \vdots & \vdots & \vdots & \end{bmatrix} \cdot \vec{b} = \mathcal{F}\vec{h} \circledast \mathcal{F}\vec{b}$$

Konvolutionssatz: $\mathcal{F}(\underbrace{\vec{b} * \vec{h}}_{\text{Kernel}}) = \mathcal{F}(\vec{b}) \circledast \mathcal{F}(\vec{b})$

FFT

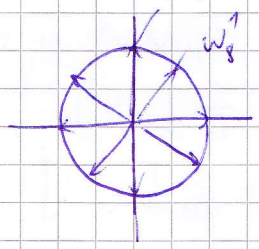
$$\vec{f} = \mathcal{F}\vec{x}$$

$$\vec{f} = \underbrace{\begin{pmatrix} w_0 & w_2 & w_4 \dots & w_1 & w_3 & w_5 \dots \end{pmatrix}}_{\mathcal{F}} \begin{pmatrix} x_0 \\ x_2 \\ x_4 \\ \vdots \\ x_1 \\ x_3 \\ x_5 \\ \vdots \end{pmatrix}$$

Trennung nach geraden und ungeraden Spalten

$$\mathcal{F}^1 = \begin{pmatrix} w_n^0 & w_n^0 & w_n^0 & \dots & w_n^0 & w_n^0 \\ w_n^0 & w_n^2 & w_n^4 & \dots & w_n^n & w_n^3 \\ w_n^0 & w_n^4 & w_n^8 & \dots & w_n^2 & w_n^6 \\ \vdots & & & & & \\ w_n^0 & w_n^{\frac{n}{2}} & w_n^{\frac{n}{2}+2} & \dots & w_n^{\frac{n}{2}n} & w_n^{\frac{n}{2}+3} \\ w_n^0 & w_n^{\frac{n}{2}+2} & w_n^{\frac{n}{2}+4} & \dots & w_n^{\frac{n}{2}+n} & w_n^{\frac{n}{2}+6} \\ w_n^0 & & & & & \end{pmatrix}$$

$w_8^2 = w_4^1$



$$= \begin{pmatrix} \mathcal{F}_{n/2} & \omega_1 \circledast \mathcal{F}_{n/2} \\ \mathcal{F}_{n/2} & -\omega_1 \circledast \mathcal{F}_{n/2} \end{pmatrix}$$

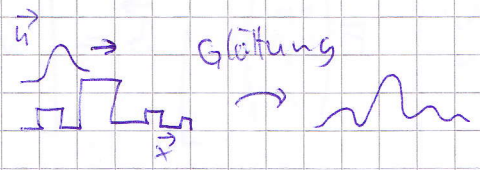
Den Prozess kann man rekursiv anwenden bis $\mathcal{F}_1$ (umsortieren nicht vergessen) oder $\mathcal{F}_2$ $\mathcal{F}_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

Die Summe des Filterkernels muss 1 sein, z.B. können wir Gauß nehmen

Bildverarbeitung
3. Vorlesung
27.4.16

Konvolution

$\vec{h} \otimes \vec{x}$



$$F_n(\vec{h} \otimes \vec{x}) = F_n \vec{h} \cdot F_n \vec{x}$$

komponentenweise

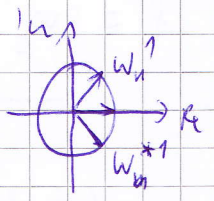
$$F^*(F\vec{h} \cdot F\vec{x}) = F^*(F(\vec{h} \otimes \vec{x})) = \vec{h} \otimes \vec{x}$$

Für die Konvolution direkt braucht man $O(n^2)$, mit dem Umweg über Fourier (über FFT), dann nur $O(n \log n)$.

Für kleine Kernel lohnt es sich allerdings nicht.

$F_n = [w_n^{ij}]$ mit $i = 0, \dots, n-1$, $j = 0, \dots, n-1$

$F_n^{-1} = \frac{1}{n}[w_n^{*ij}]$ (asymmetrisch)



oder man definiert es symmetrisch:

$$F_n = \frac{1}{\sqrt{n}}[w_n^{ij}], \quad F_n^{-1} = \frac{1}{\sqrt{n}}[w_n^{*ij}]$$

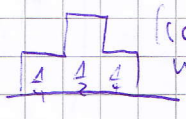
oder $F_n = \frac{1}{\sqrt{n}}[w_n^{*ij}]$, $F_n^{-1} = \frac{1}{\sqrt{n}}[w_n^{ij}]$

Filter

Dreieck-Filter

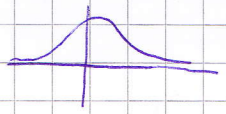
z.B.

oder



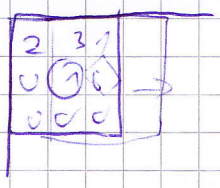
(so somit die mittlere Helligkeit des Bildes erhalten)

Gauß-Filter

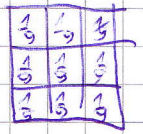


$$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

Jetzt 2D-Kernel



Uniforme Glättung "Abschattung" Soll → für Kantenerkennung



$$G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \quad G_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$$

vertikale Abschattung — horizontale Abschattung

Laplace auch für Kanten

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Konvolutionssatz für 2D

K Kernel, B Bild Seite sind gleichgroß $n \times n$.

$$\mathcal{F}(K \circledast B)\mathcal{F}$$

$$\mathcal{F}\begin{pmatrix} K_0 \odot B_0 & K_0 \odot B_1 & K_0 \odot B_2 & K_0 \odot B_3 \cdots & K_0 \odot B_{n-1} \\ K_1 \odot B_0 & & & & \\ K_2 \odot B_0 & & & & \\ \vdots & & & & \\ K_{n-1} \odot B_0 & & & & \end{pmatrix}\mathcal{F}$$

mit K_i heißt Kernel um i Positionen nach unten verschoben und B_i heißt Bild um i Positionen nach rechts verschoben)

$\odot$ ist die Operation komponentenweise Multiplizieren und dann alles Addieren.

Der Aufwand so ist $O(n^4)$.

$$\mathcal{F}\begin{pmatrix} \square & & & \\ \square & & & \\ \square & & & \\ \vdots & & & \\ \square & & & \end{pmatrix} = \mathcal{F}\begin{pmatrix} K_{00} \cdot B_{00} + K_{01} \cdot B_{01} + \cdots \\ K_{00}^{1} \cdot B_{00} + K_{01}^{1} \cdot B_{01} \\ K_{00}^{11} \cdot B_{00} + \cdots \\ \vdots \\ K_{00}^{n-1} B_{00} + \cdots \end{pmatrix}$$

1 dimensionale Konvolution

komponentenweise Multiplikation ohne Addition

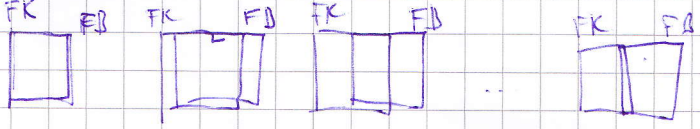
$$= \left[\mathcal{F}K_0 * \mathcal{F}B_{00} + \mathcal{F}K_{01} * \mathcal{F}B_{01} + \cdots \right]$$

$$= \mathcal{F}K * \mathcal{F}B \, \mathbb{1}_n$$

$$\mathbb{1}_n = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}_n$$

$$\mathcal{F}(K \circledast B)\mathcal{F} = \left[\mathcal{F}K * \mathcal{F}B\,\mathbb{1} \;\middle|\; \mathcal{F}K * \mathcal{F}B'\mathbb{1} \;\middle|\; \mathcal{F}K * \mathcal{F}B''\mathbb{1} \;\middle|\; \cdots \right]\mathcal{F}$$

1 dimensionale Konvolution



$= (\mathcal{F}K\mathcal{F}) * (\mathcal{F}B\mathcal{F})$ ← hier ist man nach im Frequenzraum

$$K \circledast B = \mathcal{F}^* (\mathcal{F}K\mathcal{F}) * (\mathcal{F}B\mathcal{F}) \mathcal{F}^*$$

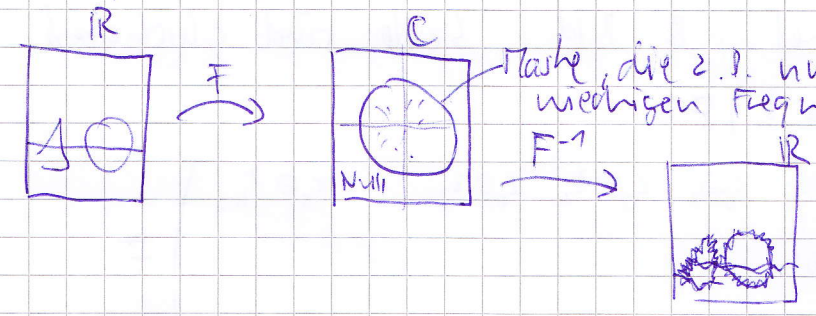
Hier ist der Aufwand $\frac{1}{2}n^2 \log_2 n$, also enorm gespart.

Man kann dass auch auf 3D erweitern..

$$\mathcal{F}(K_{3D} \overset{F}{\circledast} B_{3D})\mathcal{F} = \mathcal{F}K_{3D}\mathcal{F} \overset{F}{\cdot} \mathcal{F}B_{3D}\mathcal{F}$$

Anwendung:

1) Filtering im Frequenzraum (K gegeben)
2) Filtering im Frequenzraum (ohne K gegeben)



Damit man im reellen landet, muss man entweder den imaginären Teil wegschmeißen, oder den absoluten Wert nehmen

(Das wäre ein Tiefpassfilter)

Der Hochpassfilter wäre die Maske invertiert. So kann man dann natürlich beliebige Frequenzfilter machen.

3) Dekonvolution

$$\mathcal{F}(K \circledast B)\mathcal{F} = \mathcal{F}K\mathcal{F} \cdot \mathcal{F}B\mathcal{F}$$

(B_G)

$$\mathcal{F}B\mathcal{F} = \mathcal{F}B_G\mathcal{F} \;/\; \mathcal{F}K\mathcal{F}$$

komponentenweise Division

$$B = \mathcal{F}^*\left(\frac{\mathcal{F}B_G\mathcal{F}}{\mathcal{F}K\mathcal{F}}\right)\mathcal{F}^*$$

Dazu muss man natürlich den Kernel kennen.

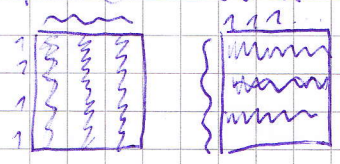
Man kann auch blinde Dekonvolution machen, wo man aus dem geglätteten Bild versucht auf den Kernel zu schließen.

dwear-
eitung

. Vorlesung
. 5. 2016

Wir wollen Muster suchen.

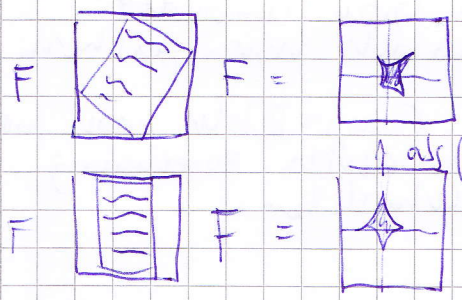
Einfache Muster sind

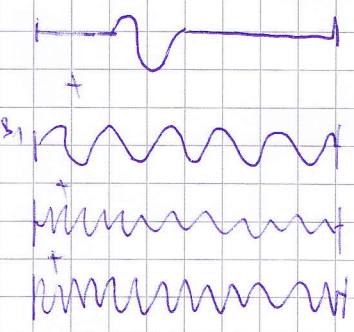


Wir wollen z.B. die Drehung eines Schriftdokuments beim Scannen wissen.

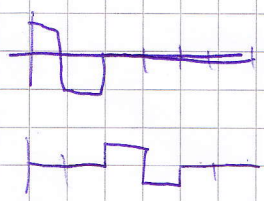
Ein simpler Ansatz wäre ein Histogramm über die "Tinten"-Pixel in allen Spalten zu machen.

Bei schiefen Dokumenten muss man das Koordinatensystem drehen und mehrere Histogramme erstellen.



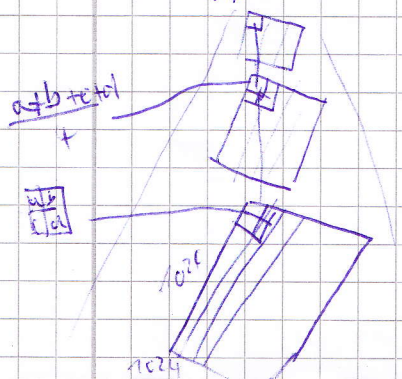


Lokale Basis



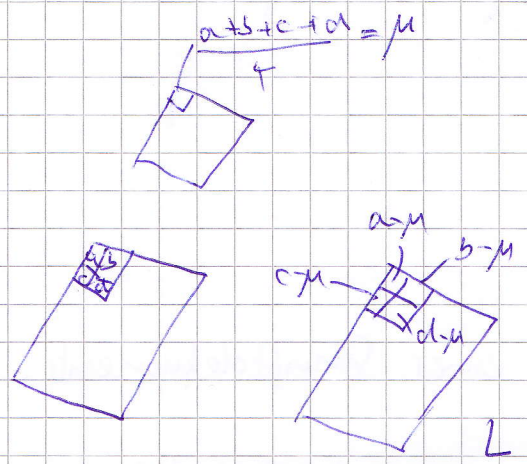
Wavelet

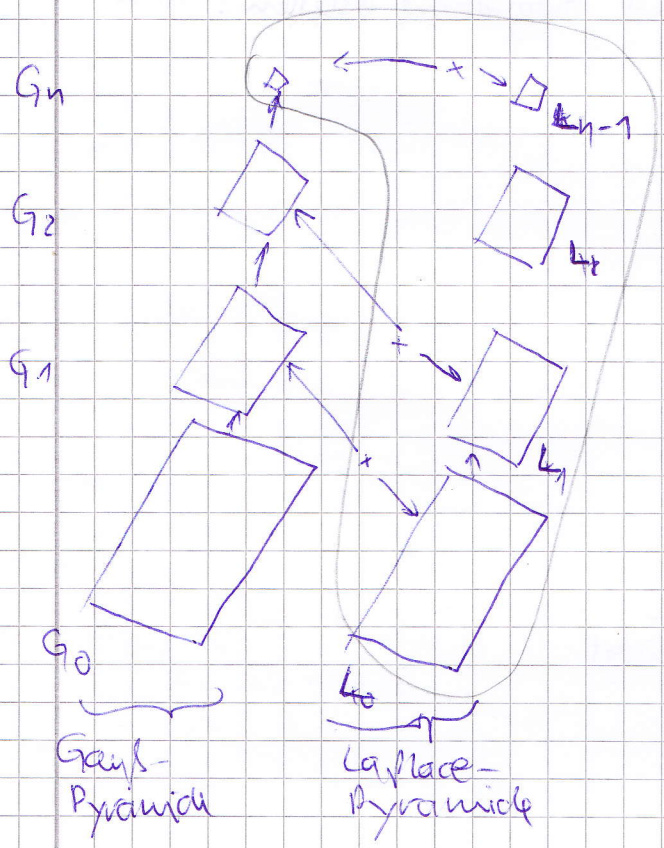
Bildpyramiden



Mittelung nach oben, das nennt man auch Gauß-Pyramiden.

Das ist rekursiv gehalten.





Das Gauß umwandeln muss ich nur speichern.

Die Gauß-Bilder kann man dann durch Berechnung wiederherstellen.

59	43	11	27	95	45	37	13

51	19	45	25	8	-8	0	12

$\frac{a+b}{2}$ $\frac{a-b}{2}$

35	35	16	10	8	-8	0	12

35	0	16	10	8	-8	0	12

Wavelet-Transformation ①

Rücktransformation zum Bild

35	0	16	10	8	-8	0	12

+0 / -0

35	35	16	10	8	-8	0	12

+16 / -16 / +10 / -10

51	19	45	25	8	-8	0	11

+8 / -8 / -8 / +8 / +0 / -0 / +11 / -11

59	43	11	27	45	36	14

Das oberste Bild ist kleiner (da kleinere Zahlen) und speichert den Kontrast.

Zudem ist die Transformation linear (da man nur Addition, Subtraktion und Multiplikation verwendet).

Es gibt also auf jedenfall eine Matrix, die die Transformation macht.

$$\vec{w} = W\vec{x}$$

Wie findet man nun W und die Inverse W^{-1}?

$$W = [\vec{w}_0 \; \vec{w}_1 \; \dots \; \vec{w}_{n-1}]$$

$$\vec{x} = W^{-1} \begin{pmatrix} \vec{w} \\ 0 \\ 0 \\ \vdots \end{pmatrix}$$

wir kennen allerding W^{-1} nicht, aber eine andere Methode um von $\vec{w}$ zu $\vec{x}$ zu kommen.

1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1

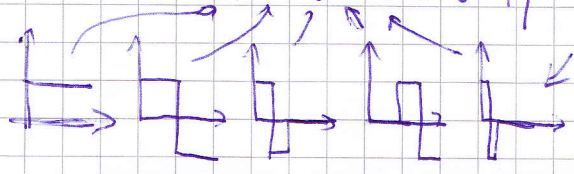
1. Spalte → die Matrix

$$W^{-1} \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{pmatrix}$$

0	1	0	0	0	0	0	0
1	-1	0	0	0	0	0	0
1	1	-1	-1	0	0	0	0
1	1	-1	-1	1	1	-1	-1

2. Spalte → der Matrix

$$W^{-1} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 \\ 1 & -1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 1 & 0 & 0 & -1 & 0 \\ 1 & -1 & 0 & -1 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & -1 & 0 & 0 & 0 & -1 \end{pmatrix}$$



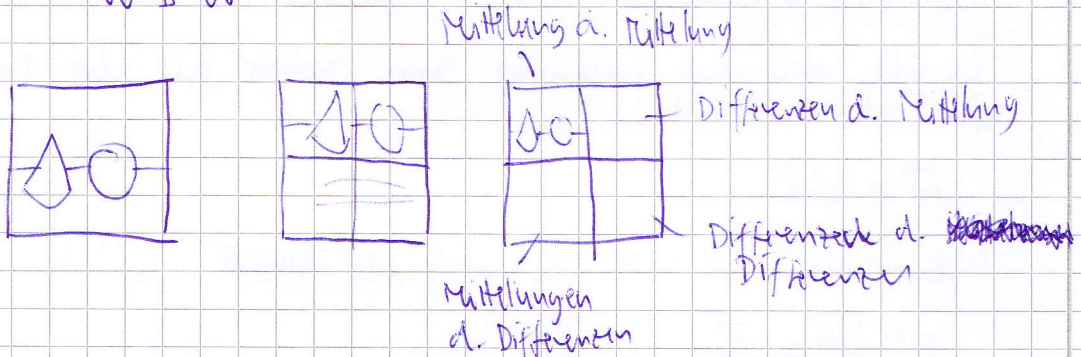
$$W = \begin{pmatrix} 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 \\ 1/8 & 1/8 & 1/8 & 1/8 & -1/8 & -1/8 & -1/8 & -1/8 \\ -1/4 & 1/4 & -1/4 & -1/4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/4 & 1/4 & -1/4 & -1/4 \\ 1/2 & -1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & -1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & -1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & -1/2 \end{pmatrix}$$

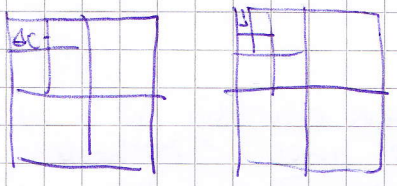
Dieses Spezielle Wavelet nennt man auch Haar-Transformatio

Anwendung

Man kann das jetzt auch wieder 2D machen

$W B W^T$





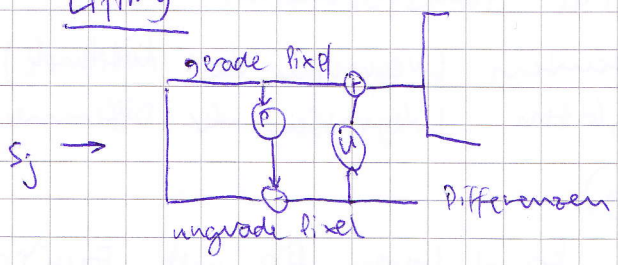
Hier könnte ich jetzt Thresholding machen z.B. kleine Werte auf 0 setzen) um damit Kompression zu machen.

Übersetzung

. Vorlesung

1. 5. 2020

Lifting



P = Prediction - Funktion
U = Update - Funktion

Inverse Transformation:

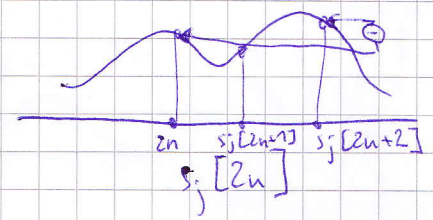
$$\begin{cases} s_j[2n] = s_{j-1}[n] - \frac{1}{4}\left(d_{j-1}[n-1] + d_{j-1}[n]\right) \\ s_j[2n+1] = d_{j-1}[n] + \frac{1}{2}\left(s_j[2n] + s_j[2n+2]\right) \end{cases}$$

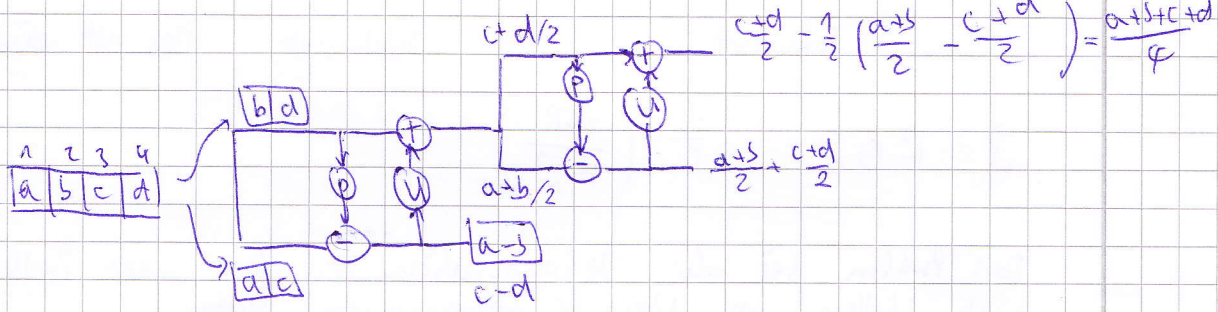
Transformation:

$$\begin{cases} d_{j-1}[n] = s_j[2n+1] - \underbrace{\frac{1}{2}\left(s_j[2n] + s_j[2n+2]\right)}_{P} \\ s_{j-1}[n] = s_j[2n] + \underbrace{\frac{1}{4}\left(d_{j-1}[n-1] + d_{j-1}[n]\right)}_{U} \end{cases}$$

Der Index j gibt jeweils die Ebene an, auf der wir arbeiten. Bei 1024 Pixel würde man bei Ebene 10 anfangen ($2^{10} = 1024$).

CDF-Wavelet (CDF(2,2))





Für $d_{j-1}[n] = s_j[2n+1] - s_j[2n]$

$s_{j-1}[n] = s_j[2n] + \frac{1}{2} d_{j-1}[n]$

Haar-Wavelet (wie altes mal)

Alle Wavelet-Transformationen die in dieses Lifting-Schemen passen, sind schnell berechenbar, da sich das Bild ja in jeder Stufe halbiert.

CDF(2,4):

$$s_{j-1}[n] = s_j[2n] - \frac{1}{64}\left(3 d_{j-1}[n-2] - 19 d_{j-1}[n-1] - 19 d_{j-1}[n] + 3 d_{j-1}[n+1]\right)$$

$$d_{j-1}[n] = s_j[2n+1] - \frac{1}{2}\left(s_j[2n] + s_j[2n+2]\right)$$

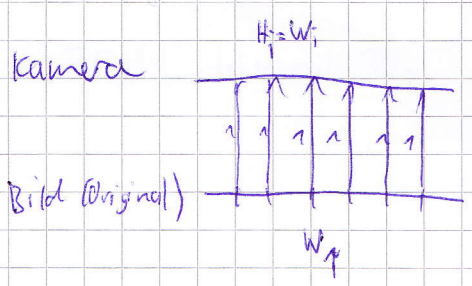
Anwendungen der Wavelet-Transformation

WT
- Filtering
  - Kanten (Wegwerfen der Mittelwerte)
  - Glätten (Wegwerfen der Differenzen)
- SWT (Lifting)

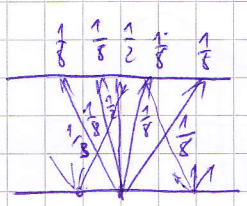
Gehen auch Konvolutionen die mit Fourier?
Nein!

JPEG benutzt zur Kompression Fourier, JPEG 2000 benutzt dazu Wavelets.

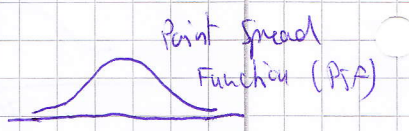
Konvolution / Dekonvolution



Fokussiert



Defokussiert



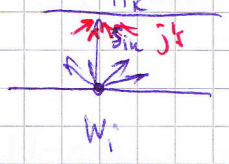
$H_i = \frac{1}{8} W_{i-2} + \frac{1}{4} W_{i-1} \cdots$

Das ist eine Konvolution

$$\vec{H} = \vec{K} \otimes \vec{x} \qquad \vec{F}^{-1}\left(\frac{F\vec{d}}{F\vec{K}}\right)$$

Das Problem bei der Dekonvolution ist, dass man Probleme mit Nullen im Nenner bekommen kann.

Lucy-Richardson



Helligkeit $[0..1]$

$\sum W_i = W$ (Summe der Helligkeiten soll erhalten bleiben)

$$P(W_i \mid H_k) = \frac{P(H_k \mid W_i) \cdot P(W_i)}{\sum_j P(H_k \mid W_j) \cdot P(W_j)}$$

Bayes-Formel

$P(H_k \mid W_i) = S_{ik}$ (Verbindungsstärke)

Im Zähler steht also das Licht von i nach k
Im Nenner steht dann das Licht, das von allen Pixeln auf H_k strahlt.
$P(W_i \mid H_k)$ ist also der Anteil von W_i an der Helligkeit von H_k (prozentual).

$P(W_i)$ möchten wir nun bestimmen.

$$P(W_i) = \sum_{\substack{k \\ \text{alle von} \\ i \text{ bestimmten} \\ \text{Pixel}}} \left(\frac{P(H_k|W_i) \cdot P(W_i)}{\sum_j P(H_k|W_j) \cdot P(W_j)} \cdot P(H_k) \right)$$

Das Problem ist, dass das Gesuchte $P(W_i)$ auch auf der rechten Seite auftaucht...

Wir machen nun ein iteratives Verfahren daraus (EM-Algorithmus):

$$P(W_i, r) = P(W_i, r-1) \cdot \sum_k \frac{P(H_k|W_i) \cdot P(H_k)}{\sum_j P(H_k|W_j) \cdot P(W_j)} \qquad r = 0, 1, 2 \ldots \text{ Iteration}$$

$$P(W_i) = \frac{W_i}{W} \quad , \quad P(H_k) = \frac{H_k}{W} \qquad P(H_k|W_i) = s_{ik}$$

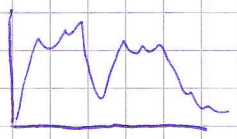
$$\frac{W_{i,r}}{W} = \frac{W_{i,r-1}}{W} \left(\sum_k \frac{s_{ik} \cdot \frac{H_k}{W}}{\sum_j s_{jk} \cdot \frac{W_j}{W}} \right)$$

$$W_{i,r} = W_{i,r-1} \sum_k \frac{s_{ik} \cdot H_k}{\sum_j s_{jk} \cdot W_j}$$

Das macht man so lange, wie man lustig ist. Meistens bis die Differenz von $W_{i,r}$ und $W_{i,r-1}$ klein genug ist.

diver-
seitung
Vorlesung
5.2016

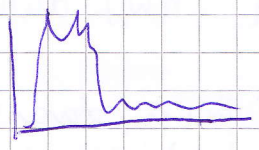
Histogramm equalization



Dynamic range

Histogramm zeigt uns, wie die Helligkeiten im Bild verteilt sind

Beispiel:



zu wenig Kontrast

Es geht darum, dass man jetzt versucht wieder den bestmöglichen Kontrast zu bekommen.

Aus dem Histogramm berechnen wir dann die kumulierte Wahrscheinlichkeiten (CDF)

Histogramm → CDF

$$P_x(i) = P(x = i) = \frac{n_i}{n}$$

$$cdf_x(i) = \sum_{j=0}^{i} P_x(j)$$

Wir suchen eine pixelweise Transformation $T: y = T(x)$ so dass

$cdf_y(i) = ki$, d.h. die Verteilung von y wird linear

⋮ (ausgelassener Beweis)

$y = T(x) = cdf_x(x)$

$y' = y(\max\{x\}, \min\{x\}) + \min\{x\}$

Beispiel:

$$\begin{bmatrix} 52 & 55 & 61 & \dots & \\ 65 & & & & \\ 62 & & \ddots & & \\ \vdots & & & 154 & \\ \vdots & & & & \\ \vdots & & & & \\ & & & & 94 \end{bmatrix}$$

Das Histogramm wird

Wert	Anzahl
52	1
55	3
62	2
⋮	⋮
154	1

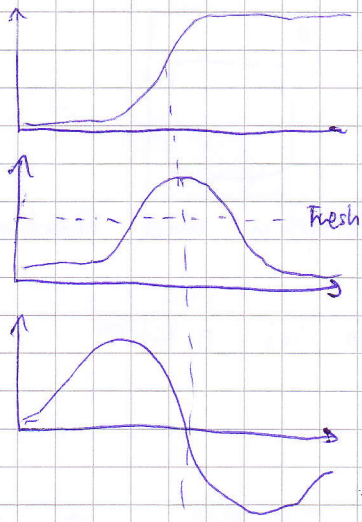
$\xrightarrow{cdf}$

52	1
55	4
62	6
⋮	⋮
154	64

Edge - Detection → Search based / Zero Crossing

T: (RGB / Gray → Binary)

Canny - Edge - Detection (1986)



kante schwer zu lokalisieren

1. good detection
2. good localization
3. minimal response

Bei Canny werden diese drei Eigenschaften erfüllt. Die Schritte sind wie folgt:

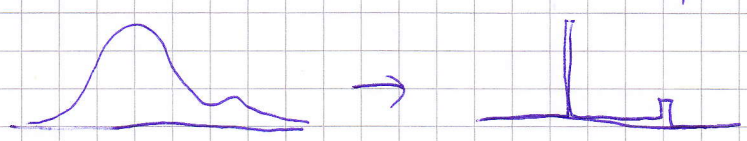
1. Noise reduction z.B. durch Gaußfilter

$$\frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \cdot \vec{B}$$

2. Derivation mit Sobel

$(S_x, S_y) \rightarrow$ Polar $\left(\sqrt{S_x^2 + S_y^2}, \operatorname{atan}\left(\frac{G_y}{G_x}\right)\right)$

3. Non-maxima Suppression (directed)



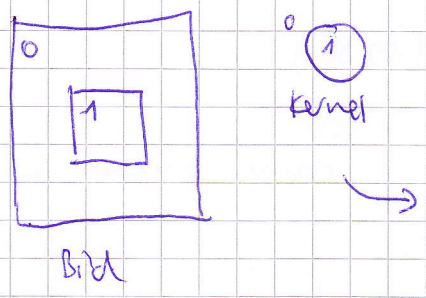
4. Thresholding and tracking

Verwendung von zwei unterschiedlichen Schwellwerten. Zunächst wird mit einem hohen Schwellwert T_H gefiltert, danach mit T_L, wobei bei T_L man nur die Kanten nimmt, wenn dort in der Nachbarschaft mit T_H was für

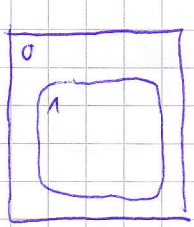
Morphology Operators

Wir arbeiten auf Binärbildern

- Dilation



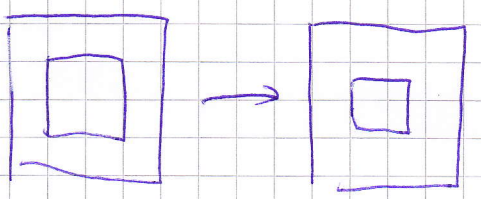
$$A \oplus B = \bigcup_{b \in B} A_b$$



Mit Dilation kann man Löcher stopfen und Flächen vergrößern

- Erosion Bild und Kernel wie bei Dilation

$$A \ominus B = \bigcap_{b \in B} A_b$$



Gegenteil von Dilation

Durch $(A \oplus B) \ominus B) \approx B$ kann man z.B. Rauschen entfernen

- Opening $A_o B = (A \ominus B) \oplus B$

$(A_o B) \cdot B = A_o B$ $\quad A \circ B \subseteq A$

- Closing $(A \oplus B) \ominus B = A \bullet B$

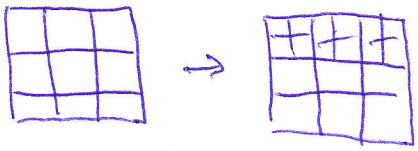
$(A \bullet B) \bullet B = A \bullet B$

$A \subseteq A \bullet B$

Bildverarbeitung, 25.5.2010

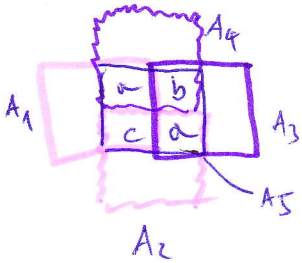
Superresolution

Man möchte bessere Auflösungen erreichen, d.h. ein Originalbild möchte man verfeinern



Mehrere Ansätze:

1) Man benutzt mehrere Bilder



Die Werte für die Pixel a,b,c,d sind dann Funktionen aus den Bildern A_1 bis A_5.

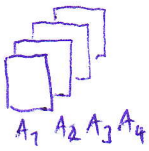
$a = f(A_1, A_2, A_3, A_4, A_5) = \alpha_1 A_1 + \alpha_2 A_2 + \alpha_3 A_3 + \alpha_4 A_4 + \alpha_5 A_5$

$b =$

$c =$

$d =$

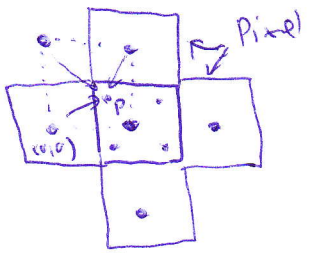
2) Sequenz von Bildern (z. B. Film)

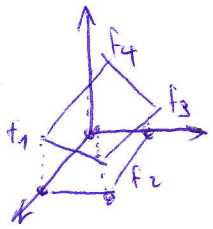


Hier muss man zwischen mehreren Bildern Gemeinsamkeiten finden usw. ist schwer.

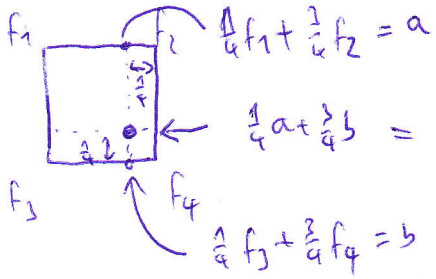
3) Interpolation

- Bilinear
- Bikubisch
- Bayes-Verfahren





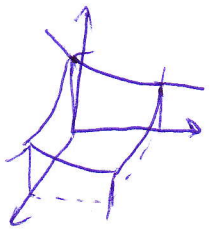
Bei Bilinear sucht man die beste Ebene für f_1, f_2, f_3, f_4 und berechne Helligkeit von P



$\frac{1}{16} f_1 + \frac{3}{16} f_2 + \frac{3}{16} f_3 + \frac{9}{16} f_4$

Bikubisch

Wir nehmen hier keine Ebene, sondern Polynome.



Wir brauchen die Ableitungen der 4 Punkte um zu wissen, in welche Richtung die kurven gehen.

Wir brauchen Ableitungen in x-Richtung, y-Richtung und Diagonal. Wir brauchen also 16 Parameter.

$$f(x,y) = \underset{1}{a_{00}} + \underset{x}{a_{01}} + \underset{x^2}{a_{02}} + \underset{x^3}{a_{03}} + \underset{1}{a_{10}} + \underset{y}{a_{11}} + \underset{y^2}{a_{12}} + \underset{y^3}{a_{13}} + \underset{1}{a_{20}} + \underset{xy}{a_{21}} + \underset{x^2y^2}{a_{22}} + \underset{x^3y^3}{a_{23}}$$

$$\frac{\partial f}{\partial x} = 0 \quad 1 \quad 2x \quad 3x^2 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad y \quad 2xy^2 \quad 3x^2y^3$$

$$\frac{\partial f}{\partial y} = 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 2y \quad 3y^2 \quad 0 \quad x \quad x^2 2y \quad x^3 3y^2$$

$$\frac{\partial}{\partial x}\frac{\partial f}{\partial y} = 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 2x \; 2y \quad 3x^2 \; 3y$$

Polynommatrix — Angaben aus dem Bild

$$F\vec{a} = \vec{f}$$

Parameter

Approximation $\frac{\partial f}{\partial x} = f(x+1) - f(x)$

$$F \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 1 & 0 \; 0 \; \cdots \; 0 \\ 1 & 0 & 0 & 0 & 1 \; 1 \; 1 \; 0 \cdots 0 \\ \vdots & & & & \\ 0 & 1 & 0 & 0 & \cdots \; 0 \\ 0 & 1 & 2 & 3 & 0 \; \cdots \; 0 \\ \vdots & & & & \\ 0 & 0 & 0 & 0 & 1 \; 0 \cdots 0 \\ \vdots & & & & \\ 0 & 0 & 0 & 0 & 0 \; 0 \; 0 \; 1 \; 0 \cdots \end{pmatrix} \begin{pmatrix} a_{00} \\ a_{10} \\ a_{20} \\ \vdots \end{pmatrix} = \begin{pmatrix} f(0,0) \\ f(1,0) \\ f(0,1) \\ f(1,1) \\ \frac{\partial f(0,0)}{\partial x} \\ \frac{\partial f(1,0)}{\partial x} \\ \vdots \end{pmatrix}$$

16 Zeilen

Hier fehlt irgendwie 4 Spalten ... richtig sie nächstes

$$\vec{a} = F^{-1}\vec{f}$$

Man kann seine F-Matrix auch noch um sehrige zeilen erweitern (z.B. 2. Ableitungen)

$$f(x,y) = a_{00} + a_{10} + a_{20} + a_{30} + a_{01} + a_{11} + a_{21} + a_{31} + a_{02} + a_{12} + a_{22} + a_{32} + a_{03} + a_{13} + a_{23} + a_{33}$$

$$1 \quad x \quad x^2 \quad x^3 \quad y \quad xy \quad x^2y \quad x^3y \quad y^2 \quad xy^2 \quad x^2y^2 \quad x^3y^2 \quad y^3 \quad xy^3 \quad x^2y^3 \quad x^3y^3$$

$$\frac{\partial f}{\partial x} = 0 \quad 1 \quad 2x \quad 3x^2 \quad 0 \quad y \quad 2xy \quad 3x^2y \quad 0 \quad y^2 \quad 2xy^2 \quad 3x^2y^2 \quad 0 \quad y^3 \quad 2xy^3 \quad 2x^2y^3$$

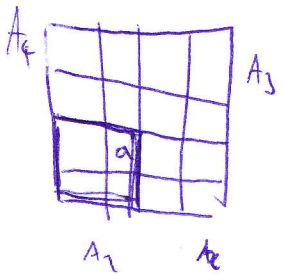
$$\frac{\partial f}{\partial y} = 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad x \quad x^2 \quad x^3 \quad 2y \quad 2yx \quad 2yx^2 \quad 2yx^3 \quad 2y^2 \quad 2y^2x \quad 2y^2x^2 \quad 2y^2x^3$$

$$\frac{\partial}{\partial x}\frac{\partial f}{\partial y} = 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 2x \quad 3x^2 \quad 0 \quad 2y \quad 4yx \quad 6yx^2 \quad 0 \quad 2y^2 \quad 4y^2x \quad 6y^2x^2$$

Eventuell ist es dann nicht mehr invertierbar, da hilft dann

$$F^T F \vec{a} = F^T \vec{f}$$

$$\vec{a} = (F^T F)^{-1} F^T \vec{f} \qquad \text{LSQ}$$



$$a = \alpha_1 A_1 + \alpha_2 A_2 + \alpha_3 A_3 + \alpha_4 A_4$$

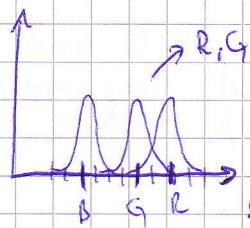
$$\begin{pmatrix} a_1 \\ a_2 \\ \vdots \end{pmatrix} = \begin{vmatrix} & & & \\ \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\ & & & \end{vmatrix}$$

Bildverarbeitung

Vorlesung

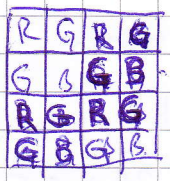
1.6.2020

Farbe

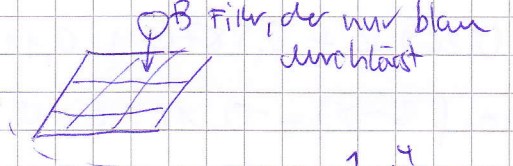


Farbe ist nicht die Reihenfolge der verschiedenen Wellenlängen, sondern die Kombination aus den Messungen für R, G und B durch die verschiedenen Rezeptoren auf der Netzhaut.

Dies wird auch auf der Kamera simuliert



Bayer-Filter

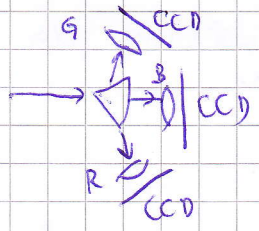


Wir müssen interpolieren: $R = \frac{1}{4}\sum_{1}^{4} R_i$

$G = \frac{1}{4}\sum_{1}^{4} G_i$

$\Rightarrow B =$

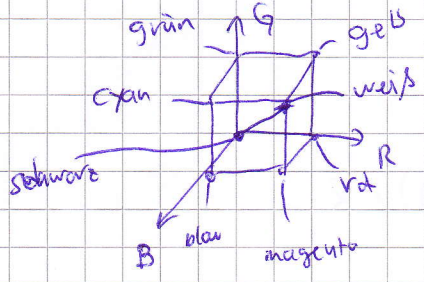
Bei 3CCD-Kameras hat man drei Chips, die wirklich für jeden Pixel die Farben nehmen.



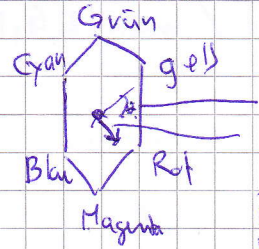
Bei Bayer bekommt man durch die Interpolation an den Kanten falsche Farben.

Da Farben subjektiv sind, definiert man verschiedene Farbräume um unterschiedliche Informationen zu bekommen.

RGB-Farbraum



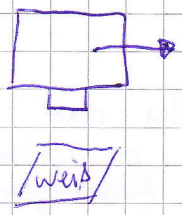
Jede Farbe kann man hier als einen Vektor definieren. Wichtig ist nicht so sehr, wie groß die Werte der einzelnen Komponenten sind, sondern die Richtung, also das Verhältnis von R, G und B.



Farbigkeit (Hue)

Sättigung ist der Abstand

Zusätzlich zur Farbigkeit und Sättigung benötigt man noch die Intensität, das ist der Abstand zum Ursprung.



Weißabgleich: Parameter für R, G und B justieren, damit was weißes wirklich weiß wird.

$\alpha R, \beta G, \gamma B$

YUV - Farbraum

Lineare Transformation vom RGB-Farbraum.

$$Y = 0{,}229\,R + 0{,}587\,G + 0{,}114\,B$$

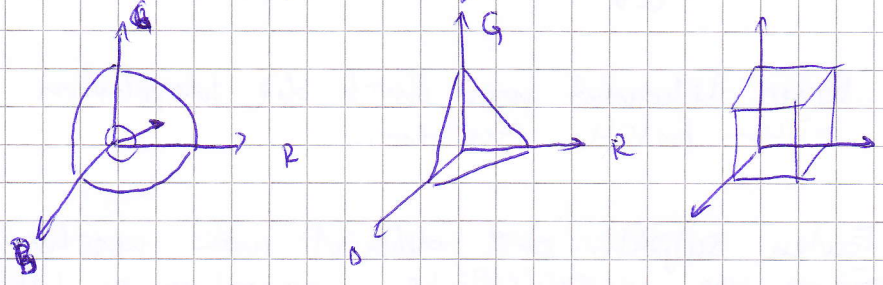
$$U = c_1 (B - Y) = -0{,}169\,R - 0{,}332\,G + 0{,}5\,B + 128$$

$$V = c_2 (R - Y) = 0{,}5\,R - 0{,}419\,G - 0{,}0813\,B + 128$$

YUV wurde für Fernsehen entwickelt, damit frühe die Leute, die nur schwarz-weiß empfangen haben, auch noch was sehen, denn diese konnten einfach nur den Y-Kanal verwenden (Intensität).

G hat so eine hohe Gewichtung, da wir das am besten sehen können.

HSV - Farbraum (HSI)



$$I_1 = \sqrt{R^2 + G^2 + B^2}$$

$$I_2 = (R + G + B) / 3$$

$$I_3 = \max(R, G, B)$$

$$I_4 = (\max(R, G, B) + \min(R, G, B)) / 2$$

$$S_1 = (\max(R, G, B) - \min(R, G, B)) / \max(R, G, B)$$

$$S_2 = \frac{\max(R,G,B) - \min(R,G,B)}{\max(R,G,B) + \min(R,G,B)}$$

$$S_3 = 1 - 3\,\frac{\min(R,G,B)}{R + G + B}$$

S_4 = Abstand $((R, G, B), (1, 1, 1))$ ← Abstand zur Geraden

Jetzt kann man sich aus den verschiedenen Definitionen seinen eigenen Farbraum zusammenbasteln.

$$H_1 = \begin{cases} \left(0 + \frac{G-B}{max-min}\right)\cdot 60 & \text{wenn } R = max \\ \left(2 + \frac{B-R}{max-min}\right)\cdot 60 & \text{wenn } G = max \\ \left(4 + \frac{R-G}{max-min}\right)\cdot 60 & \text{wenn } B = max \end{cases}$$

Dies ist quasi eine Interpolation von H_2

$max = max(R, G, B)$
$min = min(R, G, B)$

$$H_2 = \operatorname{atan}\left(\frac{v_1}{v_2}\right)$$

$$HSI = I_2, S_3, H_2$$

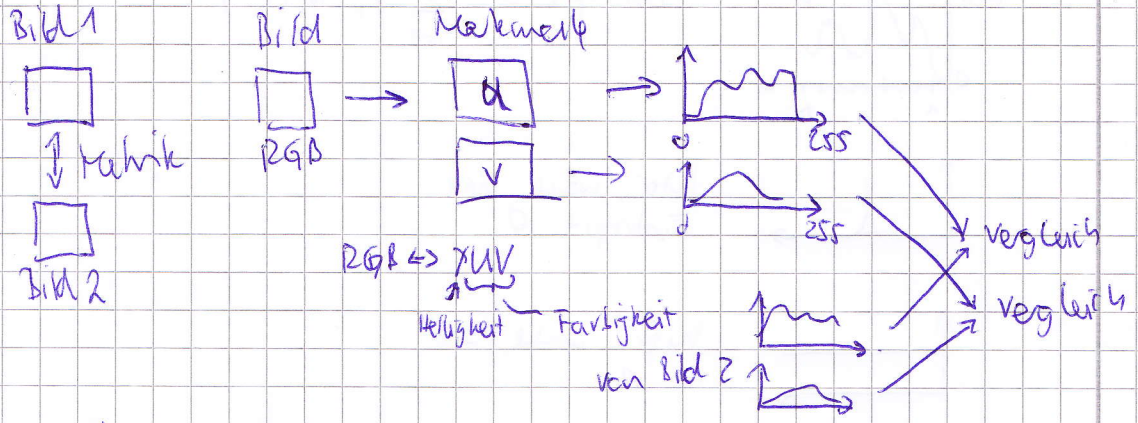
$$HSI = I_2, S_4, H_2$$

$$HSV = I_3, S_1, H_1$$

Bildverarbeitung

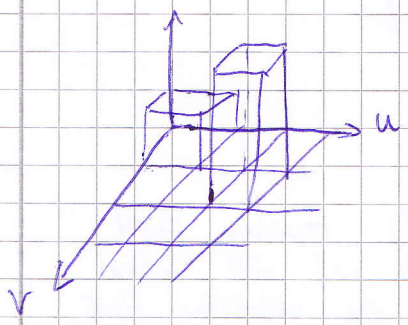
Vorlesung 15.6.2016

Histogramme



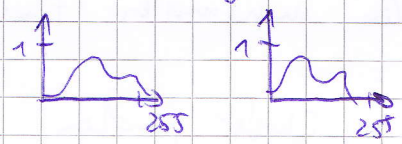
Man will einfach mehrere Bilder vergleichen können (z.B. ähnliche Szenen zu sehen) und das möglichst einfach. Das könnte man z.B. über Histogramme machen. Man nimmt an, dass ähnliche Bilder ähnliche Histogramme liefern.

2D-Histogramm

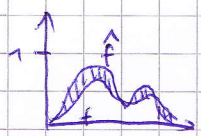


Bessere Korrelation als nur 1D.

Vergleich von Histogrammen

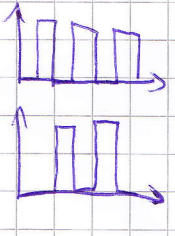


1) Metrik: Euklidischer Abstand



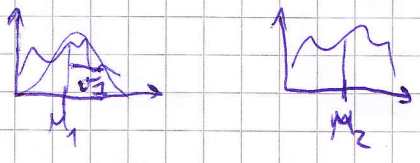
$$\text{Abstand} = \left(\sum_{i=1}^{N} \left(f(i) - \hat{f}(i) \right)^2 \right)^{\frac{1}{2}}$$

Nachteil:



Diese Histogramme sind optisch sehr ähnlich, aber der euklidische Abstand ist hier sehr groß

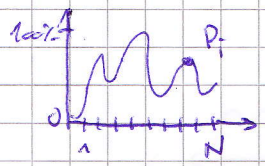
2) Mittelwert und Standardabweichung



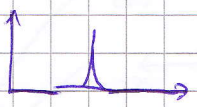
$$D(I,J) = \frac{|\mu_1 - \mu_2|}{\sigma(\mu)} + \frac{|\sigma_1 - \sigma_2|}{\sigma(\sigma)}$$

Streuung der Mittelwerte in einer Datenbank

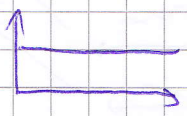
3) KL - Divergenz



$$\text{Entropie} = -\sum_{i=1}^{N} p_i \ln(p_i)$$



hier wäre die Entropie 0



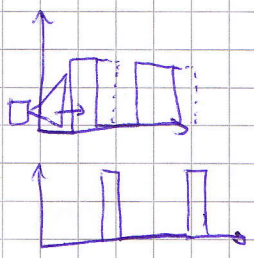
hier wäre sie 1

$$D(P,Q) = \sum_{i=1}^{N} p_i \ln\left(\frac{p_i}{q_i}\right) = \sum_{i=1}^{N} p_i (\ln p_i - \ln q_i)$$

Das ist unsymmetrisch. Man kann es wie folgt symmetrisch machen:

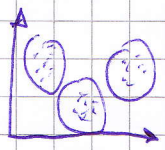
$$D'(P,Q) = \frac{1}{2}\left(D(P,Q) + D(Q,P)\right)$$

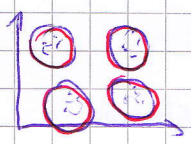
4) Earth - Movers - Metrik



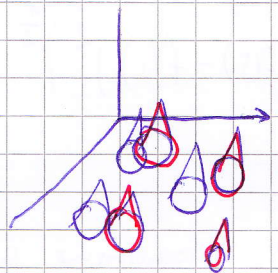
Man stellt sich das als Haufen Erde vor und überlegt sich, wie viel Arbeit es ist, die Erde so zu verteilen, dass sie das andere Histogramm entspricht.

Man sucht den minimalen Fluss.





Cluster finden



$$D(I,J) = \frac{\sum_{ij} g_{ij}\, d_{ij}}{\sum_{ij} g_{ij}}$$

g_{ij} = optimaler Fluss von i nach j

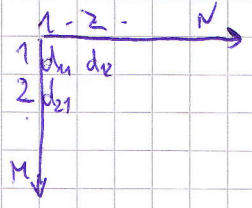
d_{ij} = Abstand zwischen i und j

Wie berechnet man nun den optimalen Fluss?

H_1 $i = 1, 2, \dots, N$ Cluster mit $p_1, p_2, \dots, p_N$ als Verteilung

H_2 $i = 1, \dots, M$ Cluster mit $q_1, q_2, \dots, q_M$ als Verteilung

d_{ij} = Abstandsmatrix



wobei $d_{ij} = d_{ji}$ ist

Wir brauch ein paar Bedingungen:

$g_{ij} \geq 0$

Min $\sum_{ij} g_{ij} d_{ij}$

$\sum g_{ij} = \min(\sum p_i, \sum q_i)$

$\sum g_{ij} \leq \sum p_i$

$\sum g_{ij} \leq \sum q_i$

Wenn die Histogramme komplett sind: $\sum p_i = 1$, $\sum q_i = 1$

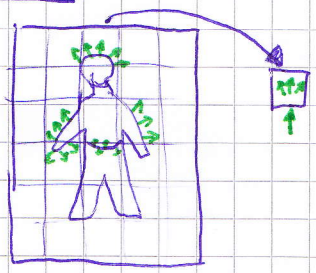
$g_{ij} \geq 0$

$\sum g_{ij} = 1$

Min $\sum g_{ij} d_{ij}$

Das ist ein Problem der linearen Programmierung.
Eins ist z.B. das Simplex Verfahren

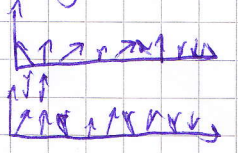
HOG - Histogram Oriented Gradients



Summe der Gradienten

Die Gradienten bekommt man natürlich über Kantenoperatoren

Wie vergleicht man nun wieder zwei solche Histogramme?



$$d(\vec{a}, \vec{b}) = \frac{\vec{a}\,\vec{b}}{|\vec{a}|\,|\vec{b}|}$$

$$D(F, G) = \sum_{i=1}^{N} \left(\frac{\vec{a}_i \cdot \vec{b}_i}{|\vec{a}_i| \cdot |\vec{b}_i|} \right)$$