Industrielle Bildverarbeitung

Prof. Dr. Mark Ross

ross@hs-koblenz.de

SS 2018
Stand: 18. März 2018

Normale Slides mit Hyperlinks

Modalitäten

Modul:	E514 – Industrielle Bildverarbeitung
	Technisches Wahlpflichtfach Master Systemtechnik, 5 ECTS, 4 SWS
Kontakt:	ross@hs-koblenz.de
Vorkenntnisse:	C++–Programmierung
Prüfungssnachweis:	Klausur (90 min, keine Hilfsmittel)
Studienleistung:	Anwesenheit, erfolgreiche Bearbeitung der Programmieraufgaben
Material:	Vorlesungsskript, Übungen: www.hs-koblenz.de/ross

Literatur

[GIMP] *GIMP - GNU Image Manipulation Program*
www.gimp.org/downloads/

[Ste05] R. Steinbrecher, *Bildverarbeitung in der Praxis*, Oldenburg, 2005,
www.rst-software.de/dbv/download.html

[Pau01] D. Paulus, *Aktives Bildverstehen*, Der Andere Verlag, 2001,
www.der-andere-verlag.de/buecher/paulus.pdf

Inhalt

1. Einleitung
2. Bildvorverarbeitung
3. Farbe
4. Segmentierung
5. Morphologie
6. Kantendetektion
7. Merkmalsextraktion
8. Klassifikation

1. Einleitung

1. Einleitung
1.1 Einleitung
1.2 Implementierung und Tools
1.3 Statistische Eigenschaften von Bildern

Abgrenzung zu verwandten Gebieten

Bilderkennen (Bildinterpretation, engl. Computer Vision) Prozess zur automatischen Auswertung/Analyse von Bildern (oder Bildfolgen, Stereobildern oder 3D-Bildern)

Wesentliche Aufgabe ist Identifikation und Lokalisation von Objekten in einer Szene

Objekte bestehen aus Merkmalen (z.B. Form, Farbe) und Beziehungen zwischen ihnen (z.B. Abstand, rel. Position)

Bildverarbeitung *automatische* Manipulation von digitalen Bildern, z.B. Konvertierung oder Verbesserung für folgende Analyse, effizientere Übertragung o.ä.

Bildbearbeitung *manuelle* Manipulation von digitalen Bildern

Computergraphik Generierung von Bildern oder Überlagerung von künstlichen Bildinhalten
Auch zur Visualisierung von Ergebnissen der Bilderkennung

Mustererkennung ist nicht auf bildhafte Information beschränkt

Künstliche Intelligenz Bilderkennen ist ein Teilgebiet der KI

Anwendungen

Qualitätskontrolle Fehlererkennung

Produktion Bestückung, Sortierung, Überwachung

Kriminologie Fingerabdrücke, Portraitauswertung

Fernerkundung Auswertung von Satellitenbildern

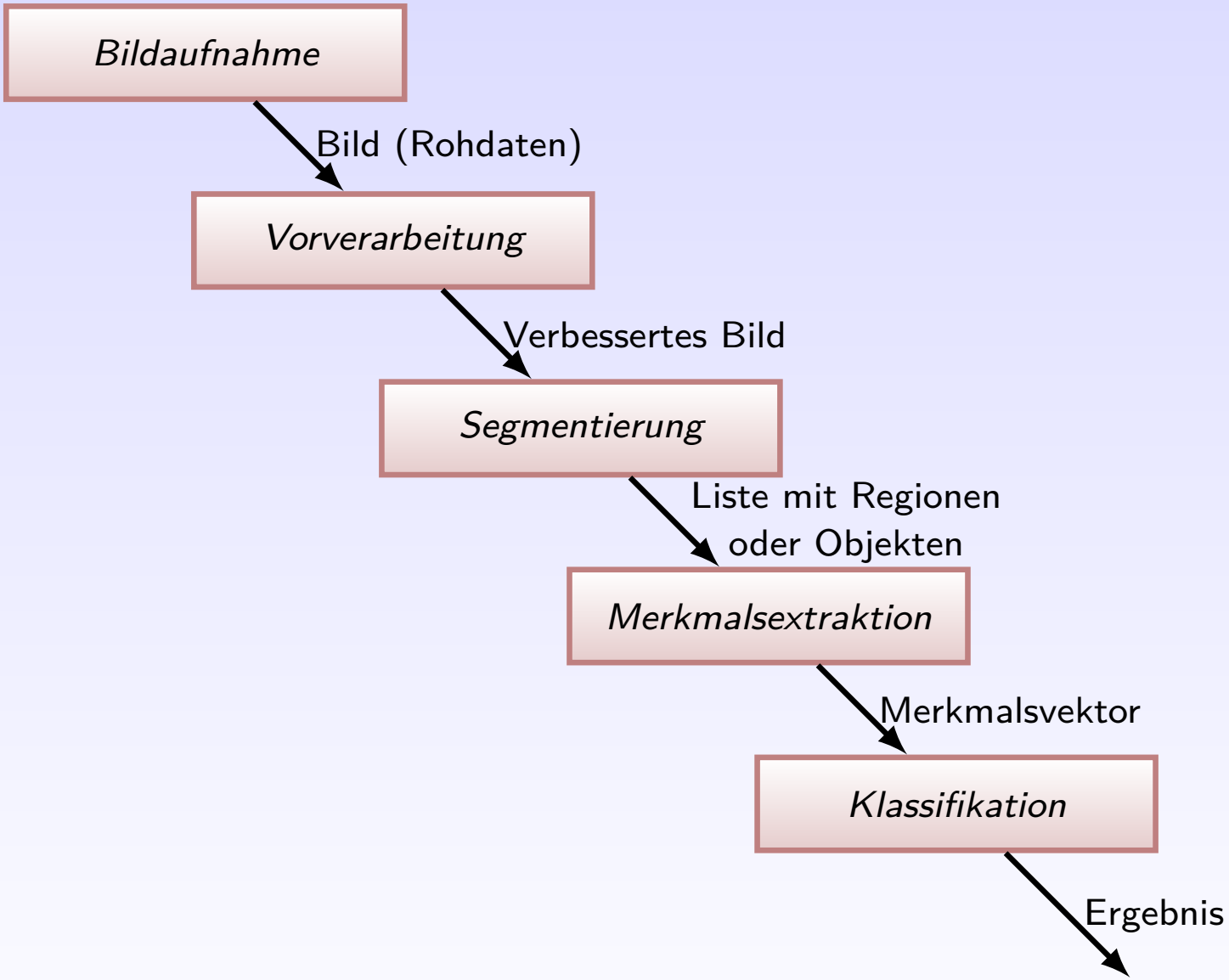
Medizin Tomographie, automatische Mikroskopie

Handel, Logistik Barcode-Auswertung

Verkehr autonome Fahrzeugführung, Maut- und Vignettensysteme

Robotik mobil: fahrerlose Transportsysteme, Lokalisation, Navigation
stationär: Analyse von Werkstücken in einer Szene
$\rightarrow$ manche Roboter serienmäßig mit BV-Systemen, z.B. Sawyer

Die klassische Bildverarbeitungskette



GIMP - GNU Image Manipulation Program

- *Manuelles* Testen von BV-Algorithmen als Vorstudie zur Automatisierung
- Bildhandling: Laden, Speichern, Anzeigen, Zoom
- Operationen: über 100 Filter
- Konvertierungen: Bildformate, Transformationen und Skalierung
- Eigene Plugins integrierbar
- Kostenfreier Download und Docu: www.gimp.org

Bildaufnahme - Komponenten

Kamera

- Spektrum: sichtbares Licht, Infrarot, Radar
- Ortsauflösung (Megapixel), Dimension (Zeile, 2D, 3D)
- Kanäle: 1 (grau), 3 (RGB) oder mehr (z.B. Multispektralbilder)
- Quantisierung: 8 Bit, 10 Bit
- Ausgabeformat: Rohdaten, komprimert (jpg)
- Qualität (Rauschen, Verwackeln, Schärfe, Verzerrung)

Beleuchtung

- zeitlich konstant (bei Bildfolgen)
- räumlich homogen
- ungestört (z.B. Infrarot, Fremdlicht verhindert)
- Technik: Auflicht/Hintergrund, Lampe/LED, ...

Szene/Hintergrund (falls wählbar)

- zeitlich konstant (bei Bildfolgen) oder bewegt (entspricht bewegter Kamera)
- monoton (einfarbig) oder natürlich (komplex)

Diskrete Darstellung von Bildern

- Zur Verarbeitung im Rechner sind Bilder diskretisiert

  Ortsbereich: $\mathbb{R}_+^2 \rightarrow \mathbb{N}_+^2$

  Koordinaten (x, y) positiv und zwischen $0 \leq x \leq W$ und $0 \leq y \leq H$

  Wertebereich: $\mathbb{R}_+^i \rightarrow \mathbb{N}_+^i$ (i=Dimension)

  Zeitachse: $t \in \mathbb{R} \rightarrow k \in \mathbb{N}$

- Konvention: 0$\hat{=}$schwarz, 255$\hat{=}$weiß/hohe Intensität
- Konvention: Nullpunkt des Koordinatensystems in linker, oberer Ecke
- Eine in räumlicher oder zeitlicher Folge geordnete Bildsequenz heißt Bildfolge
- Räumliche Bildfolge $\rightarrow$ dreidimensionale Analyse von Körpern (Computertomografie, 3D-Scanner)
- Zeitliche Bildfolge $\rightarrow$ dynamische Aussagen von Objekten (Geschwindigkeit, Beschleunigung)

Formale Beschreibung digitaler Bilder

Ortsbereich (Locations)	$\mathcal{L} = [0, W-1] \times [0, H-1] \cap \mathbb{N}^2$
	Ort $\ell = (x, y) \in \mathcal{L}$
Wertebereich (Values)	$\mathcal{V} = [0, 255]$ (grau) oder $\mathcal{V} = [0, 255]^3$ (RGB) usw.
	Wert $v = (r, g, b) \in \mathcal{V}$
Bild (Image)	$\mathcal{I} : \mathcal{L} \to \mathcal{V}$
Bildpunkt (Pixel)	$p = \big(\ell, \mathcal{I}(\ell)\big) \in \mathcal{L} \times \mathcal{V}$
	manchmal wird $\ell \in \mathcal{L}$ als Pixel bezeichnet
Kanal (Channel)	Projektion von $\mathcal{V}$ auf eine Dimension, z.B.
	$\mathcal{V} = R \times G \times B$, $R =$ Rotkanal
	$\mathcal{V} = H \times S \times V$, $H =$ Hue-kanal
Nachbarschaft	reflexive*), symmetrische Relation $\mathcal{N} \subseteq \mathcal{L} \times \mathcal{L}$
	4-Nachbarschaft: $\mathcal{N}_4(p) = \{(p, q) \in \mathcal{L}^2 : \lvert p - q \rvert \leq 1\}$
	8-Nachbarschaft: $\mathcal{N}_8(p) = \{(p, q) \in \mathcal{L}^2 : \lvert p - q \rvert < 2\}$
	Beachte: jeder Ort gehört zu seiner eigenen Nachbarschaft

*) reflexiv: Relation gilt für jedes Element zu sich selbst, Beispiel: $\leq$-Relation, $\forall x : x \leq x$

1. Einleitung

1. Einleitung
1.1 Einleitung
1.2 Implementierung und Tools
1.3 Statistische Eigenschaften von Bildern

Smart-Kameras

Kamera mit integrierter Bildverarbeitung

- \+ Einfach zu benutzen, wenig Programmierkenntnisse nötig
- \+ kompakter Aufbau, dezentrale Intelligenz
- – Herstellerspezifische Algorithmen und Oberflächen (neues System $\rightarrow$ neue Einarbeitung)
- – Teuer
- – Oft nur schwer erweiterbar

BV-Tools

- Viele Tools zur *manuellen* Bildverarbeitung verfügbar:

  Gimp, Photoshop, Irfanview, ...
- Teilweise durch Plugins erweiterbar (nicht trivial)
- Keine Möglichkeit zur Bildinterpretation

$\Rightarrow$ Zum schnellen, manuellen Ausprobieren prima, zur Automatisierung selten geeignet

Eigene Implementierung

- Anforderung: Effizienz der Algorithmen und Speicherverwaltung

  erweiterbar
- Matlab: eingeschränkte Möglichkeiten
- C#/Java: benutzerfreundlich, nicht echzeitfähig
- C/C++: weniger benutzerfreundlich, aber echtzeitfähig
- OpenCV: freie Bibliothek in C/C++, unzählige Algorithmen, sehr schnell, erweiterbar
- Effizienz vs. hübscher Klassenentwurf mit gekapseltem Datenzugriff

```
uchar get(int x, int y){if(x>=0...) return data[y*width+x]; else return 0;}
uchar get(int i) {return data[i];}
```

1. Einleitung

1. Einleitung
1.1 Einleitung
1.2 Implementierung und Tools
1.3 Statistische Eigenschaften von Bildern

Mittelwert und Kontrast

- Der mittlere Grauwert des Bildes $\mathcal{I} : [0, W-1] \times [0, H-1] \to [0, 255]$

$$\mu_{\mathcal{I}} = \frac{1}{W \cdot H} \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} \mathcal{I}(x, y) \tag{1}$$

  sagt aus, ob das Bild insgesamt dunkler oder heller ist.

- Eine Kenngröße für den Kontrast ist die mittlere quadratische Abweichung

$$\sigma_{\mathcal{I}}^2 = \frac{1}{W \cdot H} \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} \Big(\mathcal{I}(x, y) - \mu_{\mathcal{I}}\Big)^2 \tag{2}$$

- Bei mehrkanaligen Bildern $\to$ getrennte Berechnung in jedem Kanal.

`Farben` $\to$ `Helligkeit/Kontrast...`

Grauwertprofile

- Grauwerte entlang einer bestimmten Linie
- (entspricht Höhenprofil mit Steigungen entlang einer Autoroute)
- Aussage über Hintergrund und Beleuchtung
- Aussage über Kantensteilheit

Histogramme

- Histogramm = Anzahl der Bildpunkte mit identischem (Grau-)Wert
- (entspricht Notenspiegel einer Prüfung)
- Absolutes Histogramm $H_{\mathcal{I}}(g) : \mathcal{V} \to \mathbb{N}$ (Grauwert $g \to$ Anzahl)
- Relatives Histogramm $h_{\mathcal{I}}(g) = \frac{H_{\mathcal{I}}(g)}{W \cdot H}$ (Grauwert $g \to [0, 1]$)
- Berechnung von Mittelwert und mittlerer quad. Abweichung aus Histogramm

$$\mu_{\mathcal{I}} = \sum_{g=0}^{255} g \cdot h_{\mathcal{I}}(g) \qquad \sigma_{\mathcal{I}}^2 = \sum_{g=0}^{255} (g - \mu_{\mathcal{I}})^2 \cdot h_{\mathcal{I}}(g) \tag{3}$$

`Farben` → `Kurven...`

Zentralmomente

- Aus Grauwerthistogramm lassen sich zentrale Momente n-ter Ordnung ableiten

$$m_n = \sum_{g=0}^{255} (g - \mu_{\mathcal{I}})^n \cdot h_{\mathcal{I}}(g) \tag{4}$$

- Offensichtlich: $m_1 = \mu$ und $m_2 = \sigma^2$
- Die *Schiefe* m_3 gibt Grad der Asymmetrie an (Abweichung von Gaußscher Normalverteilung nach links oder rechts)
- Der *Exzess* m_4 gibt an, wie weit Verteilung von Gaußscher Normalverteilung nach oben oder unten abweicht (spitz oder abgeflacht)

- Momente und Histogramme lassen sich auch von Bild*ausschnitten* berechnen

$\rightarrow$ zur Texturanalyse, z.B. falls $m_2 = 0$ ist die Region homogen (kein Muster)

$\rightarrow$ werden bei Merkmalsextraktion für Klassifikation von Objekten ermittelt

2. Bildvorverarbeitung

2. Bildvorverarbeitung
2.1 Bildpunktoperationen
2.2 Lokale Operationen (Filter)
2.3 Lokale Operationen (Geometrische Transformationen)

Ziel und Arten der Bildvorverarbeitung

Ziel der Bildvorverarbeitung ist Bildverbesserung

- Verbesserung der Bildqualität
- Reduzierung des Aufwandes für Folgeschritte

Bildpunktoperationen

Nur einzelner Pixel des Eingangsbildes $\mathcal{I}(x, y)$ ist wichtig zur Berechnung eines Ausgangsbildpunktes $\mathcal{O}(x, y)$

Beispiel: Bildaufhellung mit konstantem Wert $\mathcal{O}(x, y) = \mathcal{I}(x, y) + k$

Reduzierung des Wertebereichs $\mathcal{O}(x, y) = \mathcal{I}(x, y) \,/\, k$

Lokale Nachbarschaftsoperationen

Lokale Nachbarschaft des Pixels $\mathcal{I}(x, y)$, z.B. 3×3-Umgebung, zur Berechnung von $\mathcal{O}(x, y)$

Beispiel: Mittelwertfilter $\mathcal{O}(x, y) =$ Mittelwert der Nachbarschaft von $\mathcal{I}(x, y)$

Globale Nachbarschaftsoperationen

Das gesamte Eingangsbild wird genutzt, um einen Ausgangsbildpunkt zu Berechnung

Beispiel: 2D-Fouriertransformation

jpg-Bildkompressionsverfahren

Bildpunktoperationen und Look-Up-Tabellen

Aus (Grau-)Wert/Position eines Eingangsbildpunktes ohne Berücksichtigung seiner Umgebung neuen (Grau-)Wert oder neue Position eines Ausgangsbildpunktes berechnen.

Transformationsfunktion kann analytisch geschlossene Form haben, z.B

Invertierung $\mathcal{O}(x,y) = 255 - \mathcal{I}(x,y)$

Lineare Kennlinie $\mathcal{O}(x,y) = k_1 \cdot \mathcal{I}(x,y) + k_0$

Quadratische Kennlinie $\mathcal{O}(x,y) = \frac{1}{255} \cdot \mathcal{I}^2(x,y)$

Binarisierung $\mathcal{O}(x,y) = \begin{cases} 0 : \mathcal{I}(x,y) < g_u \\ 1 : g_u \leq \mathcal{I}(x,y) \leq g_o \\ 0 : \mathcal{I}(x,y) > g_o \end{cases}$

`GIMP` → `Farben` → `Posterisieren (Farbanzahl vermindern)...`

Look-Up-Tabellen

Look-Up-Tabelle (LUT) = Speicher, bei dem an Stelle/Adresse x der Inhalt y gespeichert ist.

$$y = LUT(x)$$

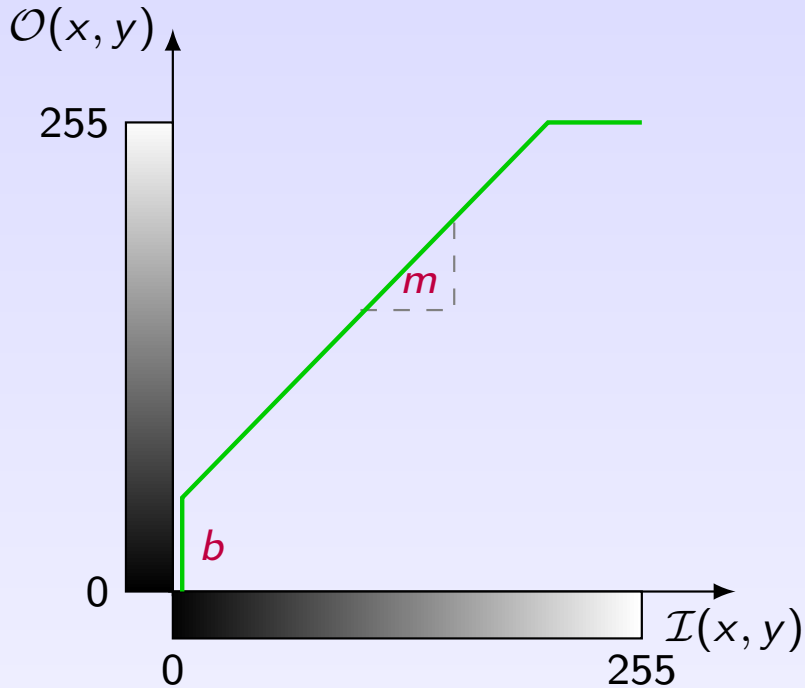
Inhalt nur einmal/initial berechnet → LUT-Operationen sind sehr schnell

Allgemeine Grauwerttransformation

- Mit Transformationsgleichung

$$\mathcal{O}(x,y) = m \cdot \mathcal{I}(x,y) + b$$

  werden Grauwerte *linear* transformiert
- $b \mathrel{\hat{=}}$ Helligkeit, $m \mathrel{\hat{=}}$ Kontrast
- Wenn $m > 1$ wird Kontrast höher
  Für $0 \leq m < 1$ wird Kontrast erniedrigt
- Addition von b macht Bild heller oder dunkler



- Im diskreten Fall muß Grauwertbereich noch begrenzt werden

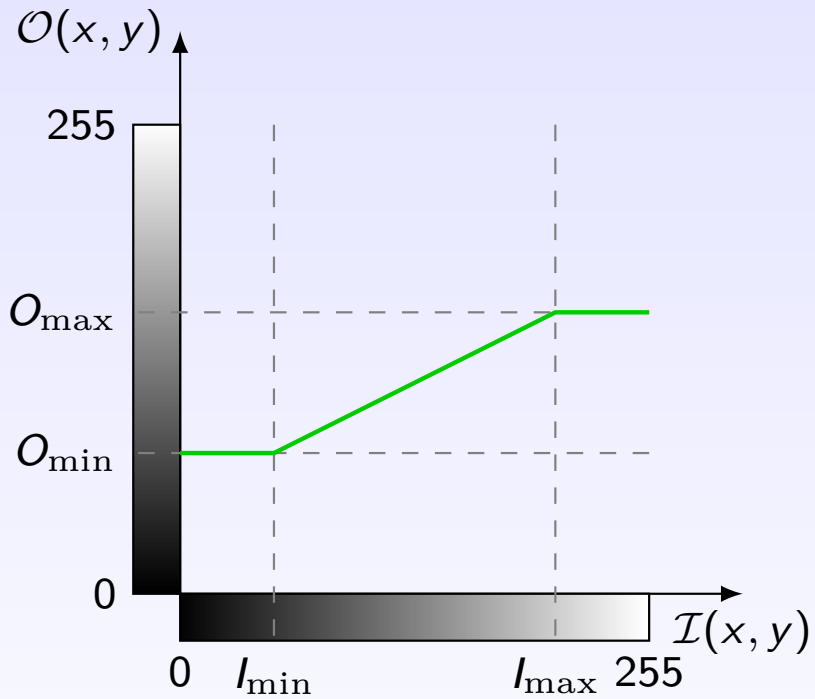
$$\mathcal{O}(x,y) = \begin{cases} 255 & : \quad m \cdot \mathcal{I}(x,y) + b > 255 \\ 0 & : \quad m \cdot \mathcal{I}(x,y) + b > 0 \\ m \cdot \mathcal{I}(x,y) + b & : \quad \text{sonst} \end{cases}$$

`GIMP` $\rightarrow$ `Farben` $\rightarrow$ `Kurven...`

Lineares Dehnen der Grauwerte

- Ein Bild $\mathcal{I}$ habe minimalen und maximalen Grauwert, $I_{\min}$ und $I_{\max}$.
- Zur besseren visuellen Betrachtung soll z.B. der Bereich [0,255] voll genutzt werden
- oder zur effizienteren Bearbeitung soll der Wertebereich reduziert werden

$\Rightarrow$ transformiertes Ausgangsbild $\mathcal{I}$ soll Grenzwerte $O_{\min}$ und $O_{\max}$ annehmen



$$\mathcal{O}(x,y) = \frac{O_{\max} - O_{\min}}{I_{\max} - I_{\min}} \cdot \left(\mathcal{I}(x,y) - I_{\min}\right) + O_{\min} \tag{5}$$

Weitere Bildpunktoperationen

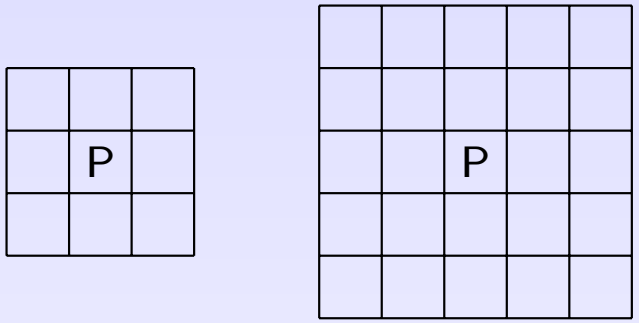
- Histogrammebnung: Transformation so, dass jeder Grauton kommt gleich oft vor
  (Bestimmung der Gewichtungsfunktion ist global)
- Farbraumtransformationen, z.B. RGB $\rightarrow$ Grau, RGB $\rightarrow$ HSV
- Räumliche Transformationen: Bildentzerrung, Verschiebung, Drehung, Zuschneiden

2. Bildvorverarbeitung

2. Bildvorverarbeitung
2.1 Bildpunktoperationen
2.2 Lokale Operationen (Filter)
2.3 Lokale Operationen (Geometrische Transformationen)

Lokale Operationen (Filter)

- Manipuliere (Grau-)Wert eines Pixels in Abhängigkeit einer geeigneten Nachbarschaft, z.B



3×3 5×5

- mehrmalige, iterative Anwendung vergrößert Nachbarschaft
- Die folgenden Filter dienen im Wesentlichen zur Unterdrückung von Rauschen (stochastische Störungen)
- Filter werden *kontextunabhängig* für jeden Pixel durchgeführt

linear basiert mathematisch auf Faltung (Konvolution, de.wikipedia.org/wiki/Faltung_(Mathematik))
Es gilt Kommutativität ($f * g = g * f$), Assoziativität ($f * (g * h) = ((f * g) * h)$ und Distributivität ($f * (g + h) = (f * g) + (f * h)$)

nichtlinear s.u.

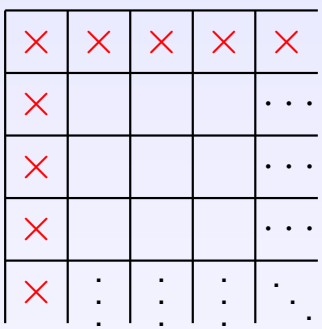
sequentiell/rekursiv Ergebnis hängt von vorherigem Teilergebnis (und Originalwerten) ab

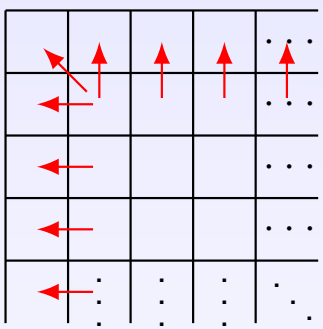
parallel/nicht-rekursiv Ergebnis nur abhängig von Originalwerten

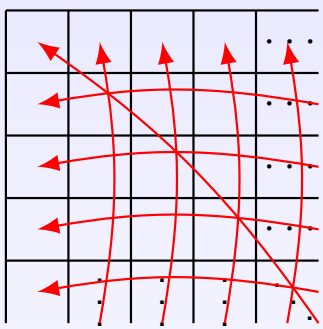
Randbehandlung

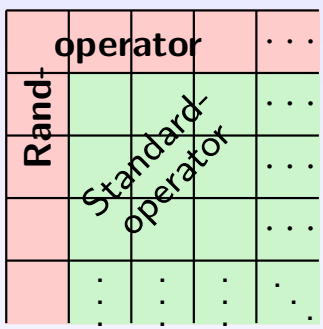
Probleme am Bildrand, weil lokale Nachbarschaft undefiniert

1. Ignorieren der Randpunkte
   z.B. bei 5×5-Filter entsteht Rand der Breite 2
2. Rand vervielfachen
   Randpunkte gehen stärker in Gewichtung ein
3. Bild periodisch fortsetzen
   nur sinnvoll bei periodischen Bildinhalten
4. speziellen „Rand-Operator“ definieren
   Hoher Programmieraufwand oder hoher Rechenaufwand (Fallunterscheidungen)









Lineare Glättung - Mittelwertfilter I

+ Vermindert Rauschen
– Kanten werden verwaschen

Boxfilter (z.B. 3×3)

$$\mathcal{O}(x,y) = \frac{1}{9} \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} \mathcal{I}(x-i, y-i) \tag{6}$$

Maskenschreibweise

$$\frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

`GIMP → Filter → Allgemein → Faltungsmatrix ...`

Lineare Glättung - Mittelwertfilter II

Gaußfilter

$$\frac{1}{6}\begin{pmatrix}0&1&0\\1&2&1\\0&1&0\end{pmatrix} \quad \frac{1}{8}\begin{pmatrix}0&1&0\\1&4&1\\0&1&0\end{pmatrix} \quad \frac{1}{10}\begin{pmatrix}1&1&1\\1&4&1\\1&1&1\end{pmatrix} \quad \frac{1}{16}\begin{pmatrix}1&2&1\\2&4&2\\1&2&1\end{pmatrix} \quad \frac{1}{256}\begin{pmatrix}1&4&6&4&1\\4&16&24&16&4\\6&24&36&24&6\\4&16&24&16&4\\1&4&6&4&1\end{pmatrix}$$

`GIMP` $\rightarrow$ `Filter` $\rightarrow$ `Weichzeichnen` $\rightarrow$ `Gaußscher Weichzeichner ...`

Triviale/Naive Implementierung eines 3×3-Mittelwertfilters

```
1  uchar* mean3x3(uchar* input, int w, int h)
2  {
3      uchar* output = new uchar[w*h];
4      for(int x=0;x<w;x++) for(int y=0;y<h;y++){          // each pixel
5          int count = 0;
6          int sum = 0;
7          for(int i=-1;i<=1;i++) for(int j=-1;j<=1;j++){// filter mask
8              if(x+i>0 && x+i<w && y+j>0 && y+j<h){      // valid pos?
9                  count++;
10                 sum += input[(y+j)*w+(x+i)];
11             }
12         }
13         output[y*w+x] = (uchar)(sum/count);
14     }
15     return output;
16 }
```

+ einfach programmiert, als Prototyp oder Vergleichsalgorithmus
– 9·w·h if-Abfragen
– w·h Berechnungen des Divisors (i.d.R. count=9)
– ≈10·w·h Aufwändige Berechnungen von Position im Datenarray: (y+j)*w+(x+i)

Effizientere Implementierung eines 3×3-Mittelwertfilters

```
1  uchar* mean3x3(uchar* input, int w, int h)
2  {
3      uchar* output = new uchar[w*h];
4      calculateBorder(input, output, w, h);
5      for(int y=1;y<h-1;y++){
6          int idx=y*w+1;
7          for(int x=1;x<w-1;x++){
8              int sum = 0;
9              for(int j=-w;j<=w;j+=w) for(int i=-1;i<=1;i++){
10                 sum += input[idx+i+j];
11             }
12             output[idx++] = (uchar)(sum/9);
13         }
14     }
15     return output;
16 }
```

+ effizienter (kein `if`, kein `count`, `output[y*w+x]` → `output[idx++]`)
– separate Randbehandlung notwendig, falls Rand egal: `memset(output,0,w*h);`
– 4 for-Schleifen → 9·w·h Zugriffe auf Inputdaten

Noch effizientere Implementierung eines 3×3-Mittelwertfilters

```
1  uchar* mean3x3(uchar* input, int w, int h)
2  {
3      uchar* tmp = new uchar[w*h];
4      uchar* output = new uchar[w*h];
5      memset(tmp,0,w*h);
6      memset(output,0,w*h);
7      for(int y=1;y<h-1;y++){
8          int i=y*w+1;
9          for(int x=1;x<w-1;x++){
10             tmp[i++] = (uchar)(input[i-1]+input[i]+input[i+1]/3);
11         }
12     }
13     for(int y=1;y<h-1;y++){
14         int i=y*w+1;
15         for(int x=1;x<w-1;x++){
16             output[i++] = (uchar)(tmp[i-w]+tmp[i]+tmp[i+w]/3);
17         }
18     }
19     delete tmp;
20     return output;
21 }
```

+ Lineare Filter sind separierbar → 6·w·h Zugriffe auf Inputdaten

Noch effizientere Implementierung eines 3×3-Mittelwertfilters

Version A

```
1    for(int y=1;y<h-1;y++){
2        int i=y*w+1;
3        for(int x=1;x<w-1;x++){
4            tmp[i++] = (uchar)(input[i-1]+input[i]+input[i+1]/3);
5        }
6    }
```

Version B

```
1    for(int y=2,i=w+1;y<h;y++,i+=w+2) for(int x=2;x<w;x++){
2        tmp[i++] = (uchar)(input[i-1]+input[i]+input[i+1]/3);
3    }
```

+ Effizientere Laufvariablenberechnungen (`i=y*w+1` → `i+=w+2`)
+ Effizientere Laufvariablenvergleiche (`y<h-1` → `y<h`)
– Code kaum verständlich

Nichtlineare Filter

- Lineare Glättungsfilter führen zur Verwaschung und Abschwächung der Kanten
  da keine Adaption an lokale Strukturänderungen
- Mittelung erfolgt unabhängig davon, ob durch Nachbarschaft eine Kante verläuft und daher einzelne Pixel zu unterschiedlichen Regionen gehören

$\rightarrow$ Abhilfe: Nichtlineare Filter

Grundidee: Ignoriere Ausreißer innerhalb Nachbarschaft
ggf. erfolgt Mittelung mit reduzierter Menge der Pixel

Bekanntester Operator ist Medianfilter aus Gruppe der Rangordnungsfilter

Rangordnungsfilter

- Sortiere Grauwerte innerhalb der Maske nach Größe

  Die sortierte Folge der N Grauwerte ist $g_1, \ldots, g_N$
- Rangordnungsfilter bildet eine Linearkombination der sortierten Grauwerte

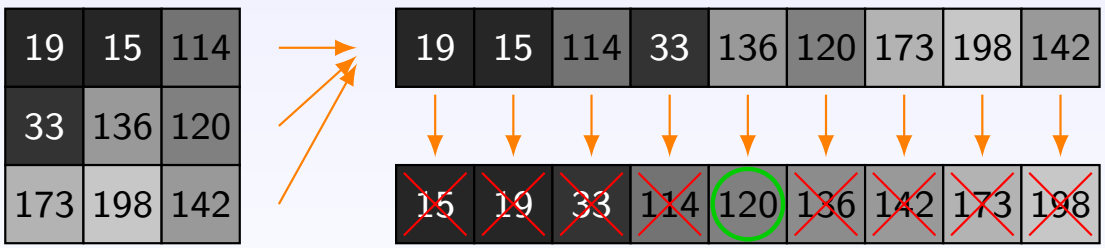
$$g_{\text{neu}} = \sum_{i=1}^{N} w_i g_i \tag{7}$$

Medianfilter

Für $n \times n$-Nachbarschaft mit ungeradem n ist $N = n^2$ ungerade und Median ist definiert als

$$g_{\text{neu}} = g_{(N+1)/2} \tag{8}$$

$$w_i = \begin{cases} 1 : i = (N+1)/2 \\ 0 : \text{sonst} \end{cases}$$

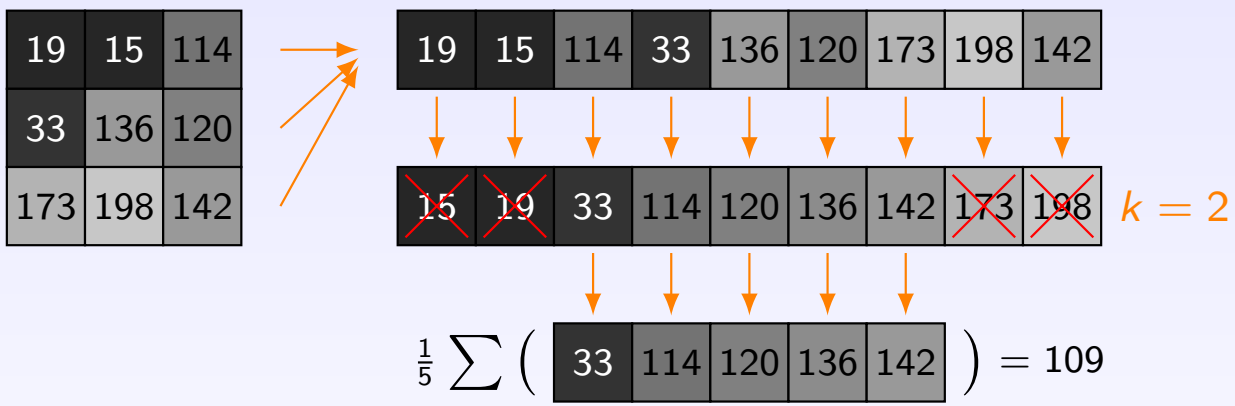


`GIMP → Filter → Verbessern → Flecken entfernen...`

k-trimmed-mean Filter

$$g_{\text{neu}} = \frac{1}{N-2k} \sum_{i=k+1}^{N-k} g_i \tag{9}$$

- $2k$ Werte (Ausreißer) werden entfernt,
  dann folgt gleichgewichtete Mittelwertbildung



Farb-Medianfilter

- Für Farben (Vektoren) gibt es keine lineare Ordnung ($\leq$)
- Unabhängiges Medianfiltern in einzelnen Farbkanälen kann zu Farben führen, die nichts mit Ausgangsfarben zu tun haben
- Der Euklidische Abstand zwischen Farbwerten c_1, c_2 ist

$$|c_1 - c_2| = \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2}$$

- Der Vektor-Median ist der Wert $c_m \in c_1 \ldots c_N$, bei dem

$$\sum_{i=1}^{N} |c_i - c_m| \text{ minimal ist}$$

Weitere nichtlineare Filter

k-Nearest Neighbor (knn-Filter)

- Alle Pixel der Nachbarschaft werden mit Zentrumspixel verglichen
- Auswahl von k Pixeln $x_1 \dots x_k$ mit kleinsten Abständen zum Zentrumspixel
- Gleichgewichtete Mittelung der k Pixel

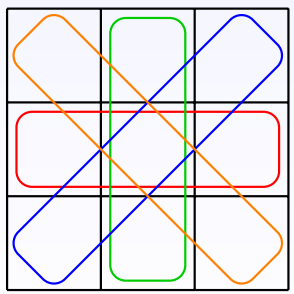
$$g_{\text{neu}} = \frac{1}{k} \sum_{i=1}^{k} g_i \tag{10}$$

- Steuerparameter k (und Maskengröße n) sind Maß für Glättung
- Für 3×3-Maske liefert $k = 6$ die besten Ergebnisse (nach Davis & Rosenfeld)

SNN - Symmetric Nearest Neighbor (Harwood et. al., 1987)

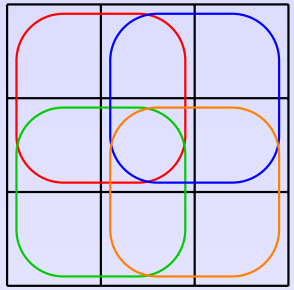
Aus vier zentrumssymmetrischen Paaren der 3×3-Maske wird jeweils zum Zentrum ähnlichstes Pixel ausgewählt → Mittelwert dieser vier Werte $g_1 \dots g_4$

$$g_{\text{neu}} = \frac{1}{4} \sum_{i=1}^{4} g_i \tag{11}$$



Filter nach Kuwahara/Nagao (1976)

- 3×3-Maske wird in vier überlappende 2×2-Bereiche unterteilt
- Wähle Mittelwert des Bereiches mit kleinster Varianz σ^2
- Effiziente Implementierung möglich, nur eine σ^2- und μ-Berechnung pro Pixel



2. Bildvorverarbeitung

2. Bildvorverarbeitung
2.1 Bildpunktoperationen
2.2 Lokale Operationen (Filter)
2.3 Lokale Operationen (Geometrische Transformationen)

Lokale Operationen (Geometrische Transformationen)

Bisher: nur Veränderung der Grau- bzw. Farbwerte, unabhängig von Position im Bild
Manchmal auch Koordinatentransformationen notwendig, z.B.

- Größenänderung
- Entzerrung (Kamerafehler, Korrektur von räumliche Projektion)
- Pattern-Matching (Vergleich von Strukturen zwischen verschiedenen Bildern, Berechnung von Bewegungsparamtern in Bildfolgen)
- Bildstabilisation in Bildfolgen

Dazu gehören

Translation Verschiebung von Bildern oder Regionen

Rotation Drehung von Bildern oder Regionen

Skalierung Vergrößern oder Verkleinern von Bildern

$\rightarrow$ Affine Transformationen

Geometrische Transformation besteht aus zwei Schritten

1. Räumliche Transformation, definiert neue Anordnung der Bildpunkte (Bildpunktoperation)
2. Grauwertinterpolation, um Zusammenhang im diskreten Gitter zu erhalten (Lokale Operation)

Räumliche Transformation in homogenen Koordinaten

Translation

$$\begin{aligned} x' &= 1 \cdot x + 0 \cdot y + d_x \\ y' &= 0 \cdot x + 1 \cdot y + d_y \end{aligned} \qquad \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{12}$$

Skalierung

$$\begin{aligned} x' &= s_x \cdot x + 0 \cdot y + d_x \\ y' &= 0 \cdot x + s_y \cdot y + d_y \end{aligned} \qquad \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & d_x \\ 0 & s_y & d_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{13}$$

Rotation (um Ursprung)

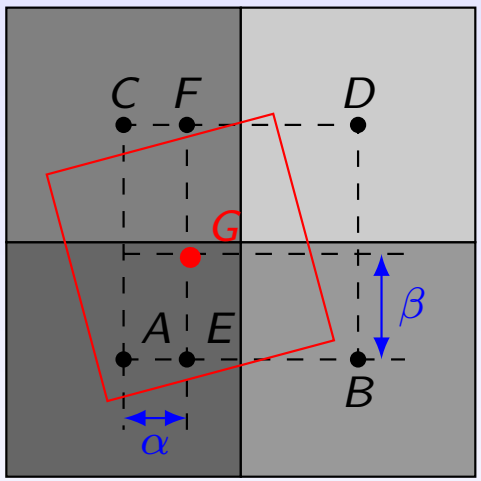
$$\begin{aligned} x' &= \cos\alpha \cdot x + \sin\alpha \cdot y + d_x \\ y' &= -\sin\alpha \cdot x + \cos\alpha \cdot y + d_y \end{aligned} \qquad \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & \sin\alpha & d_x \\ -\sin\alpha & \cos\alpha & d_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{14}$$

- Kombination mehrerer Transformationen → Multiplikation der Transformationsmatrizen
  Beispiel: Beliebige Rotationsachse $\hat{=}$ Translation in Ursprung + Rotation + Rücktranslation
- Transformationsparameter können auch positionsabhängig sein,
  Beispiel: Scherung ($\square \to \diagup\!\square$) $\hat{=}$ Translation mit variierenden Parametern

Bilineare Interpolation

- Bei ganzzahligen Translationen und Rotationen um 90° bleibt Bildraster erhalten
- Beliebige Rotationen und Skalierungen erfordern Interpolationsverfahren
- Durch Transformation entstehen keine ganzzahligen Koordinaten
- Ergebnispixel wird auf Position von vier Originalpixeln abgebildet

$\Rightarrow$ Interpolation notwendig



Seien $A \ldots D$ die Grauwerte der Originalpixel, in denen die Ecken des zu berechnenden Pixel liegen.

Die bilineare (Gesamt-)Interpolation erhält man aus linearen Interpolationen zwischen je zwei Punkten:

$$\begin{aligned} \overline{AB}: &\quad E = A + \alpha(B - A) = (1-\alpha)A + \alpha B \\ \overline{CD}: &\quad F = C + \alpha(D - C) = (1-\alpha)C + \alpha D \\ \overline{EF}: &\quad G = E + \beta(F - E) = (1-\beta)E + \beta F \end{aligned}$$

$$G = (1-\beta)(1-\alpha) \cdot A + (1-\beta)\alpha \cdot B + \beta(1-\alpha) \cdot C + \beta\alpha \cdot D \tag{15}$$

Effizientere Alternative: *„Nächster Nachbar“ statt Interpolation*

3. Farbe

3. Farbe
3.1 Farbwahrnehmung
3.2 Farbräume

Farbwahrnehmung

- Der physikalische Reiz des Sehens ist Licht
- Sichtbares Licht ist Teil des elektromagnetischen Energiespektrums ($\approx$ 380 nm bis 780 nm)

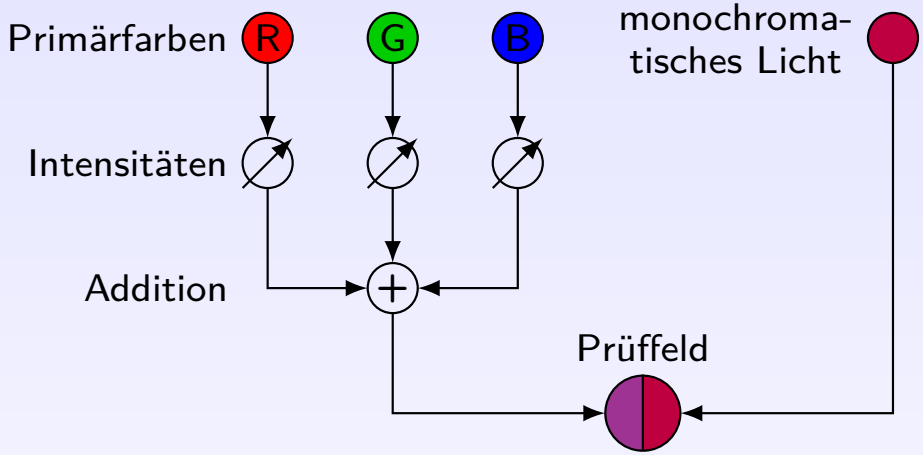
- Die meisten Körper sind nicht selbstleuchtend
  $\rightarrow$ nur sichtbar, weil Licht von Lichtquelle remittiert wird
- Oberflächen sind selektive Reflektoren
  $\rightarrow$ Remittierung in bestimmten Wellenlängen bevorzugt $\rightarrow$ Farbeindruck

- Rezeptoren im Auge: Stäbchen und Zapfen
- Zapfen: farbempfindlich, drei Arten mit unterschiedlicher spektraler Empfindlichkeit
- Stäbchen: Sensor für Helligkeit, 10.000$\times$ empfindlicher als Rezeptoren

Farbmetrik

= Lehre von Maßbeziehungen zwischen Farben

- Additive Farbmischung: Zusammenwirken mehrerer unterschiedlicher Farbreize $\rightarrow$ Mischfarbe
- Mischung durch gleichzeitige Überlagerung (mehrere Lichtquellen), räumliches Nebeneinander (kleine Pixel nah bei einander) oder schnelles Nacheinander (Bildfolge)
- Experiment der additiven Farbmischung zur Entscheidung über Gleichheit von Farben
- Wählt man als Primärfarben RGB, lässt sich fast jede Farbe mischen
- Metamere: Auch bei gleicher Farbwahrnehmung können sich spektrale Zusammensetzung der beiden Farben unterscheiden

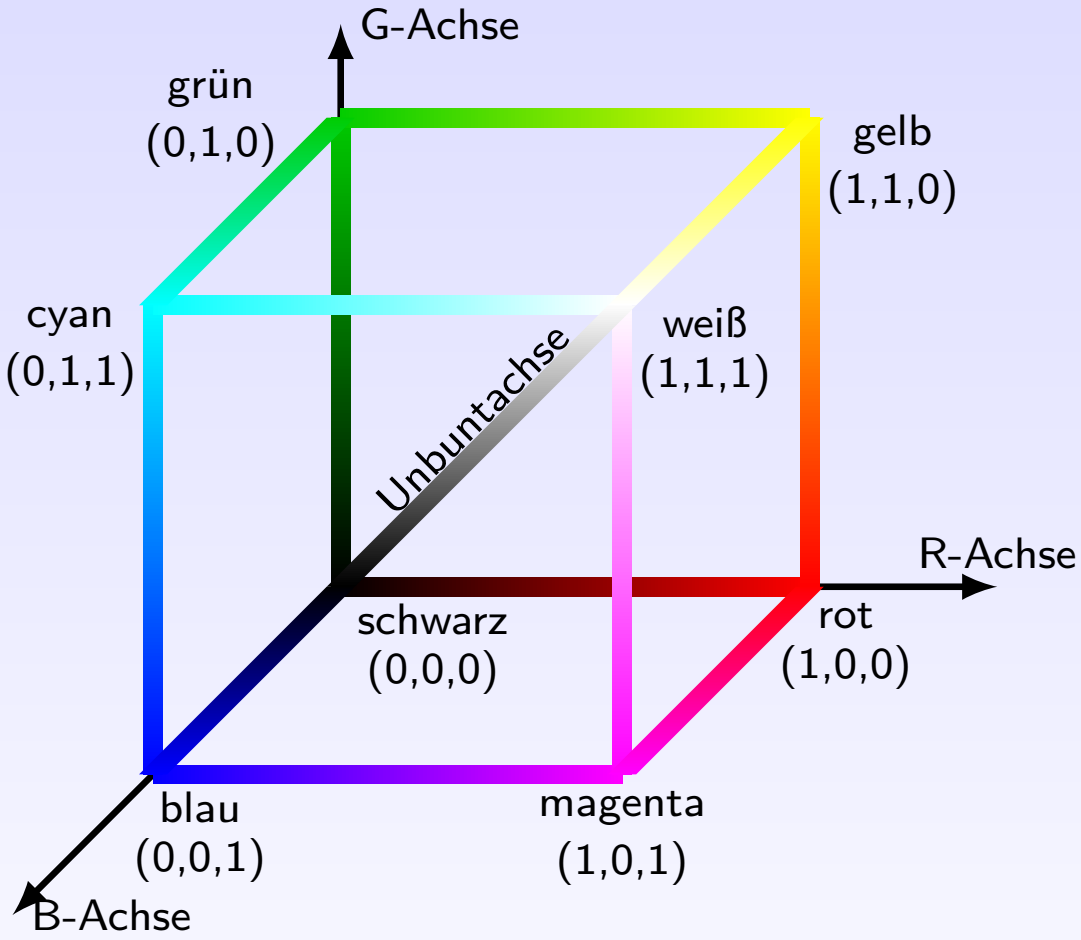


- Wahl der voneinander unabhängigen Primärfarben prinzipiell beliebig

3. Farbe

3. Farbe
3.1 Farbwahrnehmung
3.2 Farbräume

RGB-Farbraum



`GIMP → Ansicht → Rahmenfarbe → Benutzerdefinierte Farbe wählen ...`

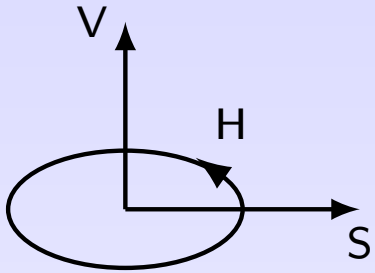
HSV-Farbraum

Beschreibung von Farbe durch Attribute

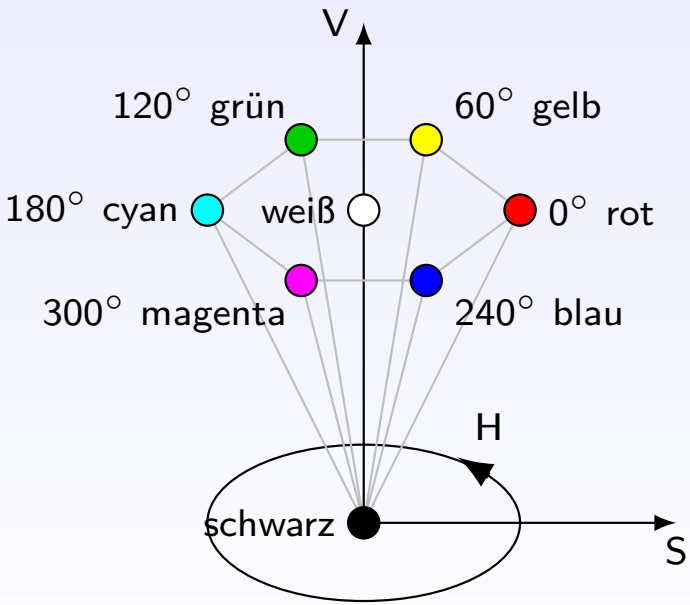
Hue (Farbton) Farbwinkel, rot bei 0°

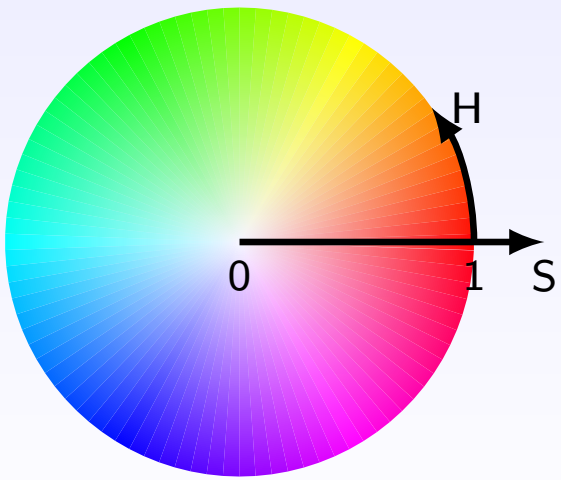
Saturation (Sättigung) Entfernung von Unbuntachse
Grauwerte bei S=0

Value (Helligkeit) schwarz bei V=0, hell bei V=1



- Singularität bei S=0: Farbwinkel undefiniert
- Anwendung: Trennung von Bildern in Regionen (Objekte) unterschiedlicher Farbe unahbhängig von Helligkeit
- Verwandte Farbräume: HSL, HSI





RGB → HSV

```
// r,g,b in [0,1]; h in [0,360]; s und v in [0,1]

max := Maximum(r,g,b);
min := Minimum(r,g,b);
v := max;
if max>0 then s:= (max-min)/max;
else s:=0;
if s=0 then h:= UNDEF;
else begin
   delta := max-min;
   if r=max then h :=     (g-b)/delta;
   if g=max then h := 2 + (b-r)/delta;
   if b=max then h := 4 + (r-g)/delta;
   h := h*60;
   if h<0 then h := h+360;
end
```

HSV → RGB

```
// r,g,b in [0,1]; h in [0,360] oder UNDEF; s und v in [0,1]

if s=0 then
  (r,g,b) := (v,v,v);
else begin
  if h=360 then h := 0;
  h := h/60;                    // h in [0,6[
  i := trunc(h);                // ganzer Anteil von h, h in {0,...,5}
  f := h-i;                     // Rest von h, f in [0,1[
  p := v * (1-s);
  q := v * (1-s*f);
  t := v * (1-s*(1-f));
  case i of
    0: (r,g,b) := (v,t,p);
    1: (r,g,b) := (q,v,p);
    2: (r,g,b) := (p,v,t);
    3: (r,g,b) := (p,q,v);
    4: (r,g,b) := (t,p,v);
    5: (r,g,b) := (v,p,q);
  end
end
```

YIQ-Farbraum

- Eingesetzt in amerikanischer Fernsehübertragung (seit 1953)
- Codierung von RGB für bessere Übertragungseffizienz und Kompatibilität mit Schwarz-Weiß-Fernsehen
- Y-Komponente = Helligkeit (entspricht CIE Normfarbsystem)
- Farbinformation in I und Q codiert

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0,299 & 0,587 & 0,114 \\ 0,500 & -0,230 & -0,270 \\ 0,202 & -0,500 & 0,298 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{16}$$

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & 1,139 & 0,650 \\ 1 & -0,325 & -0,677 \\ 1 & -1,317 & 1,780 \end{pmatrix} \cdot \begin{pmatrix} Y \\ I \\ Q \end{pmatrix} \tag{17}$$

- Das europäische PAL-System überträgt statt I und Q die Farbdifferenzen R-Y und B-Y

RGB → Grau

Farbbildverarbeitung ist **nicht** 3× Grauwertverarbeitung in 3 Kanälen mit anschließender Fusion der Zwischenergebnisse (18)

Verschiedene Methoden zur Konvertierung eines RBG-Bildes in 1-Kanaliges Bild (Grau)

- Arithmetischer Mittelwert

$$x = \frac{1}{3}(r + g + b) \tag{19}$$

- Maximum (V-Kanal von HSV)

$$x = \max(r, g, b) \tag{20}$$

- Wahrnehmungsphysiologische Konvertierung (Y-Komponente des CIE-Normfarbsystems)

$$x = 0,299 \cdot r + 0,587 \cdot g + 0,114 \cdot b \tag{21}$$

- Quick and Dirty

$$x = g \tag{22}$$

Literatur, Vertiefung - Farbe

- WIKIPEDIA de.wikipedia.org/wiki/Zapfen_(Auge)
- WIKIPEDIA de.wikipedia.org/wiki/Farbe
- WIKIPEDIA en.wikipedia.org/wiki/HSL_and_HSV

4. Segmentierung

4. Segmentierung
4.1 Allgemeines
4.2 Schwellwertverfahren
4.3 Regionenorientierte Verfahren

Ziel, Begriffe

Ziel: Zerlegung des Bildes in visuell unterschiedliche, homogene Regionen.
Homogenität bezieht sich z.B. auf Grauwert, Farbe, Textur, o.ä.

Begriffe

Region räumlich zusammenhängende Menge von Pixeln

Zwischen je zwei Pixeln existiert ein Pfad, der Region nicht verlässt

Segment Region aus paarweise ähnlichen Pixeln

Segmentierung Zerlegung eines Bildes in möglichst große, disjunkte Segmente

Objekt (im Bild) Menge von Segmenten, die zusammen ein reales Objekt darstellen, z.B. Auto

Ergebnisse können sein:

Binärbild Partitionierung in Vorder-/Hintergrund bzw. interessierend/nicht-interessierend Falls

Vordergrund nicht zusammenhängend, kann Etikettierung erfolgen

Labelbild Pixel erhalten eindeutige ID ihrer jeweiligen Regionen

Regionenbild Pixel erhalten mittleren Farb-/Grauwert ihrer jeweiligen Regionen

Unterscheidung in

- Kantenbasierte Verfahren
- Punktorientierte Verfahren
- Regionenorientierte Verfahren

4. Segmentierung

4. Segmentierung
4.1 Allgemeines
4.2 Schwellwertverfahren
4.3 Regionenorientierte Verfahren

Schwellwertverfahren

- Wichtige Annahme: Helle Objekte auf dunklem Hintergrund oder umgekehrt

$$\mathcal{O}(x,y) = \begin{cases} 1 & \mathcal{I}(x,y) \geq G \\ 0 & \text{sonst} \end{cases} \tag{23}$$

- Problem: finde geeignete Schwelle/Grenze $G \in [0, 255]$
- Berechnung aus Histogramm oder Ausprobieren verschiedener Werte mit apriori-Wissen
- Globale Schwelle für gesamtes Bild oder variable Schwelle $G(x, y)$

Eigenschaften:

+ einfache Implementierung, schnelles Verfahren
+ vollständige, überdeckungsfreie Segmentierung
+ gute Ergebnisse bei gleichmäßiger Beleuchtung und einfachen Szenen
– Anfällig für Helligkeitsänderungen
– Nur eindimensionaler (Grau-)Wert als Schwelle (Verarbeitung von Farbe?)
– Räumlicher Zusammenhang der Bildbereiche nicht gewährleistet
– Stark abhängig vom Parameter Schwellwert

Schwellwertverfahren nach Otsu, 1979

$\Rightarrow$ Aufteilung des Histogramms in (zwei) Klassen, so dass Varianz innerhalb Klassen minimal (homogene Klassen) und zwischen Klassen maximal wird

- Schwellwert G teilt Histogramm/Grauwertbereich [0,255] in Klassen

$$K_0 = \{g \,|\, 0 \leq g \leq G\} \text{ und } K_1 = \{g \,|\, G < g \leq 255\}$$

- Wahrscheinlichkeit des Auftretens von Pixeln in den Klassen sind

$$p_0 = \sum_{g=0}^{G} h(g) \text{ und } p_1 = \sum_{g=G+1}^{255} h(g)$$

- Mittelwerte des Gesamtbildes und der beiden Klassen sind $\overline{g}$, $\overline{g_0}$ und $\overline{g_1}$
- Varianzen der Klassen sind

$$\sigma_0^2 = \sum_{g=0}^{G} (g - \overline{g_0})^2 h(g) \text{ und } \sigma_1^2 = \sum_{g=G+1}^{255} (g - \overline{g_1})^2 h(g)$$

$$Q(G) = \frac{\sigma^2_{\text{zwischen}}}{\sigma^2_{\text{innerhalb}}} = \frac{p_0(\overline{g} - \overline{g_0})^2 + p_1(\overline{g} - \overline{g_1})^2}{p_0\sigma_0^2 + p_1\sigma_1^2} \stackrel{!}{=} \min \tag{24}$$

Verfahren nach Kittler & Illingworth

$\Rightarrow$ Annäherung des (tatsächlichen) Histogramms $h(g)$ durch zwei (fiktive) Gaußverteilungen

$$h(g) \approx \tilde{h}(g) = a_0 \mathrm{e}^{-(g-\overline{g_0})^2/\sigma_0^2} + a_1 \mathrm{e}^{-(g-\overline{g_1})^2/\sigma_1^2} \tag{25}$$

mit Randbedingung/Ziel $\sum_g |h(g) - \tilde{h}(g)| \stackrel{!}{=} \min$

- Schwellwert G ist der Schnittpunkt der Gaußkurven

Fazit, Modifikationen

→ obige Verfahren sinngemäß erweiterbar auf mehrere Schwellwerte

- i.d.R. ist eine kantenerhaltende Vorfilterung empfehlenswert

– Räumliche Verteilung unberücksichtigt

Variable Schwellen

- Globale Schwelle für gesamtes Bild → oft schlechte Segmentierung
- Idee: Unterteilung des Bildes in Bildausschnitte
  Ermittlung eines Schwellwertes innerhal Ausschnitt
- Extremfall: individuelle Schwellwertermittlung für jeden Pixel, abhängig von Umgebung

Rekursives Histogrammsplitting

0. Initial ist das gesamte Bild eine Region

   Speichere Region in einem Stack (push)
1. Nimm eine Region vom Stack (pop)
2. Berechne Histogramm von Region
3. Falls Histogramm multimodal, ...

   bestimme Schwellwert und partitioniere Region in neue Subregionen
4. Speichere Subregionen im Stack (push)
5. Wiederhole 1-4, bis Stack leer

4. Segmentierung

4. Segmentierung
4.1 Allgemeines
4.2 Schwellwertverfahren
4.3 Regionenorientierte Verfahren

Regionenorientierte Verfahren

= flächenorientierte Verfahren

- betrachten Punktmengen als Gesamtheit
- erzeugen prinzipiell zusammenhängende Objekte

- Region Growing (Bottom-Up)
- Split und Merge (Top-Down)
- Wasserscheidentransformation
- Pyramid Linking
- CSC nach Priese/Rehrmann

Region Growing

= Bereichtswachstumsverfahren (Bottom-Up)

- Ausgehend von bestimmten Positionen wachsen Region allmählich
- Binäres Ähnlichkeitsmaß (ähnlich/nicht ähnlich) basierend auf Farbe, Grauwert, Textur, o.ä. z.B. Differenz der Grauwerte kleiner als feste Schwelle

Einfache Verkettung

- Jedes Pixel entspricht Knoten eines Graphen
- benachbarte Pixel, die ähnlich sind, werden durch Kante verbunden
- Bildsegmente = maximales Zusammenhangskomponenten des Graphen
- Ähnlichkeitsmaß nur lokal ausgewertet

\+ einfach, sehr schnell

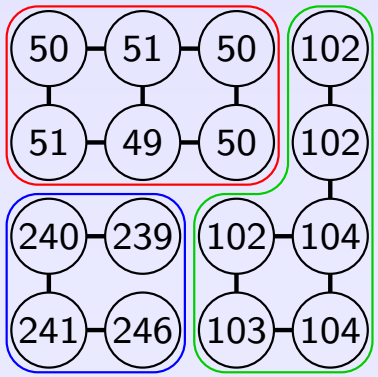
– Gefahr von Verkettungsfehlern bei sanften (Farb-)Verläufen

– eine einzige Kante genügt (z.B. ein Pixelfehler), um zwei Regionen zu verschmelzen

50	51	50	102
51	49	50	102
240	239	102	104
241	246	103	104

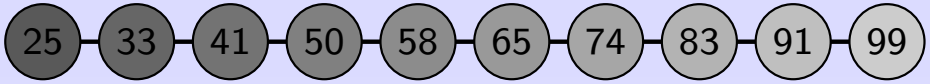
Bild

Ähnlichkeit
$\Delta g \leq 5$



Graph

Region Growing



Verkettungsfehler bei $\Delta g \leq 10$

Hybride Verkettung

- Ähnlichkeitsmaß basiert nicht auf einzelnem Pixel, sondern auf Nachbarschaft

→ Verschmelze zwei Pixel, wenn sie ähnliche Nachbarschaften haben

- Grundproblem der Verkettungsfehler bleibt, ist aber reduziert
- Aufwand steigt

Zentroide Verkettung

- Ähnlichkeitsmaß basiert Mittelwert der (noch nicht fertigen) Regionen

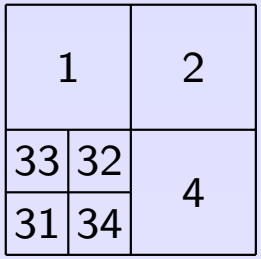
→ Verschmelze Pixel mit Region, wenn Pixel und Mittelwert der Region ähnlich sind

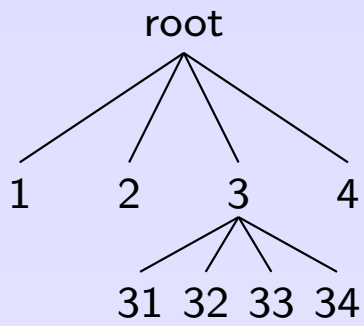
- Grundproblem der Verkettungsfehler bleibt, ist aber reduziert
- Aufwand steigt
- Ergebnis abhängig von Bearbeitungsreihenfolge

1. `Unterteile Bild in initiale, kleine Zellen, z.B.` 1×1, 2×2, `...`
2. `Berechne (statistisches) Maß für alle Zellen, z.B. Mittelwert`
3. `Wähle eine Zelle und vergleiche Statistik mit benachbarten Zellen ...`
   `falls ähnlich, verschmelze Zellen und aktualisiere Statistik`
4. `Segment wächst, bis keine weiteren Nachbarzellen mehr ähnlich sind`
5. `Nimm nächste unbearbeitete Zelle und wiederhole Vorgang`

Split and Merge, Horowitz/Pavlidis 1976

- Top-down-Verfahren
- Betrachte initial gesamtes Bild oder große Bildbereiche als Region
  falls nicht homogen, teile iterativ in Subregionen
- Algorithmus arbeitet effizient auf Quadtree Datenstruktur





`Initial` `Teile Bild in` n^2 `gleich große, quadratische Regionen auf`

`Split` `Rekursiv: teile jede nicht-homogene Region in vier (Sub-)Regionen auf`

`Merge` `Falls Vereinigung von 4 Regionen (mit gemeinsamem Parent im Quadtree)`
`homogen, verschmelze diese`

`Group` `Falls Vereinigung von 2 beliebig großen, benachbarten Regionen homogen,`
`verschmelze diese`

Homogenitätskriterium z.B. Differenz zwischen Maximal- und Minimalwert

Wasserscheidentransformation

- Interpretation der Grauwerte als topologisches Gebirge
- Wassertropfen streben entlang des größten Gefälles zu lokalen Minimum $\hat{=}$ Staubecken
- Einflusszonen der Staubecken $\hat{=}$ Region
- Wasserscheiden = Trennlinien zwischen Staubecken = Segmentgrenzen

\+ komplette Kantenzüge - kein Kantenverfolgungsalgorithmus nötig

\+ Segmentierung immer vollständig, zusammenhängend und überdeckungsfrei

– oft Übersegmentierung: Nachbearbeitung nötig

– Anfälligkeit für Rauschen

– Wasserscheide bei Höhen-Plateau nicht eindeutig

Algorithmus `Simulation von Überflutung (stetig steigender Wasserpegel)`
`Gleichzeit startet in jedem lokalen Minimum eine Überflutung`
`Wasserscheiden an Stellen, wo sich Wasser verschiedener Minima trifft`

$\rightarrow$ Final ist jedes Minimum von Dämmen (Segmentgrenzen) umgeben

! Anwendung auf Gradientenbild

`cmm.ensmp.fr/~beucher/wtshed.html`

5. Morphologie

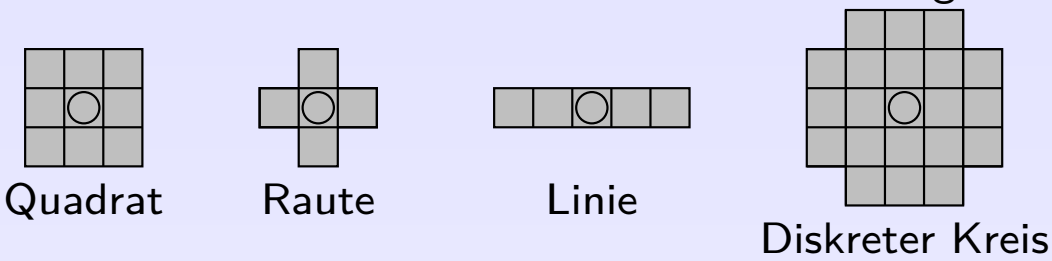
5. Morphologie
5.1 Basisoperationen
5.2 Komplexere Morphologieoperationen

Einführung

- Ziel: Extraktion relevanter Bildstrukturen
- Hier: nur Binärbild-Morphologie
- Bilder werden als Mengen betrachtet (Menge von Koordinaten, an denen Pixel=true ist)
- Anhand von A-priori-Wissen wird Strukturierendes Element SE definiert
- Basisoperationen: Erosion: passt SE vollständig in Menge der true-Pixel?
  Dilatation: berührt SE die Menge der true-Pixel?

Strukturierendes Element

- (kleine) Menge an Koordinaten, die Form bzw. Nachbarschaft beschreibt
- enthält Bezugspunkt/Ursprung ○
- Form und Größe an jeweilige Aufgabe angepasst,
  z.B. linienförmige SE zum Extrahieren oder Unterdrücken linienförmiger Strukturen



- Oft parallele Anwendung verschiedener Rotationen des selben SE, anschl. Kombination der Ergebnisse
- Oft iterative Anwendung verschiedener einfacher SE um komplexe SE nachzubilden

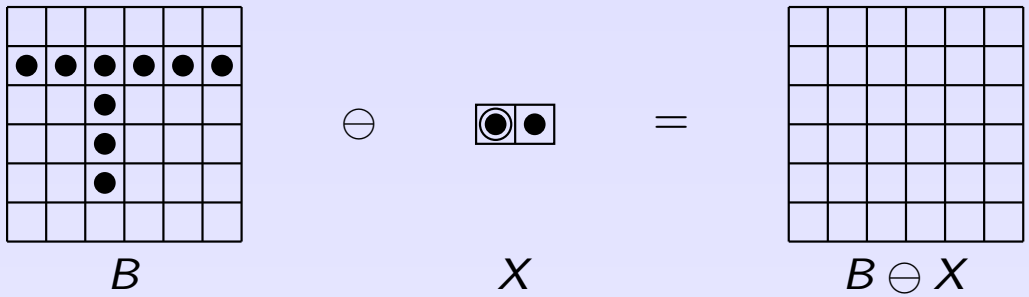
Erosion

= Minkowski-Subtraktion

- Passt das SE vollständig in die Menge?
- Erosion ist die Menge E der Pixel b eines Bildes B, so dass das SE X vollständig in B enthalten ist, wenn sich Bezugspunkt an b befindet

$$D = B \ominus X = \{b | X_b \subseteq B\} \tag{26}$$

X_b Translation von X um b



- Erosion trägt Rand des Objektes im Bild entsprechend der Größe/Form des SE ab

⇒ große Objekte werden kleiner
kleine Objekte verschwinden ganz

`GIMP → Filter → Allgemein → Erodieren`

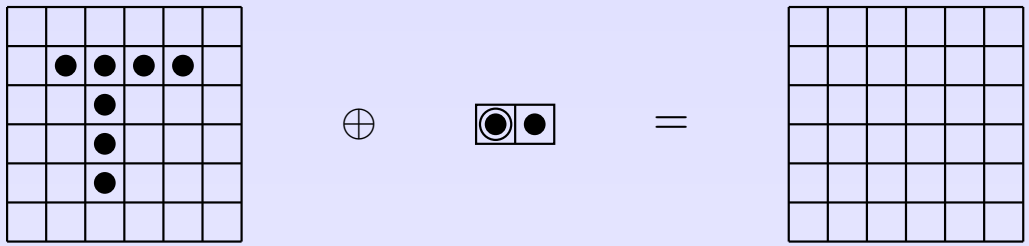
Dilatation

= Minkowski-Addition

- Berührt das SE die Menge?
- Dilatation ist die Menge D der Pixel b, so dass das SE X die Menge B berührt (schneidet), wenn sich Bezugspunkt an b befindet

$$E = B \oplus X = \{b | X_b \cap B \neq \emptyset\} \tag{27}$$

X_b Translation von X um b



- Dilatation erweitert Rand des Objektes im Bild entsprechend der Größe/Form des SE

⇒ Objekte werden größer

`GIMP → Filter → Allgemein → Erweitern`

Eigenschaften/Rechenregeln von Dilatation und Erosion I

- Erosion ist monoton wachsend

$$B_1 \subseteq B_2 \Rightarrow (B_1 \ominus X) \subseteq (B_2 \ominus X) \tag{28}$$

Falls Objekt B_1 in anderem Objekt B_2 enthalten ist, gilt dies auch für die erodierten Objekte

- Dilatation ist monoton wachsend

$$B_1 \subseteq B_2 \Rightarrow (B_1 \oplus X) \subseteq (B_2 \oplus X) \tag{29}$$

Falls Objekt B_1 in anderem Objekt B_2 enthalten ist, gilt dies auch für die dilatierten Objekte

- Dilatation ist kommutativ

$$B \oplus X = X \oplus B \tag{30}$$

Analog zur Addition von Zahlen

- Erosion ist nicht kommutativ

$$B \ominus X \neq X \ominus B \tag{31}$$

Analog zur Subtraktion von Zahlen

Eigenschaften/Rechenregeln von Dilatation und Erosion II

- Dilatation ist extensiv

$$B \subseteq B \oplus X, \text{ falls } X \text{ den Ursprung enthält} \tag{32}$$

In diesem Fall bleibt jeder Punkt eines Objekts bei Dilatation erhalten; Objekt wird an keiner Stelle kleiner

- Erosion ist anti-extensiv

$$B \ominus X \subseteq B, \text{ falls } X \text{ den Ursprung enthält} \tag{33}$$

In diesem Fall ist das erodierte Objekt Teilmenge des Originalobjektes; Objekt wird an keiner Stelle größer

- Translationsinvarianz

$$B \ominus X_k = (B \ominus X)_k \tag{34}$$

Verschiebung des Zentrums der Maske entspricht Verschiebung des Ergebnisbildes

- Dilatation und Erosion sind komplementäre Operationen

$$\overline{B} \oplus X = \overline{B \ominus X} \tag{35}$$

Dilatation des Komplementbildes ist gleich Komplement des erodierten Bildes

Eigenschaften/Rechenregeln von Dilatation und Erosion III

- Falls B konvex, ist auch $B \ominus X$ konvex

  Konvex: Jede Verbindungslinie zwischen beliebigen Pixeln verläuft vollständig innerhalb Objekt

- Dilatation und Erosion lassen sich separieren

$$B \oplus (X_1 \cup X_2) = (B \oplus X_1) \cup (B \oplus X_2)$$
$$B \ominus (X_1 \cup X_2) = (B \ominus X_1) \cap (B \ominus X_2) \tag{36}$$

Anwendung einer großen Maske lässt sich durch logische Verknüpfung der Ergebnisse mit kleineren Teilmasken realisieren

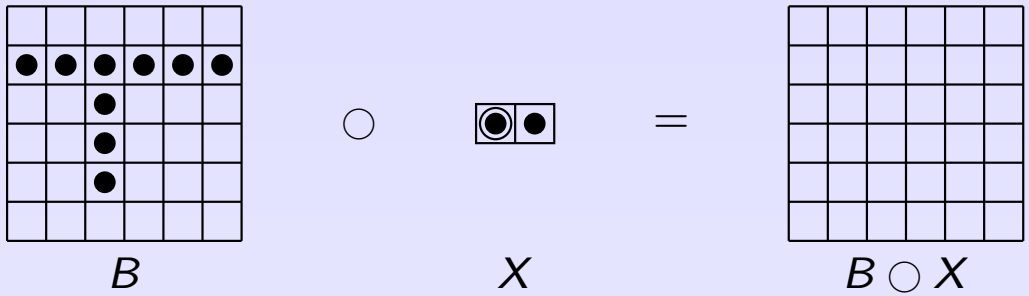
5. Morphologie

5. Morphologie
5.1 Basisoperationen
5.2 Komplexere Morphologieoperationen

Öffnen - Opening

- Durch Erosion verschwinden unerwünschte Strukturen im Bild ☺ und interessierende Objekte schrumpfen ☹
- Dilatation stellt bereits gelöschte Strukturen teilweise wieder her
- Öffnen = Erosion gefolgt von Dilatation (mit selbem SE)

$$O = B \bigcirc X = (B \ominus X) \oplus X \tag{37}$$

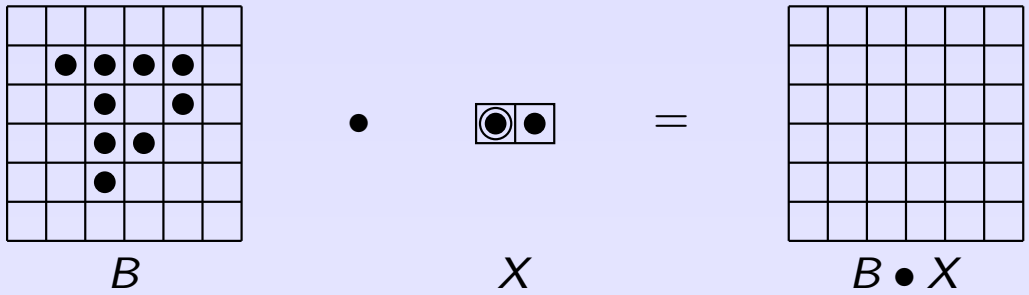


⇒ kleine Objekte verschwinden vollständig
konvexe/zackige Ränder werden abgetragen
Objekte werden getrennt

Schliessen - Closing

- Durch Dilatation verschwinden unerwünschte Lücken im Objekt ☺ und Objekte werden aufgebläht ☹
- Erosion reduziert zuvor aufgeblähte Ränder
- Schließen = Dilatation gefolgt von Erosion (mit selbem SE)

$$O = B \bullet X = (B \oplus X) \ominus X \tag{38}$$



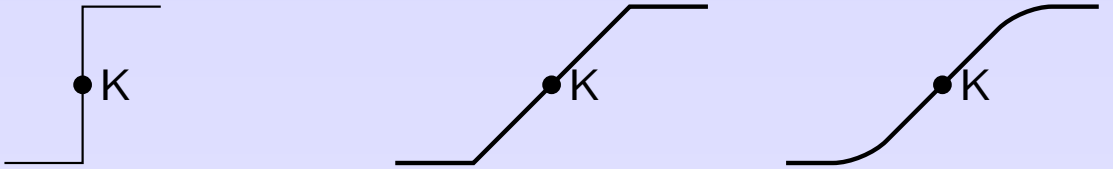
⇒ Löcher und Lücken werden aufgefüllt
aneinanderliegende Objekte wachsen zusammen

6. Kantendetektion

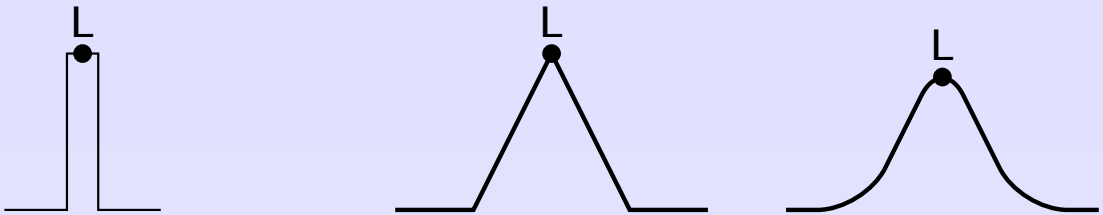
6. Kantendetektion
6.1 Begriffe, Gradienten
6.2 Konturaufbesserung

Kanten

- Grenze zwischen homogenen Flächen
- Diskontinuität im Werteverlauf (Grauwert, Textur oder Farbe)



Idealisierte Darstellung verschiedener Grauwertprofile von Kanten



Idealisierte Darstellung verschiedener Grauwertprofile von Linien

- reale Kante: überlagert mit Störungen

Detektion von Kanten

Ziele:

- geringe Fehlerrate (finde alle Kantenpunkte, ignoriere Nichtkantenpunkte)
- Kanten sollten nur 1 Pixel breit sein (kein Verschmieren)
- Detektion an wirklicher Position
- Kanten sollten i.d.R. geschlossene Linienzüge sein
- Numerische Kriterien: schnelle Berechenbarkeit, Ganzzahlarithmetik
- Problem: Unterscheidung zwischen zufälligen Signaländerungen und gesuchten Diskontinuitäten, die mit realen Kanten korrespondieren

Generelle Vorgehensweise:

1. Glättung: Rauschen im Originalbild führt zu Artekfakten
2. Kantendetektion
3. Konturaufbesserung: Ausdünnung breiter Kanten, Eliminierung kurzer Kantenstücke
4. Konturpunktverkettung: Verbindung der Konturpunkte zu geschlossenen Kantenzügen

Detektion von Kanten

- Kante = starke Grauwertänderung $\rightarrow$ Gradient

$$\text{grad}\mathcal{I}(x,y) = \left(\frac{\partial \mathcal{I}}{\partial x}, \frac{\partial \mathcal{I}}{\partial y}\right)^{\mathrm{T}} \tag{39}$$

- Gradient in Richtung der stärksten Änderung, Betrag proportional zur Grauwertänderung
- Diskrete Ableitung erster Ordnung $\rightarrow$ Differenzengleichung

$$\frac{\partial \mathcal{I}}{\partial x} \approx g_x = \frac{\mathcal{I}(x,y) - \mathcal{I}(x-\Delta x, y)}{\Delta x} \tag{40}$$

- Mit $\Delta x = 1$ erhält man: $g_x = \mathcal{I}(x,y) - \mathcal{I}(x-1,y)$, bzw. als Faltungsmatrix

$$M_x = \begin{array}{|c|c|}\hline -1 & 1 \\ \hline \end{array} \qquad M_y = \begin{array}{|c|}\hline -1 \\ \hline 1 \\ \hline \end{array}$$

- Gradientenbetrag: $\sqrt{g_x^2 + g_y^2}$
- Richtung: $\arctan \frac{g_x}{g_y}$

Weitere Vektorgradienten

- Zentrumssymmetrischer Gradient

$$M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array} \qquad M_y = \begin{array}{|c|}\hline -1 \\ \hline 0 \\ \hline 1 \\ \hline \end{array}$$

- Bildsignale oft verrauscht $\rightarrow$ größere Faltungsmatrizen mit integrierter Filterung
- Prewitt-Operator

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

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

  Mittelwertbildung in 3-Punkte Nachbarschaft orthogonal zur Differenzierungsrichtung
- Sobel-Operator (weiteste Verbreitung)

1	0	-1
2	0	-2
1	0	-1

1	2	1
0	0	0
-1	-2	-1

  Glättung orthogonal zur Differenzierungsrichtung mit Binomial-Filtermaske.
- Prewitt- und Sobel-Operator wirken am stärksten auf horizontale und vertikale Kanten $\rightarrow$ Richtungsabhängigkeit

Kompassgradienten

- Kompassgradienten berechnent Bilddifferenzen in mehreren Richtungen
- Filtermasken als Sätze von Schablonen für verschiedene Orientierungen
- Gradientenbetrag = Maximum
- Kirsch-Operator

-3	-3	5
-3	0	5
-3	-3	5

-3	5	5
-3	0	5
-3	-3	-3

5	5	5
-3	0	-3
-3	-3	-3

5	5	-3
5	0	-3
-3	-3	-3

5	-3	-3
5	0	-3
5	-3	-3

-3	-3	-3
5	0	-3
5	5	-3

-3	-3	-3
-3	0	-3
5	5	5

-3	-3	-3
-3	0	5
-3	5	5

- Qualität der Kompassgradienten vergleichbar mit Vektorgradienten
- Vorteil: direkte Ermittlung der lokalen Kantenrichtung (hier: 8 Richtungen)
- Genauigkeit abhängig von Anzahl und Größe der Masken

Ableitungen 2. Ordnung

- 2. Ableitung = zweimalige 1. Ableitung

$$\begin{aligned}\frac{\partial^2 \mathcal{I}}{\partial x^2} &= g_{x+1} - g_x \Big(\mathcal{I}(x+1,y) - \mathcal{I}(x,y)\Big) - \Big((\mathcal{I}(x,y) - \mathcal{I}(x-1,y)\Big) \\ &= \mathcal{I}(x+1,y) - 2\,\mathcal{I}(x,y) + \mathcal{I}(x-1,y)\end{aligned} \tag{41}$$

- Faltungsmatrizen: $M_x =$ | 1 | -2 | 1 | $\quad M_y =$ | 1 | -2 | 1 |

- Laplace-Operator: Summe beider partiellen Ableitungen 2. Ordnungen nach x und y

0	-1	0
-1	4	-1
0	-1	0

bzw.

-1	-1	-1
-1	8	-1
-1	-1	-1

(8er-Nachbarschaft)

- Anwendung: Suche nach Nulldurchgängen im Ergebnisbild oder Schwellwertverfahren auf Betrag der Ergebnisse
- Anfällig gegen Bildstörungen, Einzelpunkte viermal/achtmal stärker detektiert als Linien
- Rauschen erscheint als einzelne Punkte, die in Helligkeit variieren

LOG - Laplacian of Gaussian

- Kombination aus Tiefpass/Glättung und Hochpass/Kontrastverstärkung
- Gaussfunktion $G(x,y) = \frac{1}{2\pi\sigma^2}\, e^{-(x^2+y^2)/2\sigma^2}$
- Zweite Ableitung: $\frac{1}{\pi\sigma^2}\left(\frac{x^2+y^2}{2\sigma^2} - 1\right) e^{-(x^2+y^2)/2\sigma^2}$
- Nulldurchgangssuche lokal in 3×3-Umgebung
- Detektion von Nullstellen
  $\rightarrow$ Faktor $\frac{1}{\pi\sigma^2}$ kann entfallen
- Nulldurchgänge stets 1-Pixel breit
  $\rightarrow$ keine Ausdünnung nötig
- LOG liefert kein Maß für Kantenstärke
  $\rightarrow$ Detektion leichter Farbunterschiede als Kanten

0	0	-1	-2	-3	-4	-3	-2	-1	0	0
0	-1	-4	-8	-13	-15	-13	-8	-4	-1	0
-1	-4	-11	-21	-27	-27	-27	-21	-11	-4	-1
-2	-8	-21	-26	-7	9	-7	-26	-21	-8	-2
-3	-13	-27	-7	71	125	71	-7	-27	-13	-3
-4	-15	-27	9	125	200	125	9	-27	-15	-4
-3	-13	-27	-7	71	125	71	-7	-27	-13	-3
-2	-8	-21	-26	-7	9	-7	-26	-21	-8	-2
-1	-4	-11	-21	-27	-27	-27	-21	-11	-4	-1
0	-1	-4	-8	-13	-15	-13	-8	-4	-1	0
0	0	-1	-2	-3	-4	-3	-2	-1	0	0

LOG-Maske σ=1,5, Radius=5

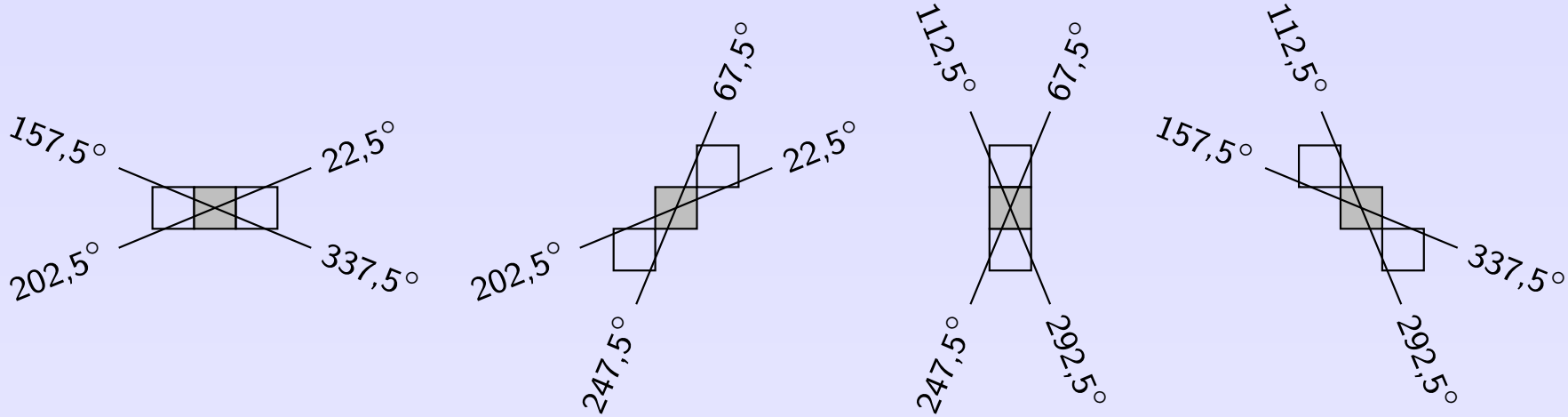
6. Kantendetektion

6. Kantendetektion
6.1 Begriffe, Gradienten
6.2 Konturaufbesserung

Nonmaxima-Unterdrückung (Ausdünnung breiter Kanten)

- Ziel: Verdünnung von Gradientenbildern quer zur Kantenrichtung
- Konturpunkt bei Maximalwert bleibt unverändert, andere Werte auf 0 gesetzt

1. Ermittle gegenüberliegende Nachbarn quer zur Kantenrichtung, d.h. in Richtung des Gradienten $\alpha = \arctan \frac{g_x}{g_y}$



2. Falls auch Gradientenrichtungen dieser Nachbarn in den Grenzen liegen, weiter bei (3), andernfalls weiter mit nächstem Pixel
3. Ist Gradientenbetrag des aktuellen Pixels kleiner als der beider Nachbarn, setze ihn auf 0
4. Optional: Falls Gradientenbetrag sehr niedrig setze ihn auf 0 trotz lokaler Maximalität

Hysterese-Schwellwert

- Mittels Nonmaxima-Unterdrückung erhebliche Reduzierung potenzieller Kantenpunkte
- Sind verbleibende Punkte tatsächlich Kantenpunkte oder durch Rauschen detektiert?
- Entscheidung mittels Gradientenbetrag und Schwellwerten T_{low} und T_{high}

$$\text{Gradient} = \begin{cases} \textit{low} & \rightarrow \text{ keine Kante} \\ \textit{medium} & \rightarrow \text{ vielleicht Kante} \\ \textit{high} & \rightarrow \text{ sicherer Kantenpunkt} \end{cases}$$

Verfahren

- Verwende Pixel mit grad $\geq T_{\text{high}}$, um neue Konturen zu beginnen
- Benutze Pixel mit $T_{\text{low}} \leq \text{grad} < T_{\text{high}}$, um Konturen fortzusetzen
- Verwerfe Pixel mit grad $< T_{\text{low}}$

Beobachtung

- Konturen ausschließlich zwischen T_{low} und T_{high} werden verworfen
- Ein einziger Punkt mit grad $> T_{\text{high}}$ genügt, um neue Konturen zu beginnen

Canny-Operator

7. Merkmalsextraktion

7. Merkmalsextraktion
7.1 Vorüberlegungen
7.2 Geometrische Merkmale

Merkmale

- Zur automatischen Erkennung von Bildern müssen Merkmale extrahiert werden
- Merkmale sind komprimierte Informationen des Bildinhaltes
- Voraussetzung i.d.R. Segmentierung (Merkmale nur für bestimmte Bildbereiche)
- Visuelle Merkmale: Fläche, Form, Farbe, Umfang, Lage, ...
- Mathematische Merkmale: Frequenzen ...
- Art und Anzahl abhängig von Aufgabe und Klassifikator

Allgemeine Anforderungen:

- robust (Bounding Box bei Rotation)
- Einfach zu berechnen
- Einfach zu verarbeiten (Kein Histogramm mit 256 Werten)
- Aussagekräftig (Unterscheidbarkeitskriterium für Folgeschritte)

Für bestimmtes visuelles Kriterium (z.B. Form) stehen meist unterschiedliche Merkmale zur Verfügung (Kontur, Bounding Box, Kompaktheit), die sich in Berechenbarkeit, Robustheit und Genauigkeit unterscheiden.

Werte

Grauwert

- Mittlerer Grauwert und Varianz
- Histogramm (ohne räumliche Information, reduzierte Anzahl Klassen)
- Grauwertverteilung (räumlich gefiltert oder reduzierte Auflösung)

Farbe

- Mittlerer Farbwert (RGB, HSV, . . .)
- Farbhistogramm (ohne räumliche Information)
- Farbverteilung (räumlich gefiltert oder reduzierte Auflösung)

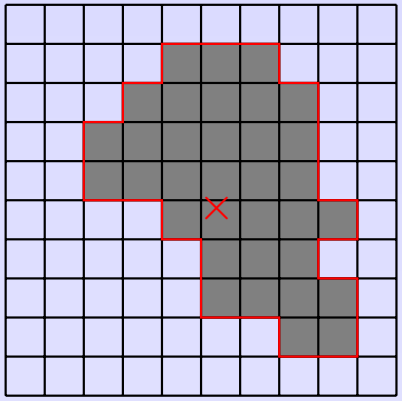
Textur

- Haralick-Merkmale

7. Merkmalsextraktion

7. Merkmalsextraktion
7.1 Vorüberlegungen
7.2 Geometrische Merkmale

Geometrie - Größe & Position



n=34
Schwerpunkt=$(5,38/4,82)$

- Flächeninhalt (Anzahl n der Pixel zählen)

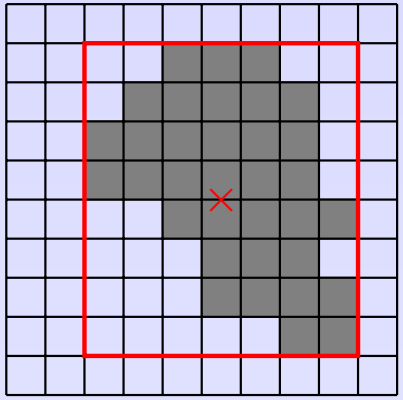
$$n = \sum_{(x,y)} \mathcal{B}(x,y) \text{ mit Binärbild } \mathcal{B}(x,y) \in \{0,1\} \tag{42}$$

Objektpunkte haben den Wert $\mathcal{B}(x,y) = 1$, Hintergrundpunkte $\mathcal{B}(x,y) = 0$
abhängig von Entfernung der Kamera

- Position, Schwerpunkt

$$\frac{1}{n} \sum_{\mathcal{B}(x,y)=1} (x_i, y_i) \tag{43}$$

Geometrie - Bounding Box

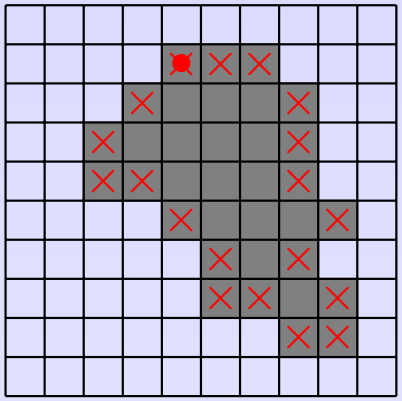


- kleinstes umschreibendes, achsenparalleles Rechteck

$$(x_{\min}, y_{\min}), (x_{\max}, y_{\max}) \tag{44}$$

- Abgeleitete Größen: Zentrum, Höhe, Breite, Größe, Form
- einfache, schnelle Berechnung
- nicht rotationsinvariant, abgeleitete Größen ungenau

Geometrie - Kontur



Kettencode=00766757643423342 11
Randpunkte=19
Umfang=$10 \cdot 1 + 9 \cdot \sqrt{2} = 22,6$
Kompaktheit=$\frac{4\pi \cdot 34}{22,6^2}$=0,84

3	2	1
4		0
5	6	7

Kettencode

- Kontur: Kettencode/Richtungscode nach Freeman
- Umfang: $U = \#_{0246} + \sqrt{2} \cdot \#_{1357}$
- Kompaktheit: berechnet sich aus Fläche n und Umfang U

$$\frac{4\pi n}{U^2} \in [0, 1] \tag{45}$$

- Konvexe Hülle / kleinstes umschreibendes Polygon
- Haupt- und Nebenachse so wie Orientierung der äquivalenten Ellipse [1]

[1] etwas aufwändiger als Bounding Box, aber genauer und rotationsinvariant

Geometrie - Momente

- Die Momente m_{pq} eines Bildes $\mathcal{I}(x, y)$ sind definiert durch

$$m_{pq} = \sum_{(x,y)} x^p y^q \cdot \mathcal{I}(x, y) \text{ mit } p, q = 0, 1, 2 \ldots \tag{46}$$

- Fläche: m_{00}
- Schwerpunkt: $x_s = \frac{m_{10}}{m_{00}} \qquad y_s = \frac{m_{01}}{m_{00}}$

  (Geometrischer Schwerpunkt für $\mathcal{R}(x, y) = 1$, Massenschwerpunkt für $\mathcal{R}(x, y) \in [1, 255]$)
- Zentralmomente (verschiebungsinvariant)

$$\mu_{pq} = \sum_{(x,y)} (x - x_s)^p (y - y_s)^q \cdot \mathcal{I}(x, y) \text{ mit } p, q = 0, 1, 2 \ldots \tag{47}$$

- Mittels normierter Zentralmomente lassen sich Formen invariant gegenüber Verschiebung, Drehung und Skalierung beschreiben

8. Klassifikation

8. Klassifikation
8.1 Klassifikation

Allgemeines

Klassifikation: Objekte in Klassen einteilen
Voraussetzung: Objekte durch n-dimensionale Merkmalsvektoren im n-dimensionalen Merkmalsraum unterscheidbar

Effizienz: Merkmalsraum so klein wie möglich, nicht alle detektierten Merkmale sind aussagekräftig $\rightarrow$ Merkmalsauswahl (z.B. durch Diskriminanzanalyse)

Cluster: Merkmalsvektoren von ähnlichen Objekten bilden Cluster im Merkmalsraum (Klassen)

Überwachtes Lernen: beginnt mit Trainingsphase zum Erlernen der Eigenschaften unterschiedlicher Objekte (vorzuziehen, wenn Trainingsdaten vorhanden)
Beispiel: Zur automatischen Leukämiediagnose stellt Experte/Arzt je einen Datensatz gesunder und erkrankter Zellen zur Verfügung.

Unüberwachtes Lernen: ermittelt aus Merkmalsverteilung die Cluster (Anzahl und Zentren) von ähnlichen Objekten
Beispiel: Schwellwertverfahren nach Otzu klassifiziert jeden Pixel im 1-dimensionalen Merkmalsraum in die Klassen *hell* oder *dunkel*, wobei die Trennlinie der Klassen aus dem Histogramm ermittelt wird.

Klassifikatoren: unterscheiden sich in Beschreibung der Musterklassen und Zuordnung eines unbekannten Objektes

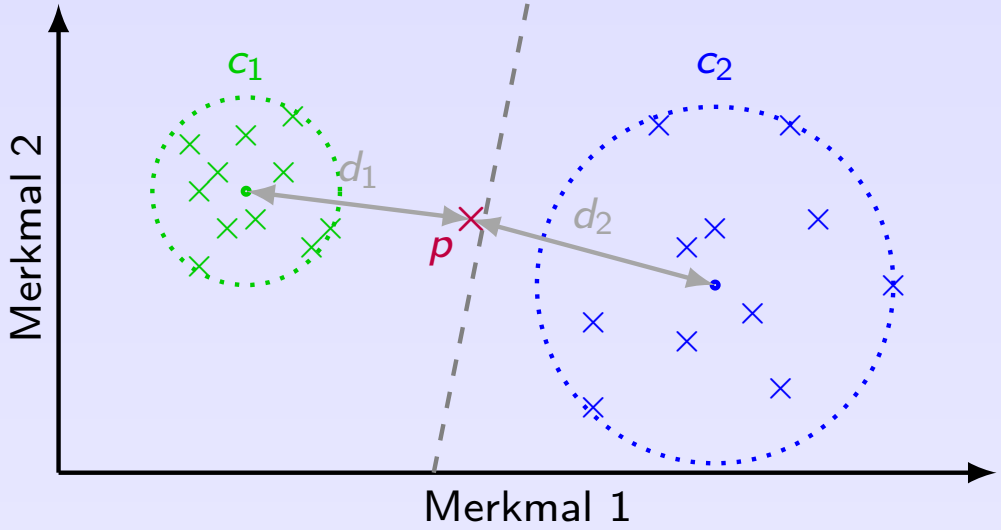
Abstandsklassifikator (Euklid)

Sei $p \in \mathbb{R}^n$ ein n-dimensionaler Merkmalsvektor
$c_i \in \mathbb{R}^n$ ist das Zentrum der i-ten Klasse im selben Merkmalsraum

Euklid'scher Abstand:

$$d_{\mathrm{E}}(p, c_i) = \sqrt{(p - c_i)^T (p - c_i)} \tag{48}$$

Der Klassifikator ordnet p der Klasse c_i mit minimalem Abstand $d_{\mathrm{E}}(p, c_i)$ zu!



$c_1 = (20, 30)$
$c_2 = (70, 20)$
$p = (44, 27)$
$d_1 =$
$d_2 =$
p wird Klasse zugeordnet

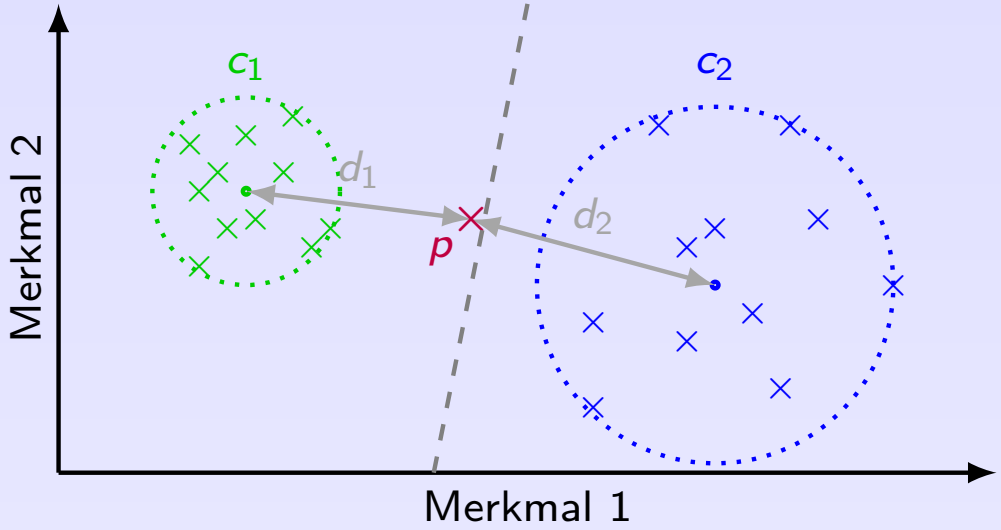
Abstandsklassifikator (Euklid)

Sei $p \in \mathbb{R}^n$ ein n-dimensionaler Merkmalsvektor
$c_i \in \mathbb{R}^n$ ist das Zentrum der i-ten Klasse im selben Merkmalsraum

Euklid'scher Abstand:

$$d_{\mathrm{E}}(p, c_i) = \sqrt{(p - c_i)^T (p - c_i)} \tag{48}$$

Der Klassifikator ordnet p der Klasse c_i mit minimalem Abstand $d_{\mathrm{E}}(p, c_i)$ zu!



$c_1 = (20, 30)$
$c_2 = (70, 20)$
$p = (44, 27)$
$d_1 =$
$d_2 =$
p wird Klasse zugeordnet

- Trennflächen zwischen zwei Klassen bilden Punkte gleichen Abstands von beiden Klassen.
- Klasse nur durch Mittelwert charakterisiert

\+ Prinzipiell genügt einzelne Stichprobe zum Trainieren

\+ Sehr schneller Klassifikator

– Streuung und Verteilung der Klassen unberücksichtigt

Kovarianzmatrix

- Streuung einer 1D-Zufallsvariable $x_1 \in \mathbb{R}$ beschrieben durch Varianz
  $Var(x_1) = \sigma^2 = \frac{1}{N} \sum (x_{1i} - \overline{x}_1)^2$
- Kovarianzmatrix Σ beschreibt Streuung und Korrelation für Zufallsvektor
  $X = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$

$$\Sigma = \begin{pmatrix} Var(x_1) & Cov(x_1, x_2) & \ldots \\ Cov(x_2, x_1) & Var(x_2) & \ldots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

mit $Cov(x_1, x_2) = \frac{1}{N} \sum (x_{1i} - \overline{x}_1) \cdot (x_{2i} - \overline{x}_2)$ usw.

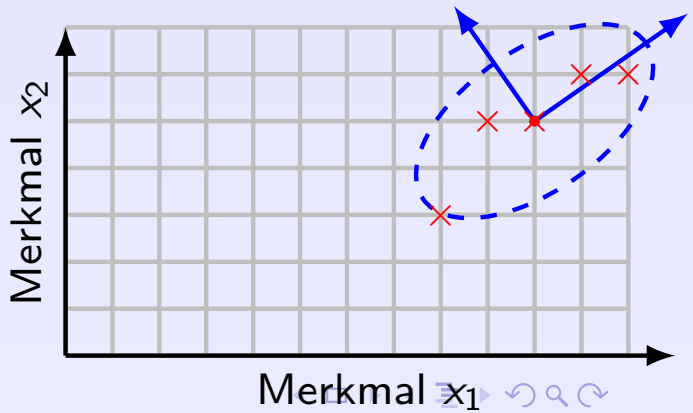
Hauptdiagonale = Varianzen der einzelnen Komponenten/Merkmale

Σ ist symmetrisch $\Leftrightarrow Cov(x_1, x_2) = Cov(x_2, x_1)$

$Cov = 0$ gdw. betreffende Merkmale voneinander unabhängig

Berechne Σ für Klasse mit den Merkmalsvektoren (8,3), (9,5), (10,5), (11,6), (12,6)

$\overline{x}_1 =$
$\overline{x}_2 =$
$Var(x_1) =$
$Var(x_2) =$
$Cov(x_1, x_2) =$



Abstandsklassifikator (Mahalanobis)

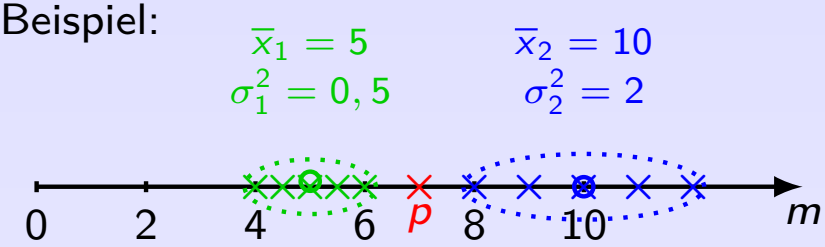
Sei $p \in \mathbb{R}^n$ ein n-dimensionaler Merkmalsvektor
$c_i \in \mathbb{R}^n$ ist das Zentrum der i-ten Klasse im selben Merkmalsraum mit Kovarianzmatrix Σ_i
Dann ist der Mahalanobis-Abstand eines Merkmalsvektors p zur Klasse c_i

Mahalanobis Abstand:

$$d_{\mathrm{M}}(p, c_i) = \sqrt{(p - c_i)^T \Sigma_i^{-1} (p - c_i)} \tag{49}$$

Der Klassifikator ordnet p der Klasse c_i mit minimalem Abstand $d_{\mathrm{M}}(p, c_i)$ zu

Beispiel:



	c_1	c_2
d_{E}		
d_{M}		

■

D.h. falls ein Merkmal m stark streut ($Var(m)$ ist groß), ist sein Beitrag zum Abstand sehr klein (Inverse von Σ). Eine Abweichung von $\overline{m}$ wird also eher toleriert.

Abstandsklassifikator (Mahalanobis)

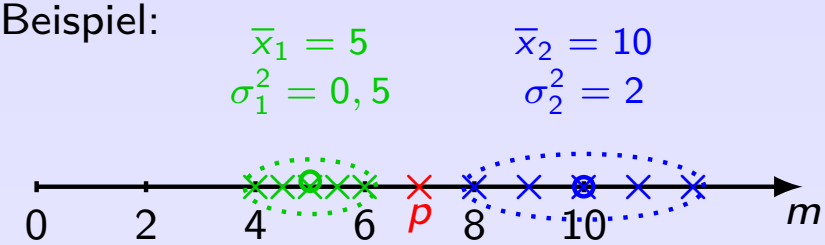
Sei $p \in \mathbb{R}^n$ ein n-dimensionaler Merkmalsvektor
$c_i \in \mathbb{R}^n$ ist das Zentrum der i-ten Klasse im selben Merkmalsraum mit Kovarianzmatrix Σ_i
Dann ist der Mahalanobis-Abstand eines Merkmalsvektors p zur Klasse c_i

Mahalanobis Abstand:

$$d_{\mathrm{M}}(p, c_i) = \sqrt{(p - c_i)^T \Sigma_i^{-1} (p - c_i)} \tag{49}$$

Der Klassifikator ordnet p der Klasse c_i mit minimalem Abstand $d_{\mathrm{M}}(p, c_i)$ zu

Beispiel:



	c_1	c_2
d_{E}		
d_{M}		

■

D.h. falls ein Merkmal m stark streut ($Var(m)$ ist groß), ist sein Beitrag zum Abstand sehr klein (Inverse von Σ). Eine Abweichung von $\overline{m}$ wird also eher toleriert.

- Trennflächen zwischen zwei Klassen bilden Punkte gleichen Abstands von beiden Klassen.
- Klasse durch Mittelwert und Verteilung charakterisiert

\+ Stichprobe zum Trainieren muss auch Streuung abdecken

\+ Berechnung von Σ_i^{-1} apriori $\rightarrow$ Sehr schneller Klassifikator

Ergänzungen

Normierung der Merkmale

- Bei manchen Abstandsmaßen (z.B. Euklid) sollten Merkmale möglichst gleichen Wertebereich haben!
- Beispiel:
  Merkmalsraum = Kompaktheit $\in [0,1]$ + Grauwert $\in [0,255]$
  Skalierung der Kompaktheit auf $[0,255]$ oder
  Skalierung der Grauwerte auf $[0,1]$ oder
  gewichteter Euklid'scher Abstand $d = \sqrt{\frac{1}{1}(p_1 - \overline{x}_1)^2 + \frac{1}{255}(p_2 - \overline{x}_2)^2}$

Zurückweisungsklasse

- falls Abstand d von p zu allen Klassen c_i sehr groß
- Zurückweisungsradius r. Zurückweisung, falls $d_i > r \quad \forall d_i = d(p, c_i)$
- Approximation der Musterklassen durch n-dim Kugeln im n-dim Merkmalsraum

Nearest Neighbor Klassifikator

Nearest Neighbor Klassifikator

- Merkmalsvektor wird Klasse mit ähnlichsten Trainingsvektor zugeordnet
- Ähnlichkeitsmaß z.B. Euklidischer Abstand
- Aufwand steigt mit Anzahl der Trainingsvektoren
- Gefahr der Falschklassifikation bei einzelnem falschen Trainingsvektor

k-Nearest Neighbor Klassifikator

- Zuweisung des Merkmalsvektors zur Klasse mit der Mehrheit seiner k nächsten Nachbarn
- Größerer Aufwand, aber robuster bzgl. einzelner Trainingsvektoren
- Optionale Zurückweisungsklasse, z.B. Mehrheit nicht signifikant