# 2D Graphik: Segmentierung

Vorlesung „2D Graphik”

Andreas Butz, Otmar Hilliges

Freitag, 16. Dezember 2005

# Themen heute

- Segmentierung
  - Region labeling
  - Flood fill
  - Relaxation labeling
- Regionenbasierte Segmentierung
  - Multiskalenstrategien
  - Region Merging
  - Split-and-Merge
  - Textursegmentierung
- Kantenbasierte Segmentierung
  - Edge Linking und Canny Edge Operator
  - Nulldurchgänge
  - Wasserscheidentransformation

# Region Labeling

- Schwellenwert zerlegt das Bild in Vordergrund und Hintergrundsegmente.
- **Region Labeling** bestimmt Ort und Anzahl aller zusammenhängenden Gebiete im Binärbild *b*:

```
region.initialise() // Region der Größe M,N erzeugen und
label=1                  // mit Null initialisieren, Startlabel=1
for (i,j) = 0, (M,N) do   // Doppelschleife über i und j
  if region.labels(i,j) = 0 then // dieser Ort ist noch nicht
                                    Teil einer Region
    label = label+1                  // neues Label vergeben
    region.flood_fill(i,j,label)  // zusammenhängendes
                                    Gebiet um (i,j) mit
                                    Label füllen
```

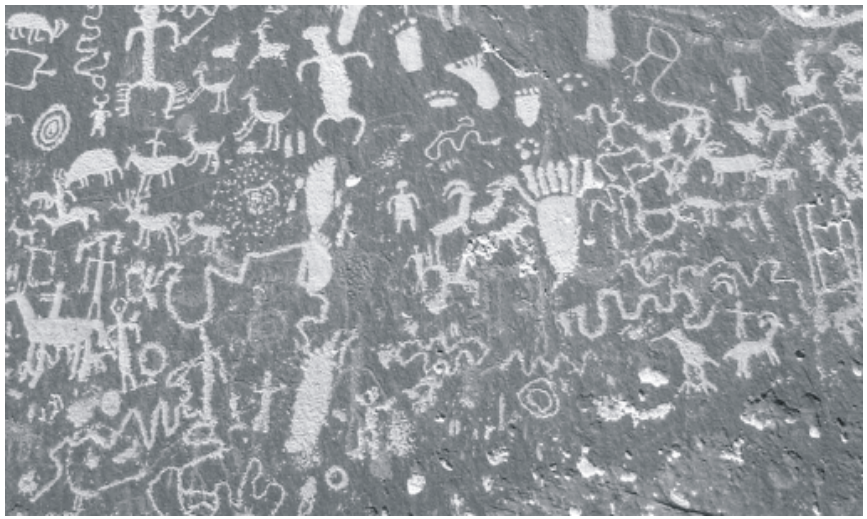
# Flood Fill

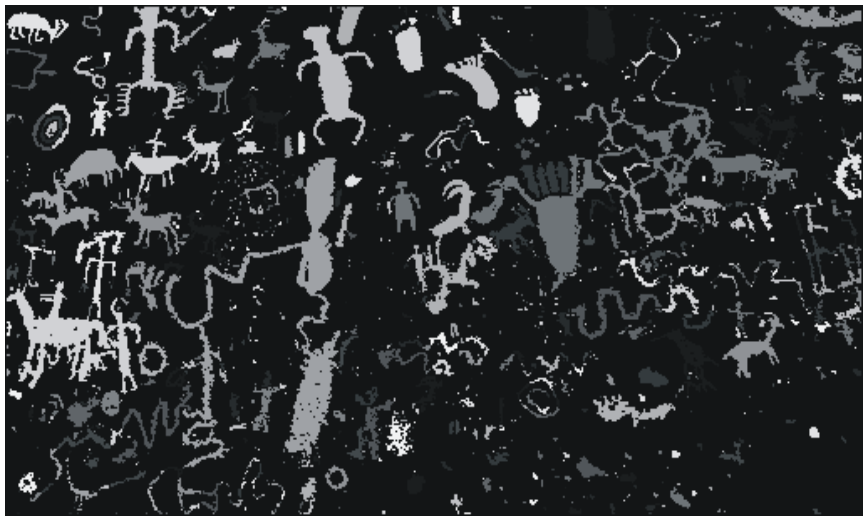
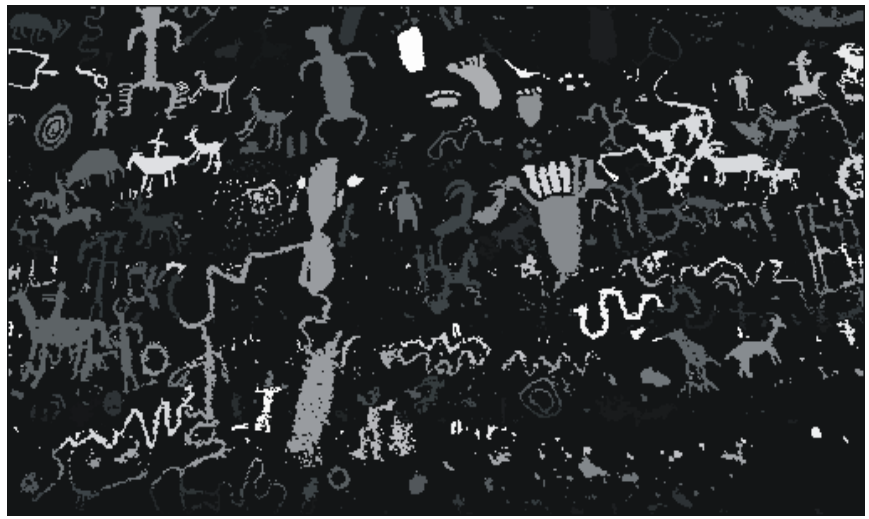
```
flood_fill(i,j,label)  // Variablen zur Auswertung der
        Zusammenhangsbedingung sind global verfügbar
if f(i,j) erfüllt Zusammenhangsbedingung then
    region(i,j) = label      // Region an (i,j) mit label
                             // versehen
    flood_fill(i-1,j,label) // Nachbarpixel untersuchen
    flood_fill(i,j-1,label)
    flood_fill(i+1,j,label)
    flood_fill(i,j+1,label)
```

Region labeling + flood fill:
- Vollständige Segm.?
- Überdeckungsfrei?
- Zusammenhängend?

Bsp. für Zusammenhangsbedingung:
- hat den gleichen Grauwert wie Saatpunkt

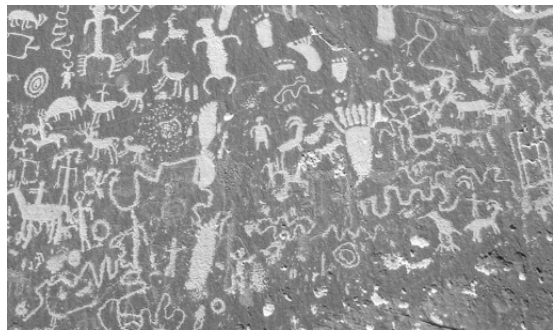
# Resultat



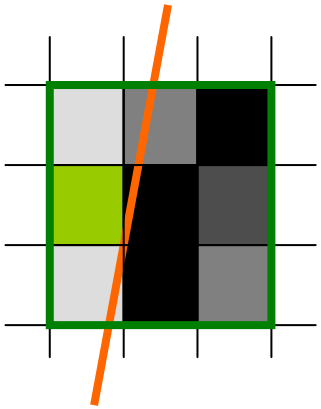
# Nachverarbeitung

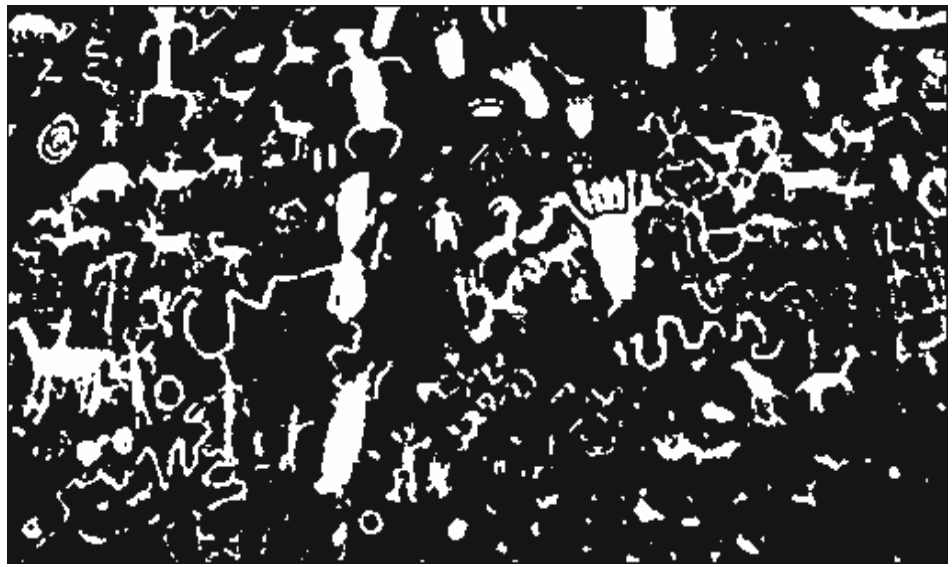
- Trennung nach Grauwerten wird nicht perfekt sein.
- Schwellenwertbild enthält falsche Regionen:
  - kleine fälschlicherweise als Segmente identifizierte Regionen.
  - Störungen am Rand von Regionen.
- Nachverarbeitung
  - Medianfilterung auf den Labeln
  - Entfernung von zu kleinen Regionen
  - Relaxation Labeling



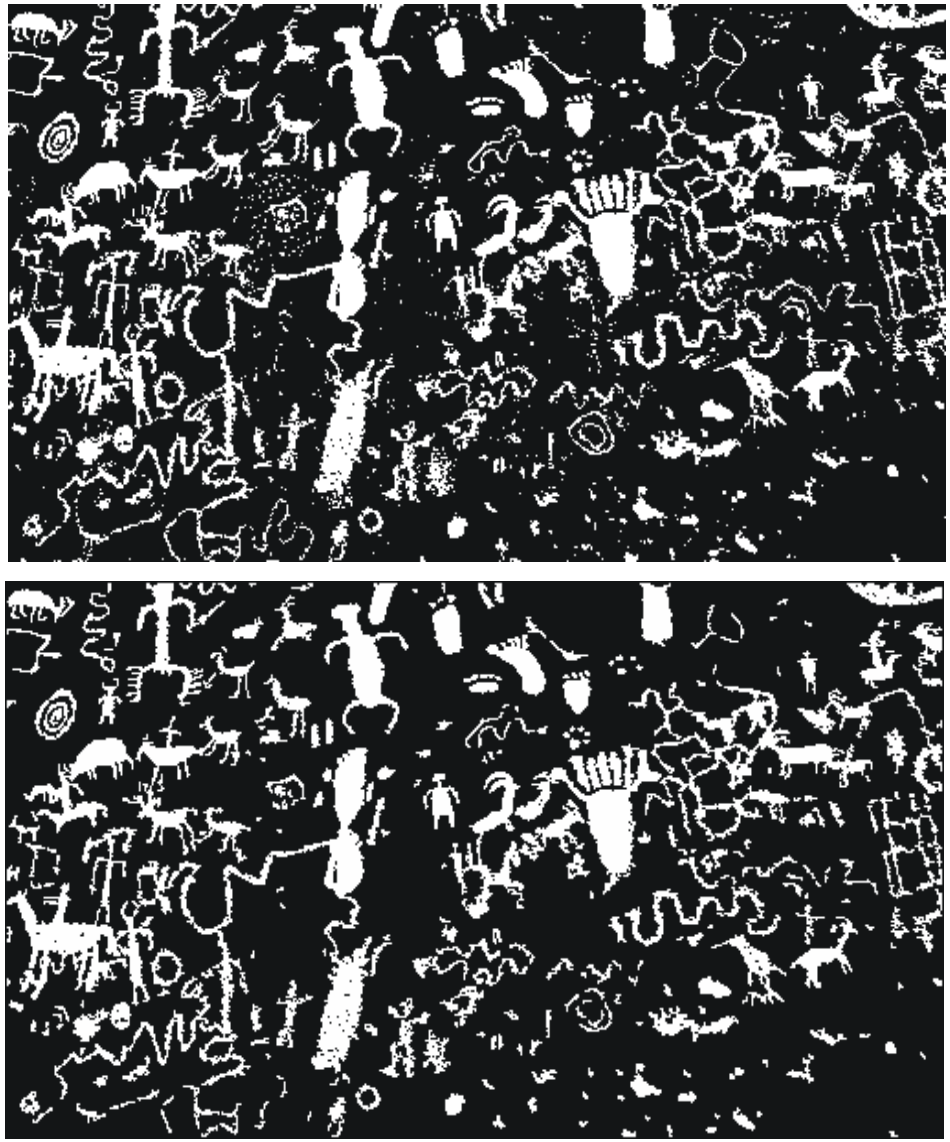
# Medianfilterung auf Labeln

Wieso geht das überhaupt?
→ siehe Def. Medianfilter





# Entfernung kleiner Gebiete



# Relaxation Labeling

- Jedes Pixel erhält jedes Label mit einer vorgegebenen Wahrscheinlichkeit

  **(Bsp.: Schwellenwertresultat, Wahrscheinlichkeiten „70% weiß“ und „30% schwarz“ für weiße Pixel; umgekehrt für schwarze Pixel)**
- Benachbarte kompatible Pixel unterstützen sich.
- Relaxationsprozess: Zuordnungswahrscheinlichkeiten ändern sich mit dem Maß der Unterstützung.
- Zu definieren:
  - Kompatibilität
  - Einfluß der Kompatibilität auf die Labelwahrscheinlichkeiten.

# Labelwahrscheinlichkeit

- Aufzählung aller Pixel in Liste $p_0, p_1, \ldots, p_{NM-1}$.
- K verschiedene Label
- Initiale Labelwahrscheinlichkeit $P^0$ für jedes Pixel, und jedes Label $l_k$ vergeben, z.B.

$$P^0(p_i, l_k) = \begin{cases} 0.8 & , \text{falls} l_k = l(p_i) \\ 0.2 & , \text{sonst} \end{cases}$$

Labelwahrscheinlichkeit darf nicht 0 oder 1 sein (Gewissheiten werden nicht verändert)

# Kompatibilität

- Ein Pixel $p_i$ mit Label $l_k$ hat eine Kompatibilität $r$ mit einem Pixel $p_j$, dessen Label $l_l$ sei:

$$r\big((p_i, l_k), (p_j, l_l)\big)$$

- Kompatibilitätskoeffizient für Binärbilder (K=2 Label) z.B.

$$r\big((p_i, l_k), (p_j, l_l)\big) = r(l_k, l_l) = \begin{cases} 1 & , \text{falls } l_k = l_l \\ 0 & , \text{sonst.} \end{cases}$$

(d.h., nur gleiche Label unterstützen sich)

# Unterstützung eines Pixels

- Unterstützung $q^{(n)}$ von Pixel $p_i$ durch Pixel $p_j$ zur Iteration $n$

$$q_j^{(n)}(p_i, l_k) = \sum_{l=0}^{K-1} P^{(n)}(p_j, l_l) \cdot r((p_i, l_k), (p_j, l_l))$$

- Unterstützung von $p_i$ durch alle Pixel

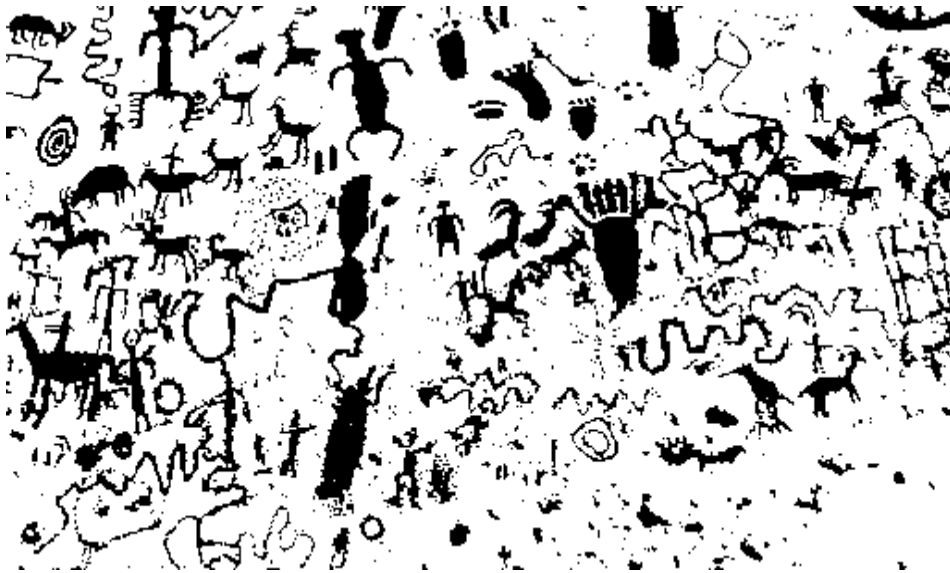
$$Q^{(n)}(p_i, l_k) = \sum_{j=0}^{NM-1} c_{ij} q_j^{(n)}(p_i, l_k)$$

mit Einflussparameter $c_{ij}$, z.B.

$$c_{ij} = \begin{cases} 1/8 & , \text{falls } p_j \in N_8(p_i) \\ 0 & , \text{sonst.} \end{cases}$$

# Iterationsschritt

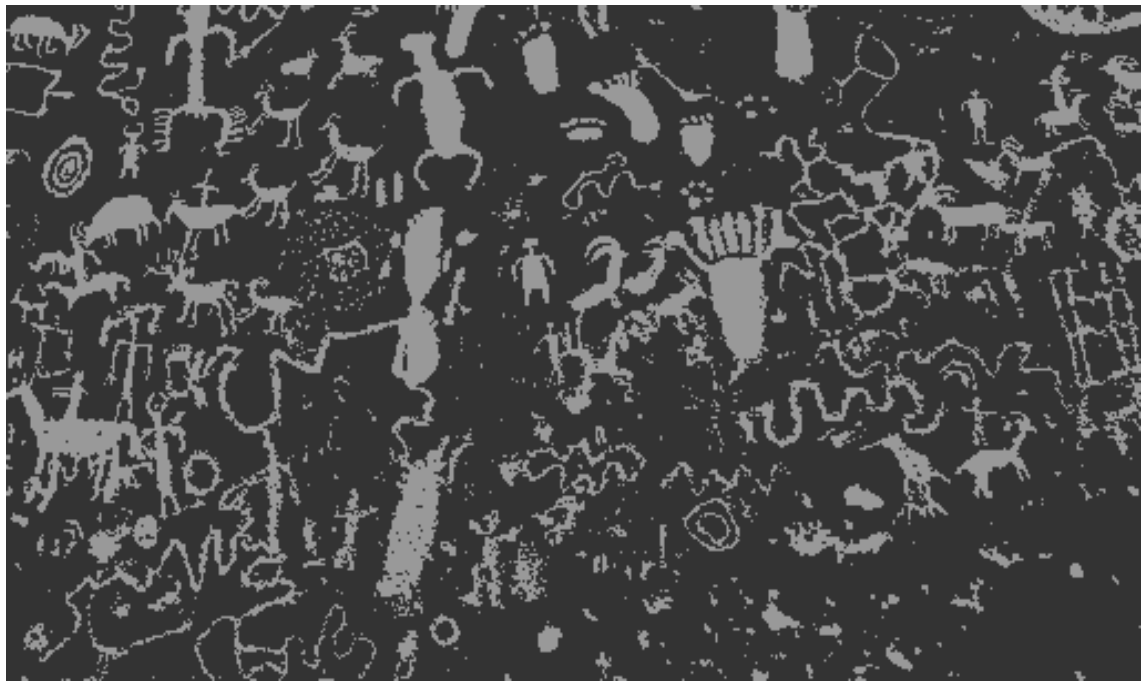
$$P^{(n+1)}(p_i, l_k) = \frac{P^{(n)}(p_i, l_k)\left[1 + Q^{(n)}(p_i, l_k)\right]}{\sum_{l=0}^{K-1} P^{(n)}(p_i, l_l)\left[1 + Q^{(n)}(p_i, l_l)\right]}.$$



# Confidence Map

- Um das Konvergenzverhalten zu beobachten, kann eine **Confidence Map** erzeugt werden.
- Confidence Map: Gibt für jedes Pixel die Zuverlässigkeit der derzeitigen Entscheidung an (weiss = sicher)
- Für 2-Label-Segmentierung: Differenz zwischen gewählten Label und nicht gewähltem Label (evtl. gewichtet mit der Anzahl der Pixel mit diesem Label).

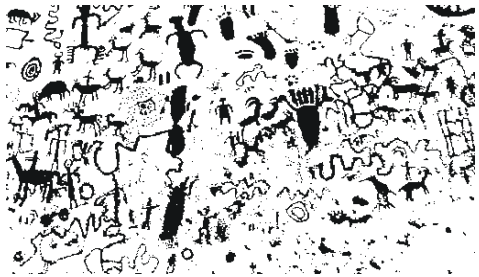
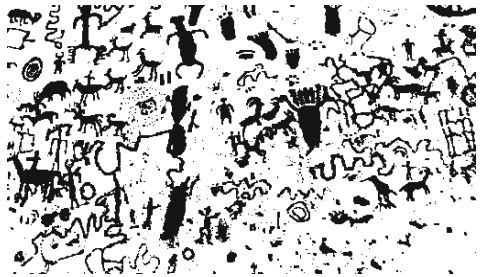
Initiale Confidence Map



Konvergenz

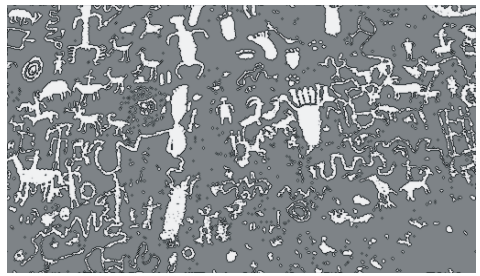
Iteration 0 1 2

Segmentierung

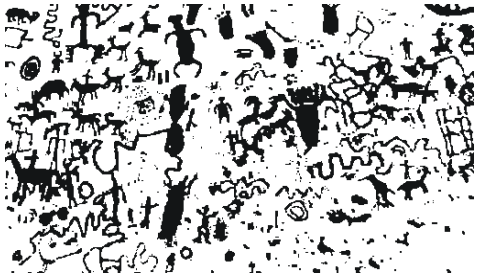
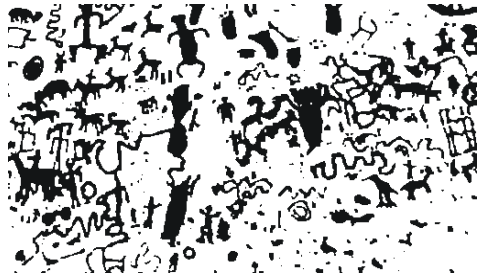
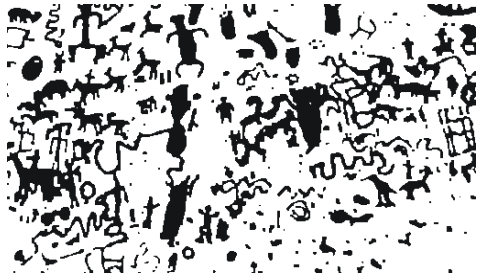




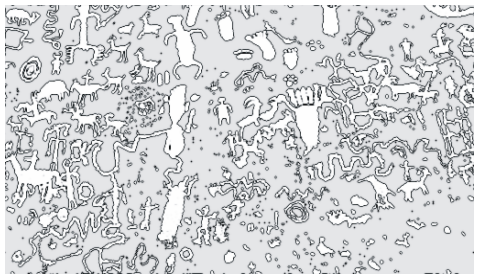
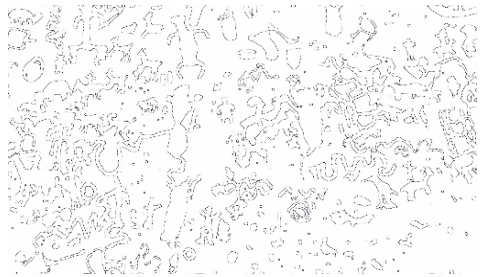
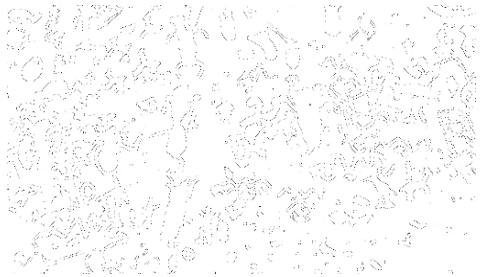
Confidence Map





4 10 20

# ..und nochmal kurz zusammengefasst:

- Letzte Woche:
  - Schwellenwertsegmentierung
  - Shadingkorrektur
- Heute:
  - Region Labeling
  - Nachverarbeitung
  - Relaxation Labeling

# Regionenbasierte Segmentierung

- Multiskalenstrategien
- Region Merging
- Split-and-Merge
- Textursegmentierung

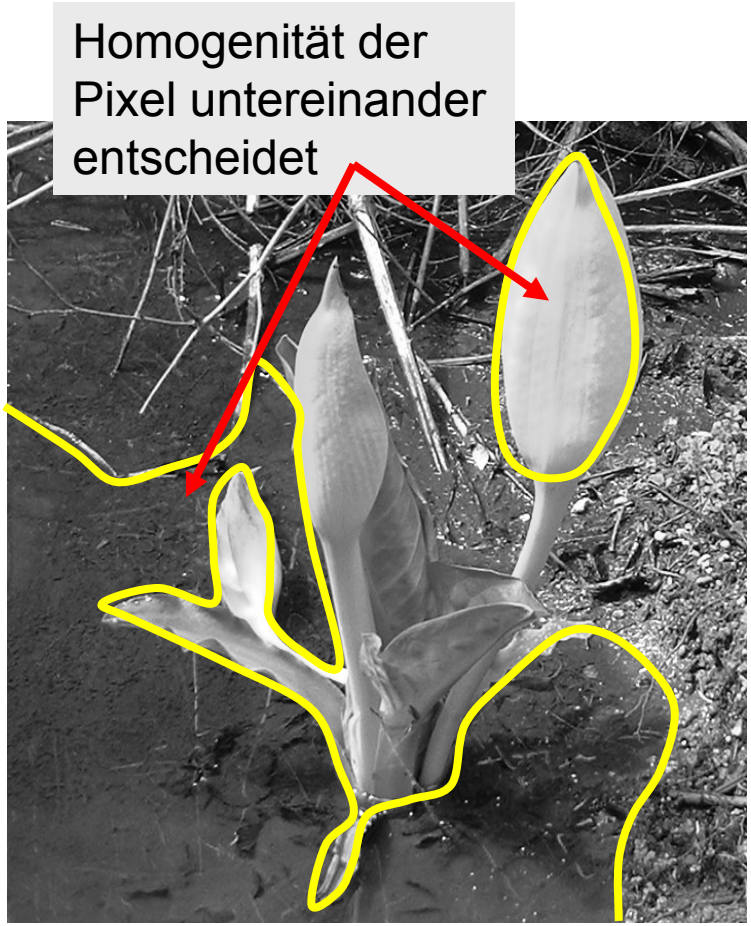
# Regionenbasierte Segmentierung

Homogenität im Inneren des Segments.

Homogenitätsbedingung wird bei der Segmentierung ausgewertet.

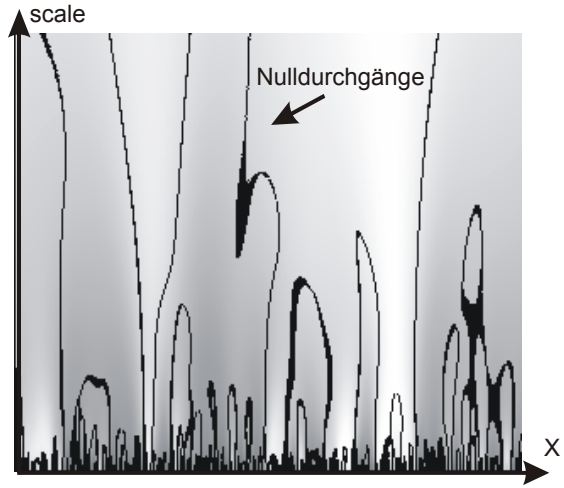
Homogenität ist relativ zu den Attributen eines Segments definiert.

Globale Zusammenhänge über Multiskalenstrategie.



# Multiskalenstrategie

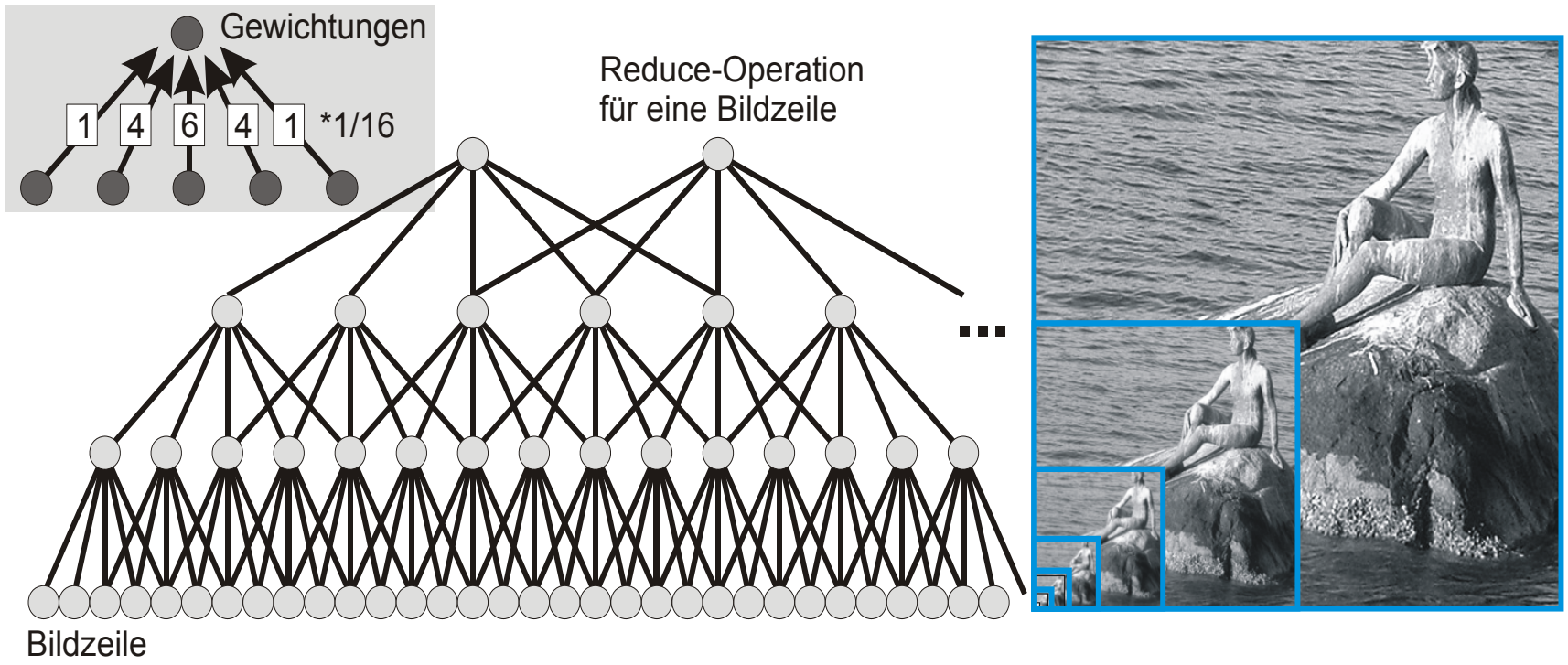
- Relative Kriterien für Homogenität können über unterschiedliche Entfernungen verschieden wirken.
- Segmentierung nach Multiskalenstrategie wertet Kriterien auf unterschiedlichen Skalierungen aus.
- Multiskalenstrategie
  - explizit auf einer Multiskalenrepräsentation
  - implizit in den Segmentierungsalgorithmus integriert.



# Gaußpyramide

- Das Originalbild wird fortlaufend durch eine „reduce“-Operation skaliert.
- Jedes Pixel der nächsthöheren Skalierungsstufe repräsentiert 4 Pixel der aktuellen Stufe.
- Vor der Reduktion wird der Frequenzumfang durch Filterung vermindert:
  - Gaußfilter $\frac{1}{16}\begin{pmatrix}0.87 & 3.91 & 6.44 & 3.91 & 0.87\end{pmatrix}$
  - Binomialfilter $\frac{1}{16}\begin{pmatrix}1 & 4 & 6 & 4 & 1\end{pmatrix}$

# Gaußpyramide



# Expand-Operation

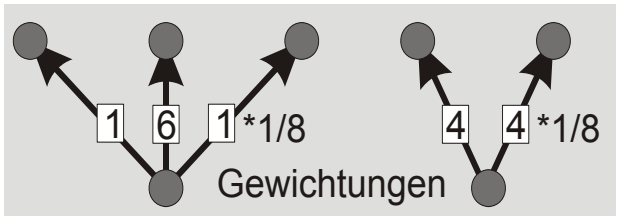
- Um die vorherige Skalierungsstufe zu erzeugen, wird eine „expand“-Operation definiert.
- Pixel der neuen Skalierungsstufe werden durch Interpolation erzeugt:
  - Pixelorte, die auf beiden Skalierungsstufen existieren:

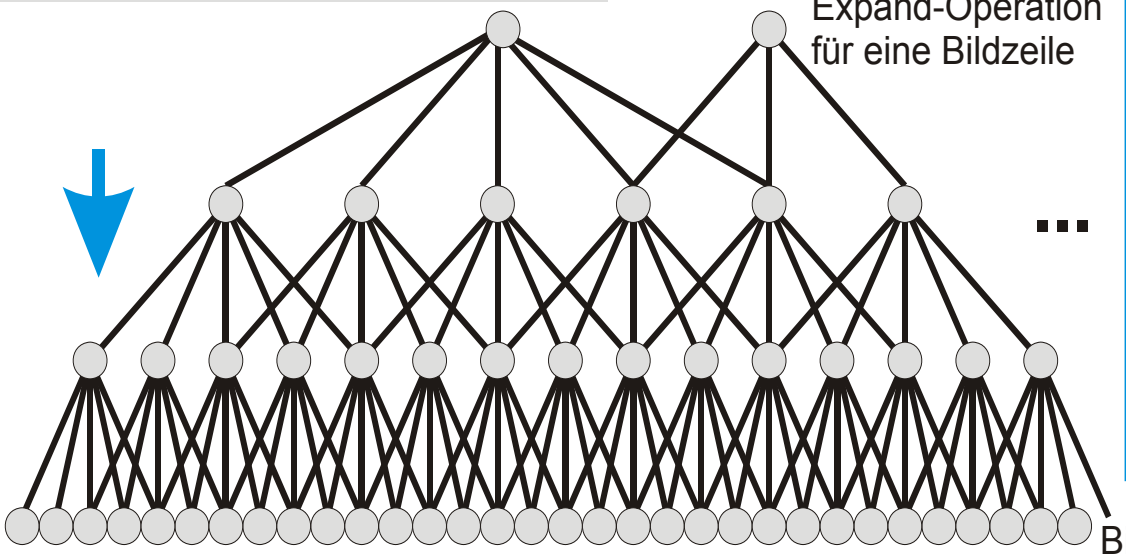
$$\frac{1}{8.18}\begin{pmatrix}0.87 & 6.44 & 0.87\end{pmatrix} \text{ bzw. } \frac{1}{8}\begin{pmatrix}1 & 6 & 1\end{pmatrix}$$

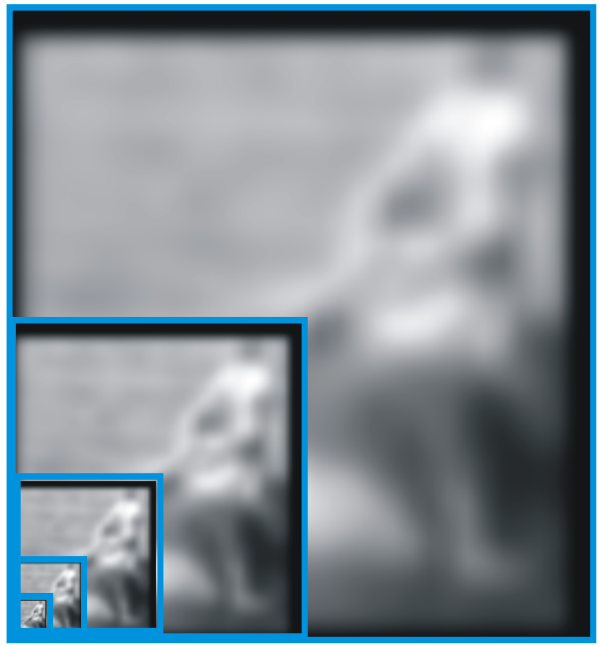
  - Pixelorte, die nur auf der vorherigen Skalierungsstufe existieren:

$$\frac{1}{7.82}\begin{pmatrix}3.91 & 3.91\end{pmatrix} \text{ bzw. } \frac{1}{8}\begin{pmatrix}4 & 4\end{pmatrix}$$

# Expand-Operation



Die Expand-Operation ist nicht verlustfrei

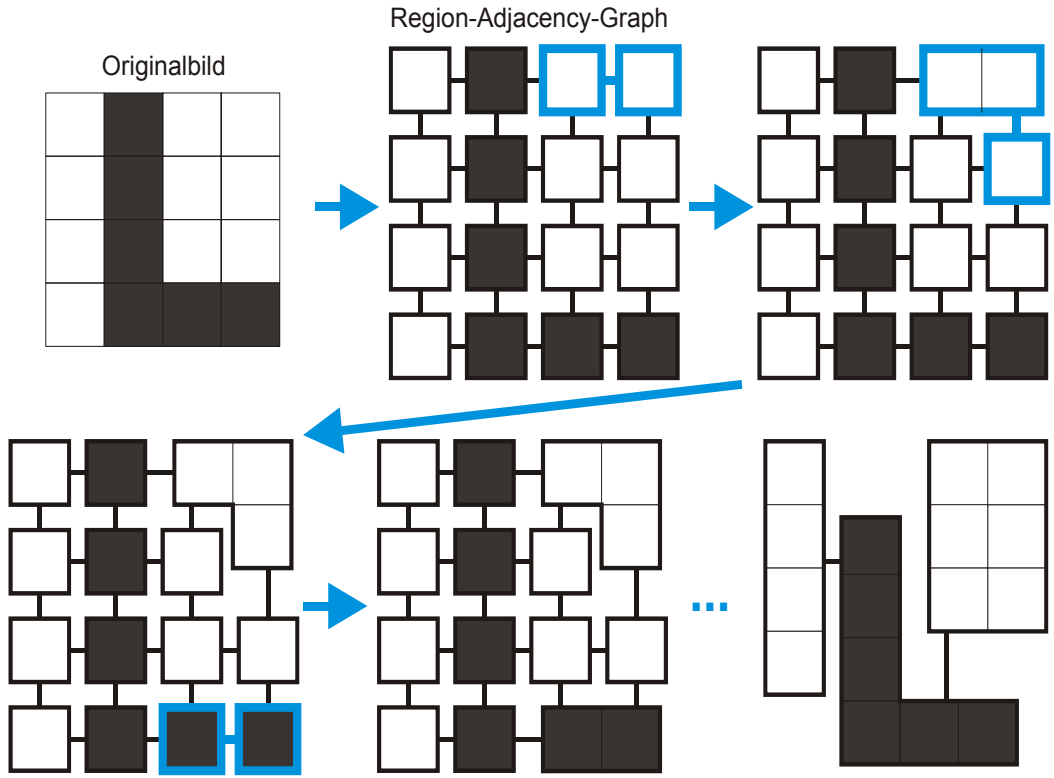
# Laplace-Pyramide

- Jede Skalierungsstufe s enthält nur den Unterschied $f_s$-expand(reduce($f_s$))
- Redundanzfreie Repräsentation

expand(reduce(image))-image

# Region Merging

- Initial wird jedes Pixel zu einem Segment erklärt.
- Zwei benachbarte Regionen werden zusammengefasst, wenn sie auch gemeinsam das Homogenitätskriterium erfüllen.
- Die Segmentierung ist beendet, wenn keine zwei Regionen mehr existieren, die zusammengefasst werden können.
- Zwischenergebnisse werden in einem Region Adjacency Graph (RAG) gespeichert.



# Region Merging

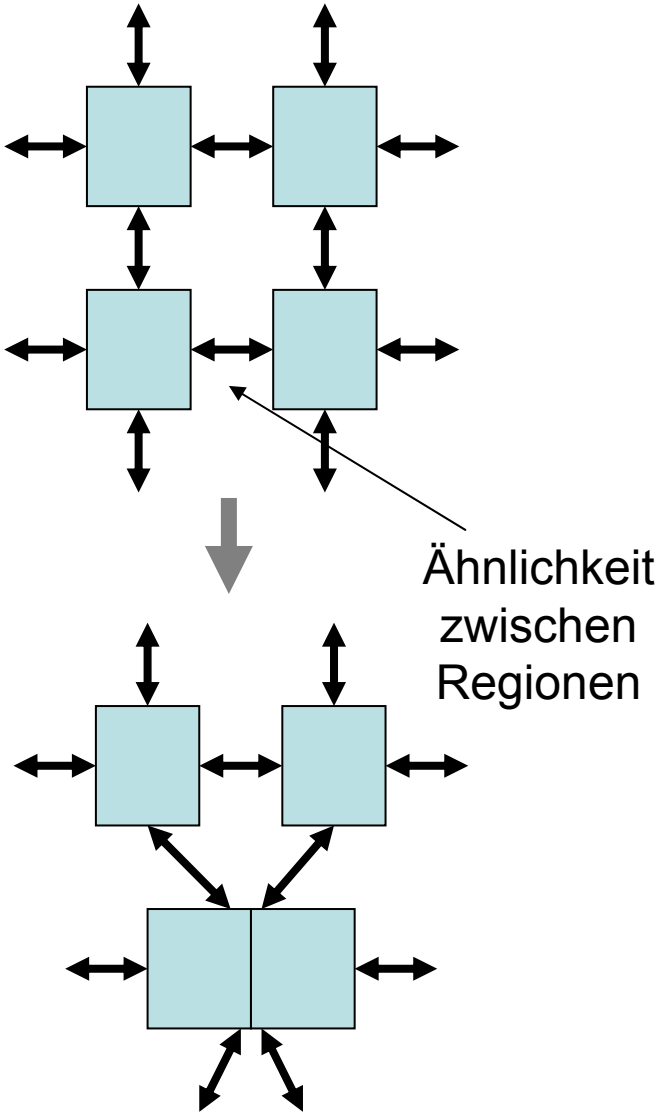
„von Pixeln zu Regionen“:

```
stopMerge = false
while not stopMerge do
  (r1,r2) =
  MaxSimilarity(region)
  if sim(r1,r2)>T then
   region.merge(r1,r2)
  else
   stopMerge=true
```

Bsp. f. Ähnlichkeitskriterium:

maximaler Grauwertunterschied zwischen Pixeln von r1 und r2.

Region Labeling kann in den Prozess integriert werden.



# Region Merging

- Homogenitätskriterium
  - Grauwertdifferenz innerhalb der Region.
  - Wahrscheinlichkeit, dass beide Regionen die gleichen Grauwertverteilungen haben.

# Region Merging und Multiskalenstrategie

Modellannahme:

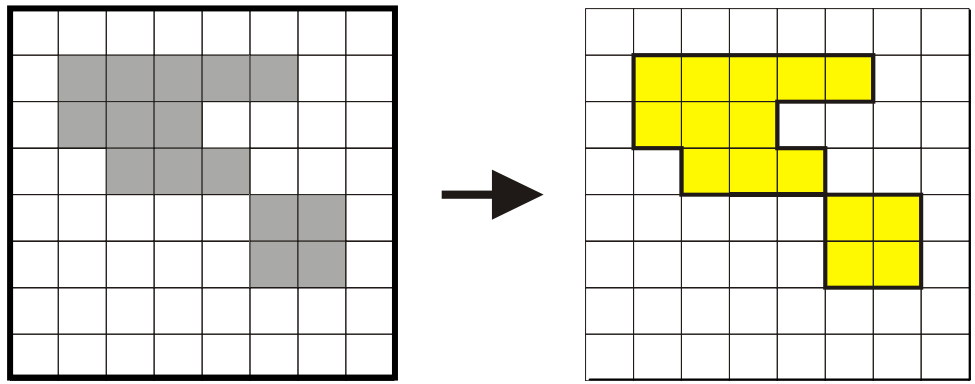
Die gröbste Skalierungsstufe, auf der sich segmentierungs-relevante Eigenschaften manifestieren, ist bekannt.

Prozess:

- Region Merging auf grober Skalierung
- Übertragung des Resultats auf die nächstfeinere Stufe.
- Alle Pixel, die zu Pixeln eines anderen Segments benachbart sind, werden nochmals geprüft.
- Verfahren endet, wenn die die feinste Skalierungsstufe erreicht ist

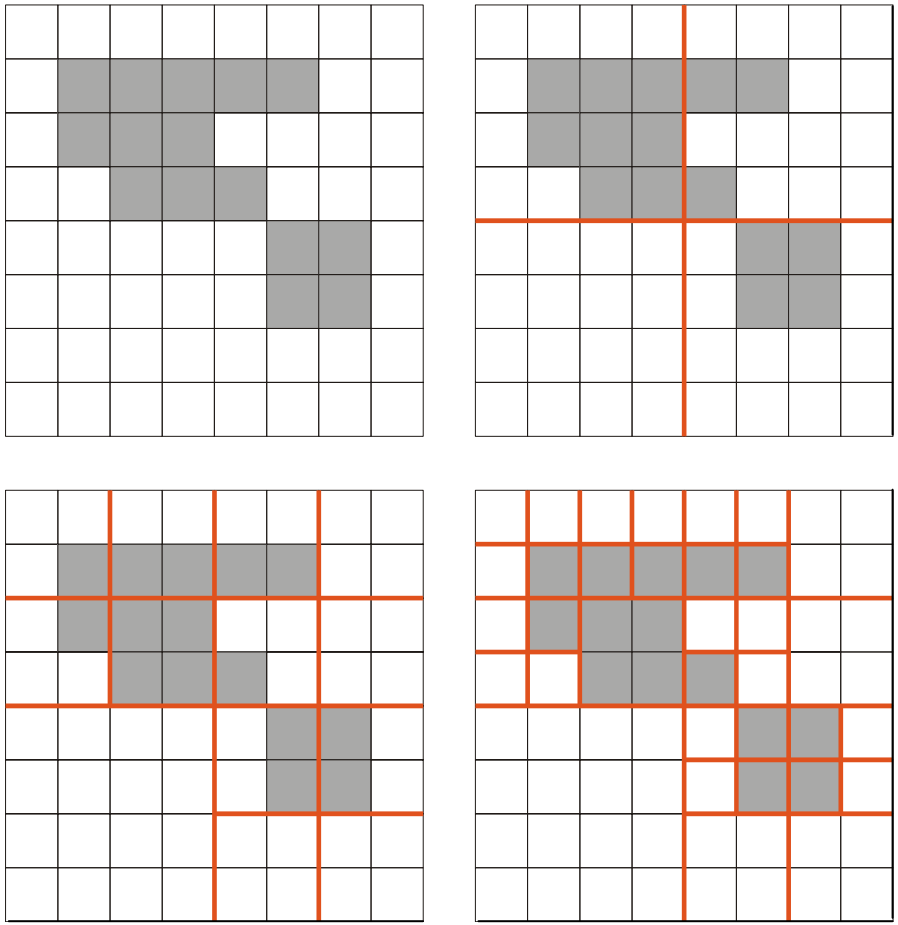
# Split & Merge-Algorithmus

Regionenbasiertes Verfahren



- Startbedingung: Das gesamte Bild ist ein Segment.
- Ein Segment wird solange in 4 Untersegmente zerlegt, wie es ein gegebenes Homogenitätskriterium nicht erfüllt
- Benachbarte Segmente werden zusammenfasst, wenn sie auch nach der Zusammenfassung das das Homogenitätskriterium erfüllen.
- Resultat ist eine vollständige, überdeckungsfreie Zerlegung des Bildes (Segmentierung gemäß Definition)

# Zerlegungsschritt

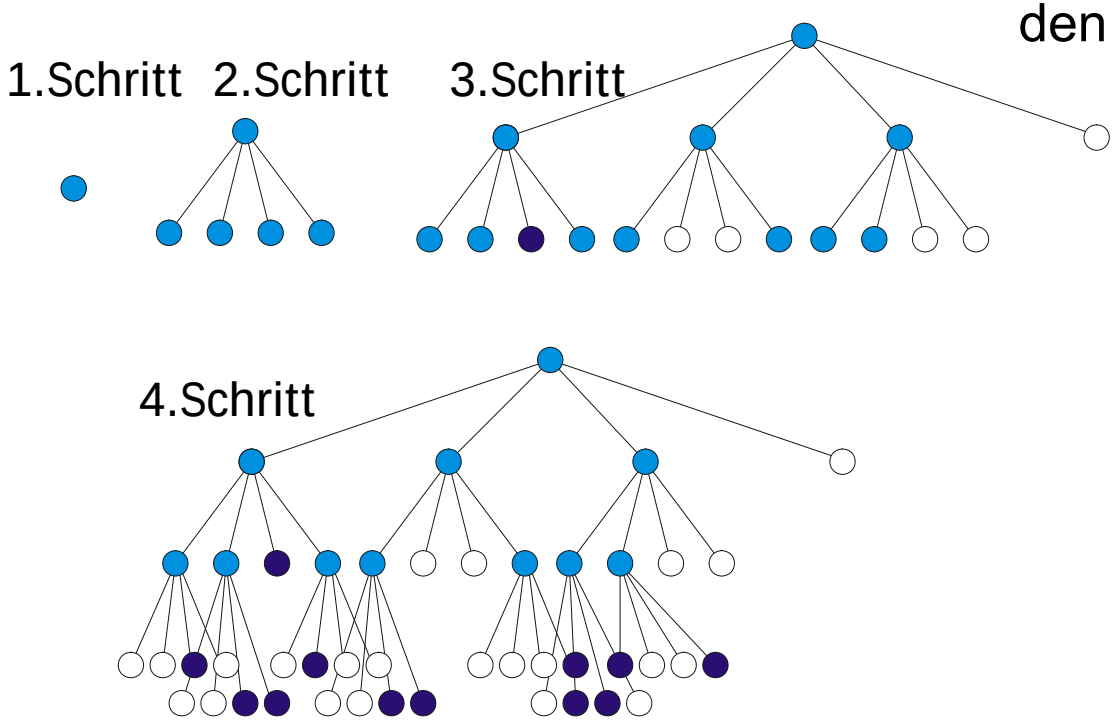


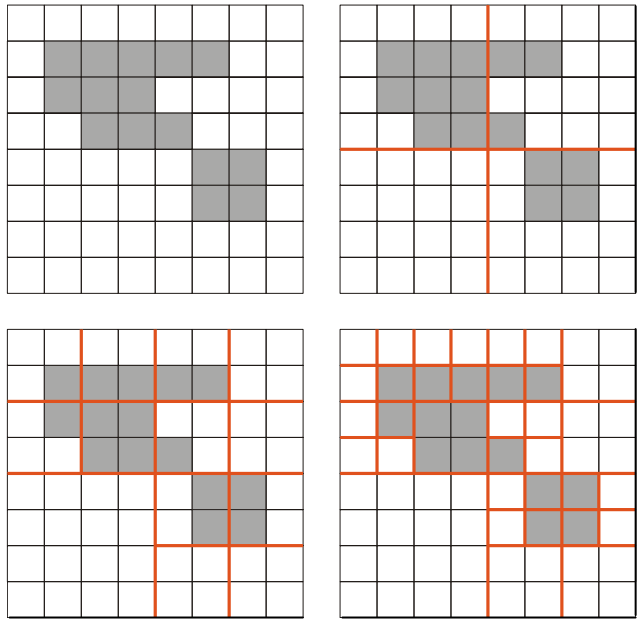
Zerlegung terminiert spätestens auf Pixelebene.

Problem: Datenstruktur zur Dokumentation der aktuellen Zerlegung

# Zerlegungsschritt (Quad-Tree Repräsentation)

Wert des Homogenitätsmerkmals einer Region wird im entsprechenden Blatt des Quad-Tree abgelegt

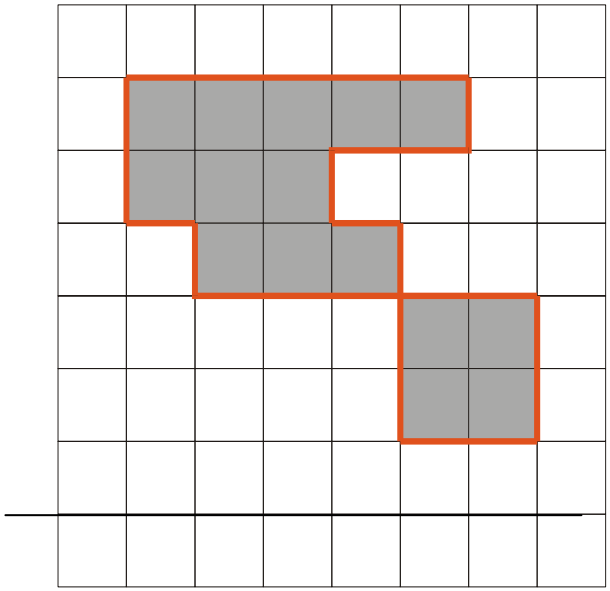


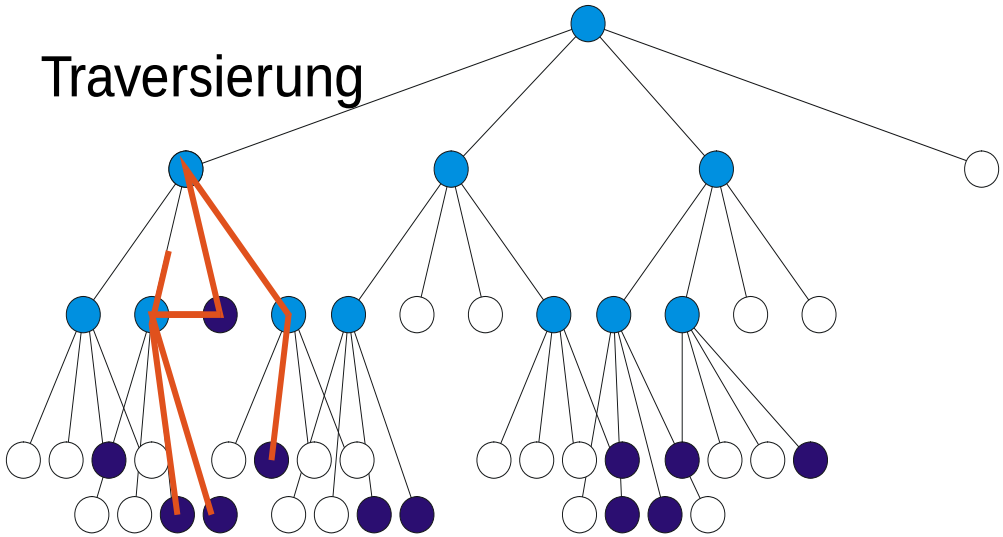



# Merging

Quadtree wird traversiert und in einen RAG überführt.

Auf dem RAG wird ein Region Merging durchgeführt.





# Split & Merge

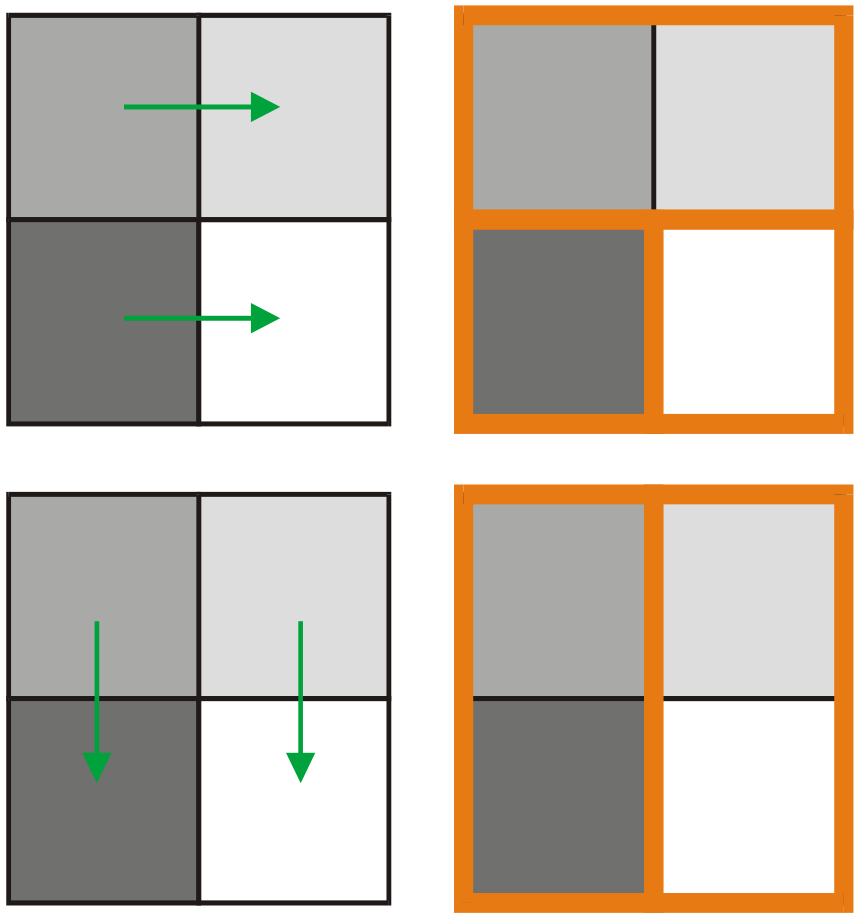
Resultat: Zerlegung des gesamten Bilds in Regionen.

Multiskalenstrategie ist integriert

Homogenitätsmerkmale wie bei Region Merging

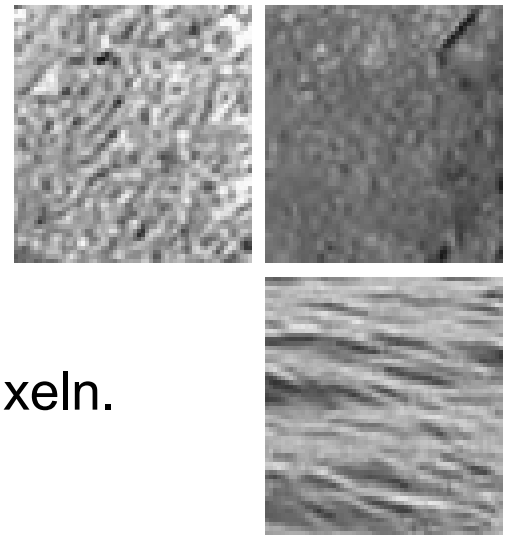
Probleme (Region Merging und Split & Merge):

- Merge-Schritt ist bei relativem Homogenitätsmerkmal nicht immer eindeutig
- Minimale Segmentzahl wird nicht immer gefunden

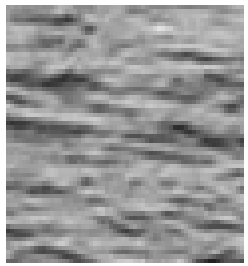


# Textur als Homogenitätsmerkmal

- Textur – Musterung der Oberfläche
  - Es existiert keine Definition von Textur
  - Es gibt eine große Anzahl von Texturmaßen
  - Textur ist inhärent skalenabhängig
  - Textur ist eine Eigenschaft einer Gruppe von Pixeln.



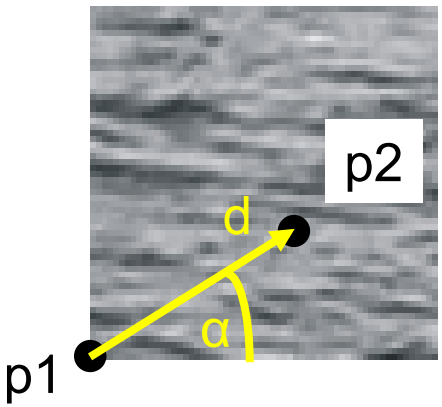
- Texturmaße
  - strukturell (Zusammensetzung aus Texturelementen – texel)
  - stochastisch (eine charakterisierbare Grauwertverteilung)
  - spektral (charakteristische Frequenzattribute)

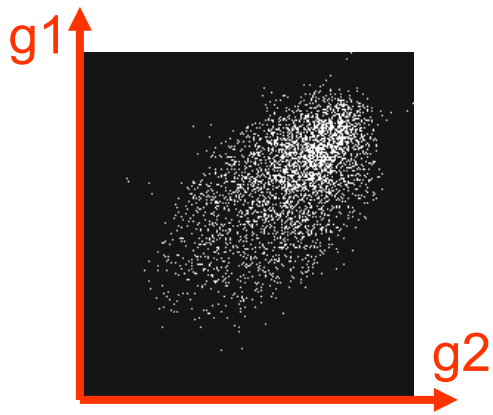


# Haralick´sche Texturmaße

- **Co-Occurrence-Matrix =** Zweidim. Histogramm für Pixelpaare.

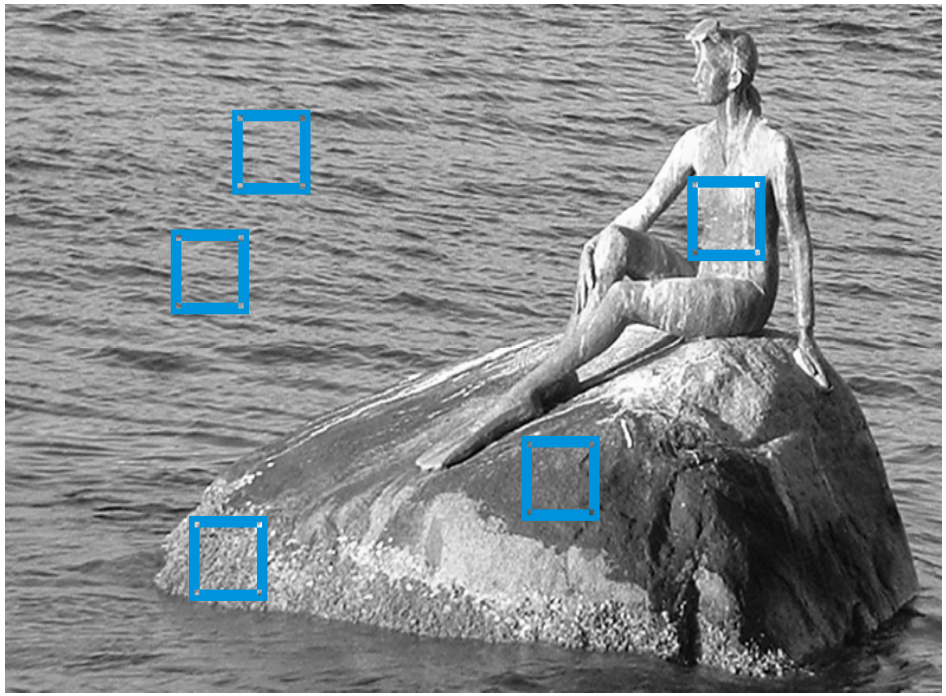
  Pixel p1 und p2 sind ein Paar, wenn sie Abstand *d* haben und auf einer Linie mit einem gegebenen Winkel $\alpha$ zur *x*-Achse liegen.
- Co-Occurence-Matrix repräsentiert die Korrelation zwischen Pixeln. (Wahrscheinlichkeit P, dass p1 und p2 Grauwerte g1 und g2 haben)
- Meist sind Pixel nicht über große Entfernungen korreliert, daher sind Werte *d*=1, *d*=2 üblich.
- Falls Korrelation über größere Entfernung vermutet wird, dann sollte eine Multiskalenstrategie angewendet werden.

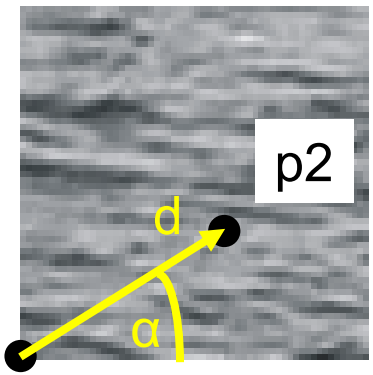


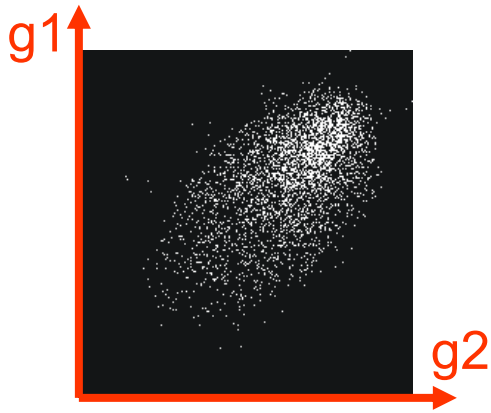



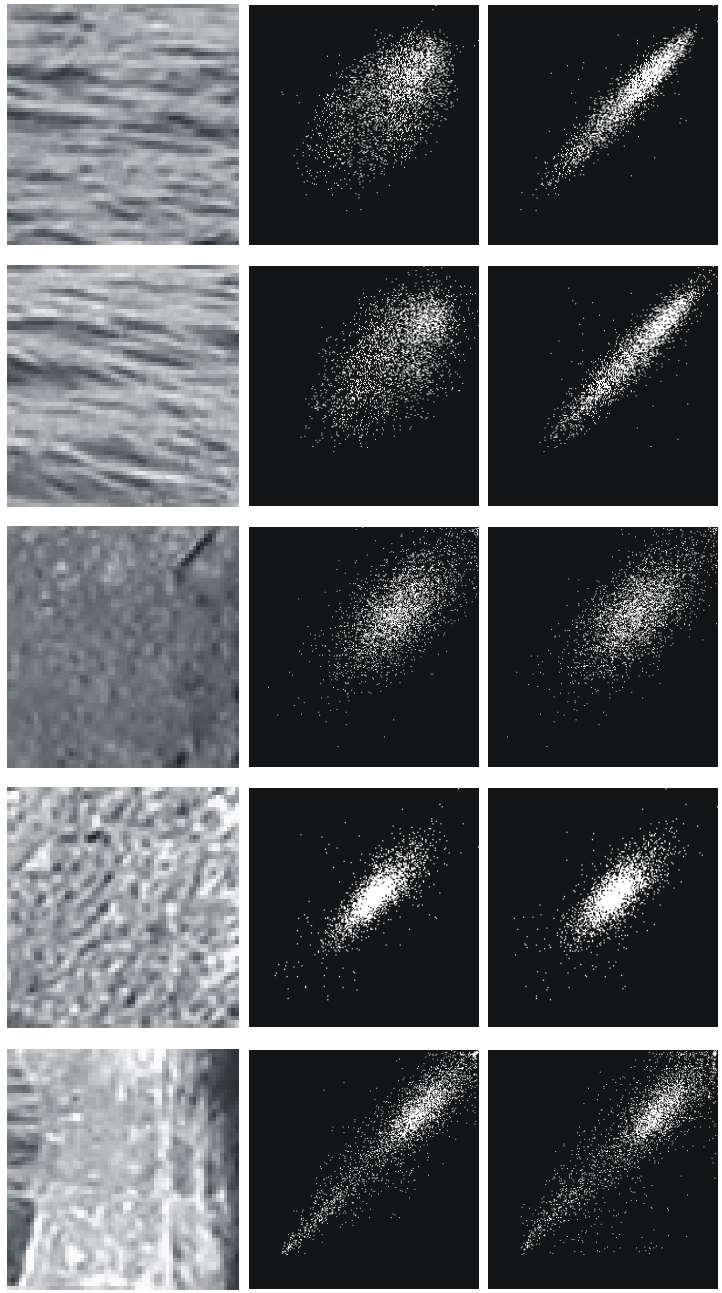

Robert Haralick

# Co-Occurrence-Matrix





d=1, α=90°, α=0°

# Haralick‘sche Texturmaße

$$\sum_{g_1=0}^{K-1}\sum_{g_2=0}^{K-1} P_{\Delta,\alpha}^2(g_1,g_2)$$ Energie

$$\sum_{g_1=0}^{K-1}\sum_{g_2=0}^{K-1} (g_1-g_2)^2 \cdot P_{\Delta,\alpha}(g_1,g_2)$$ Kontrast

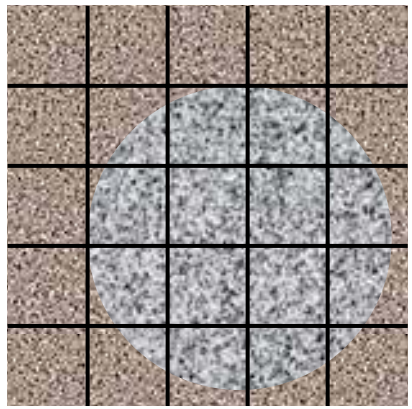
$$\sum_{g_1=0}^{K-1}\sum_{g_2=0}^{K-1} P_{\Delta,\alpha}(g_1,g_2)\cdot \log\left[P_{\Delta,\alpha}(g_1,g_2)\right]$$ Entropie

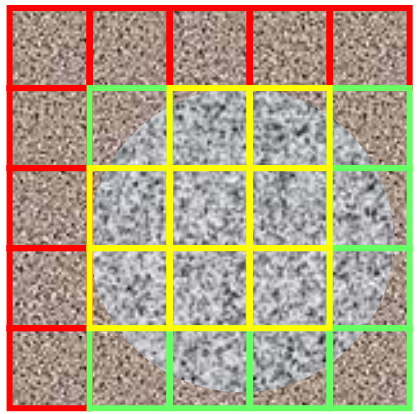
$$\sum_{g_1=0}^{K-1}\sum_{g_2=0}^{K-1} \frac{P_{\Delta,\alpha}(g_1,g_2)}{1+|g_1-g_2|}$$ Homogenität

- Lassen sich aus der Co-Occurrence-Matrix berechnen
- Liefern aussagekräftige Kennwerte für verschiedene Texturen

# Besonderheiten Texturmerkmal

- Berechnung der Texturmerkmale auf der Basis willkürlicher Regionen.
- Segmentierung
- Berechnung der Zuverlässigkeit
- Erneute Segmentierung





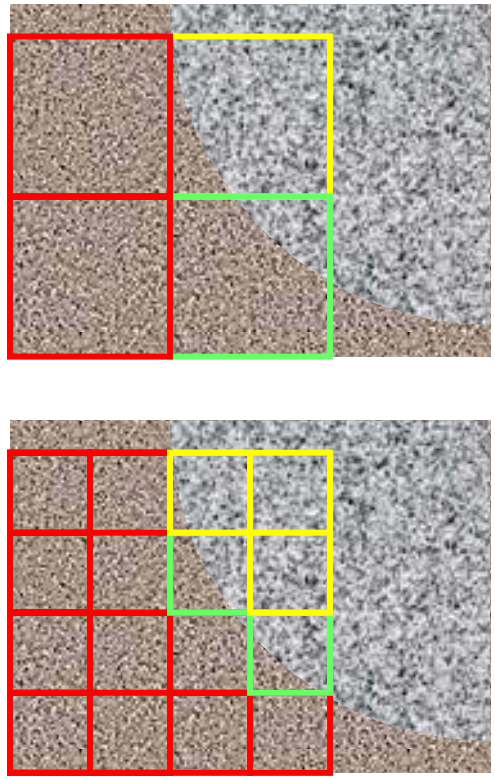
# Zuverlässigkeit Texturmerkmal

Persistenz:

Falls eine Region in Teile zerlegt wird und das Merkmal in den Teil-regionen berechnet werden kann,

dann sollte das Texturmerkmal in der Teilregion dieselben Werte annehmen, wie das der Ursprungsregion.

Achtung: In der Regel ist das berechnete Maß eine Schätzung, deren Güte von der Regionengröße abhängt.

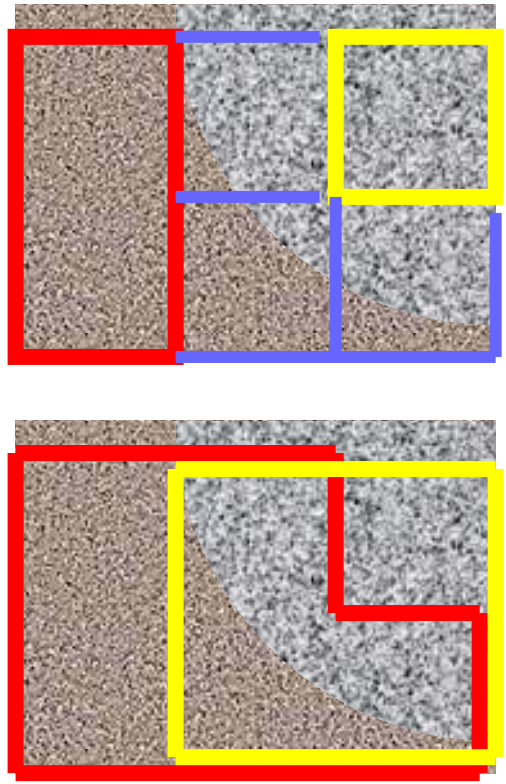


# Strategie: Erneute Segmentierung

- Texturmerkmale der Gesamtsegmente erneut berechnen
- Segmente mit hoher Unzuverlässigkeit
  - allen benachbarten zuverlässigen Segmenten zuordnen
  - ähnlichste Kombination selektieren.

Berechenbarkeit von Texturmerkmalen beliebiger Regionen ist vorteilhaft.

Kann über mehrere Auflösungsstufen erfolgen.



Die beiden Extremfälle

# Naive Textursegmentierung

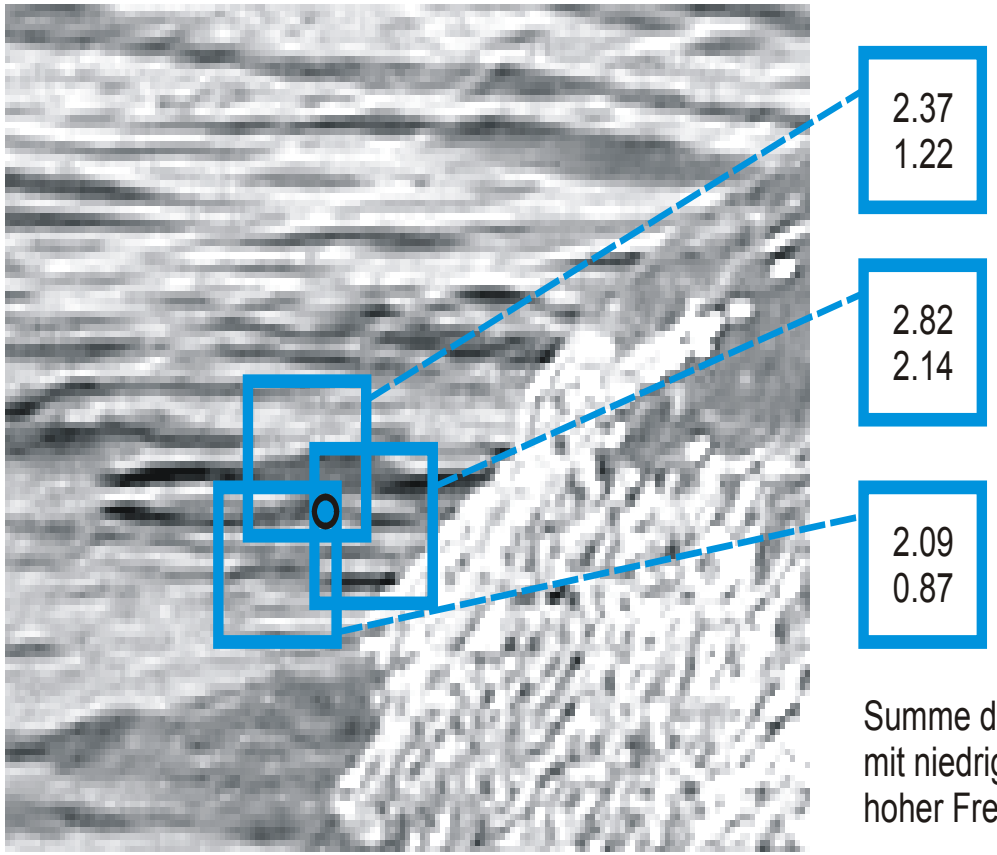
Modell:

- Skalierungsstufe der Textur ist bekannt.
- Blockgröße für die Texturberechnung ist bekannt.
- Geeignetes Texturmaß ist bekannt

Algorithmenskizze:

- Berechne für jedes Pixel das Texturmaß.
- Führe eine homogenitätsbasierte Segmentierung auf den Texturmaßen durch.
- Jedes Pixel erhält für jedes Label, dessen Block das Pixel überdeckt, eine von Null verschiedene Wahrscheinlichkeit.
- Nachverarbeitung ähnlich wie bei Schwellenwertsegmentierung.

# Textursegmentierung (naiv)



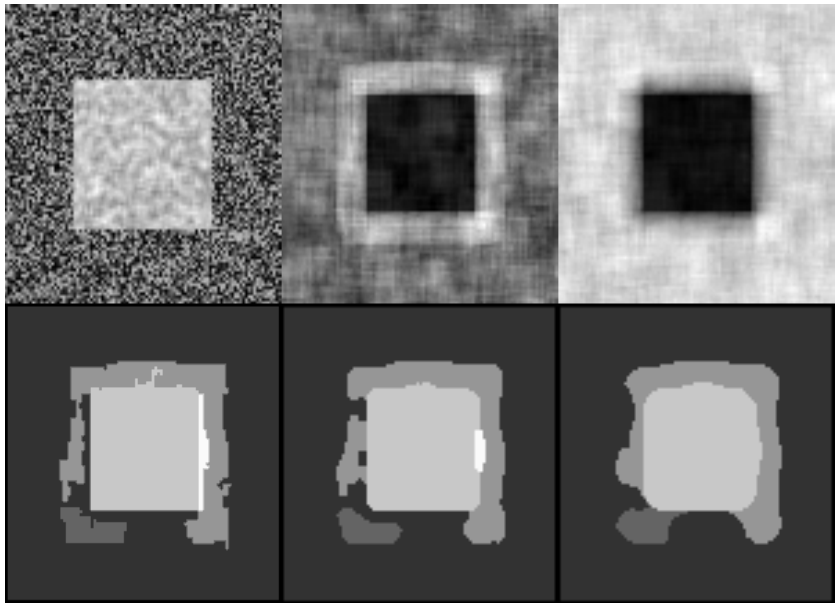

Beispiel:

Jedes Pixel wird von mehreren Blöcken überdeckt.

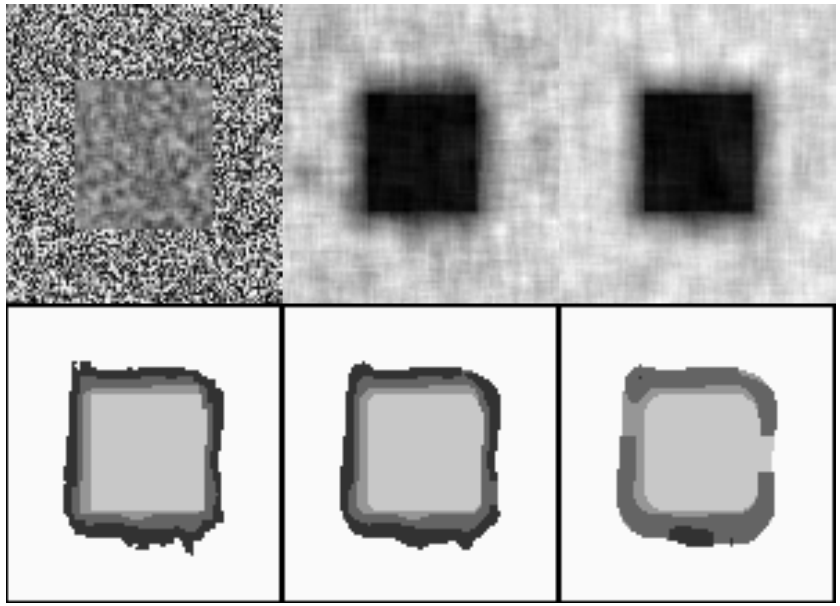
Sind die Texturen unterschiedlich, dann erhalten diese Blöcke unterschiedliche Label.

# Nachverarbeitung Medianfilterung

Segmentierung von zwei verschiedenen Texturen

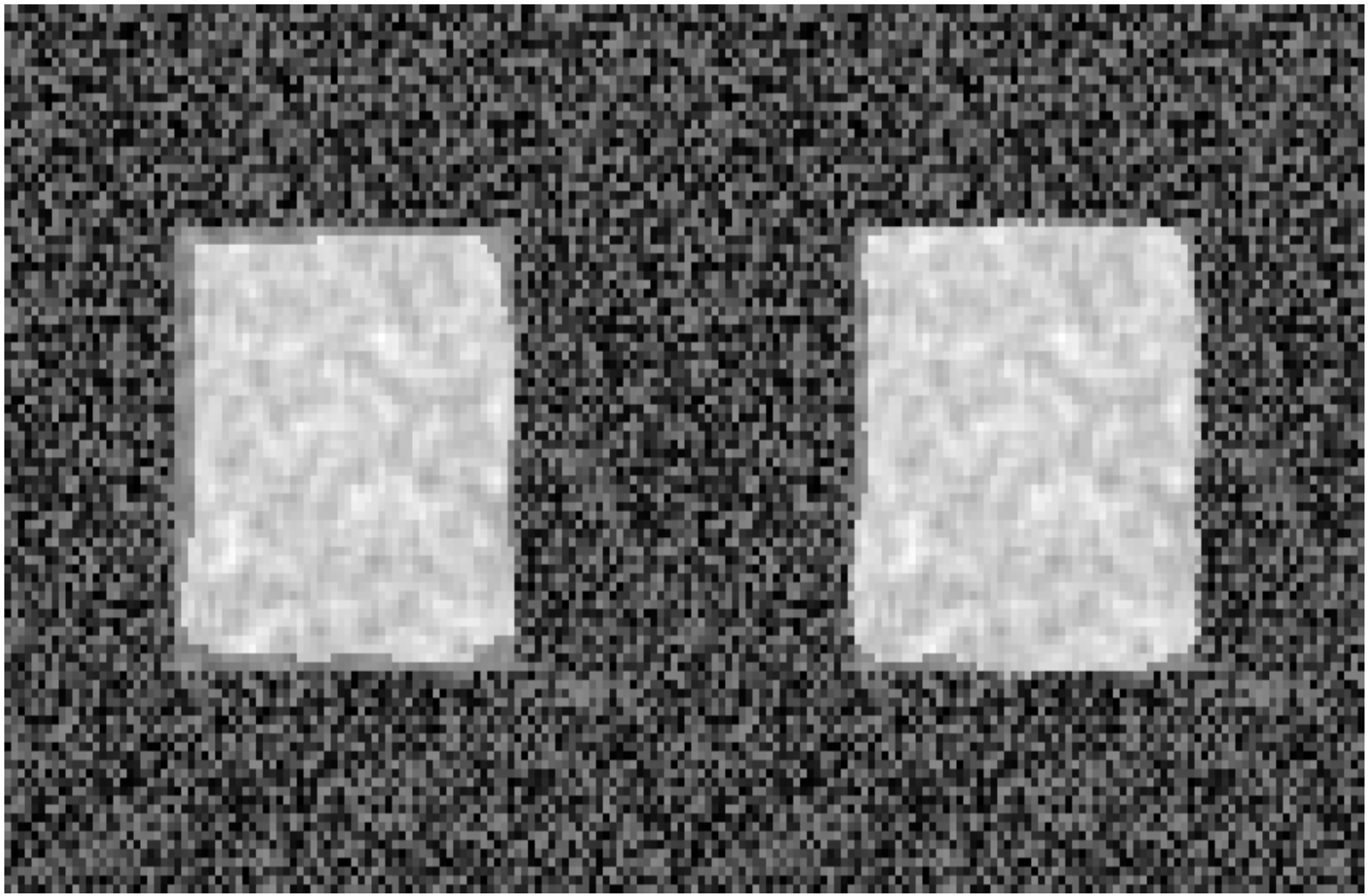


Resultat Median 5×5 Median 11×11



Resultat Median 5×5 Median 11×11

# Nachverarbeitung Relaxation Labeling



# …Also nochmal bis hierhin…

- Multiskalenstrategie
- Regionenbasierte Segmentierung (Region Merging, Split & Merge)
- Datenstrukturen RAG und Quad-Tree
- Texturen und Textursegmentierung

# Kantenbasierte Segmentierung

- Edge Linking und Canny Edge Operator
- Nulldurchgänge
- Wasserscheidentransformation

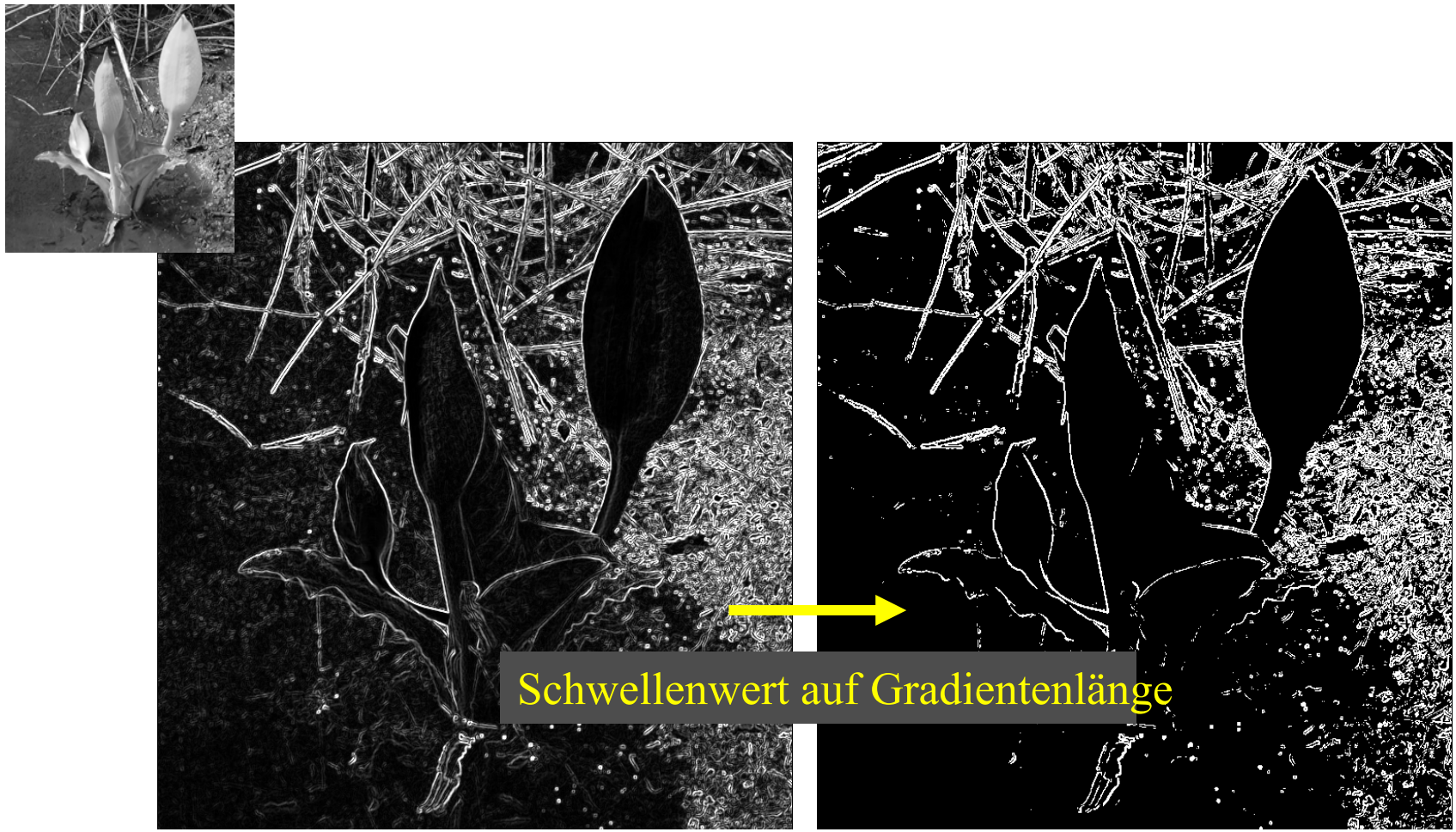
# Segmentierung durch Kantenerkennung


**Vorteil**: Kantenmerkmale sind robuster gegenüber Shading

Einfache Methode:

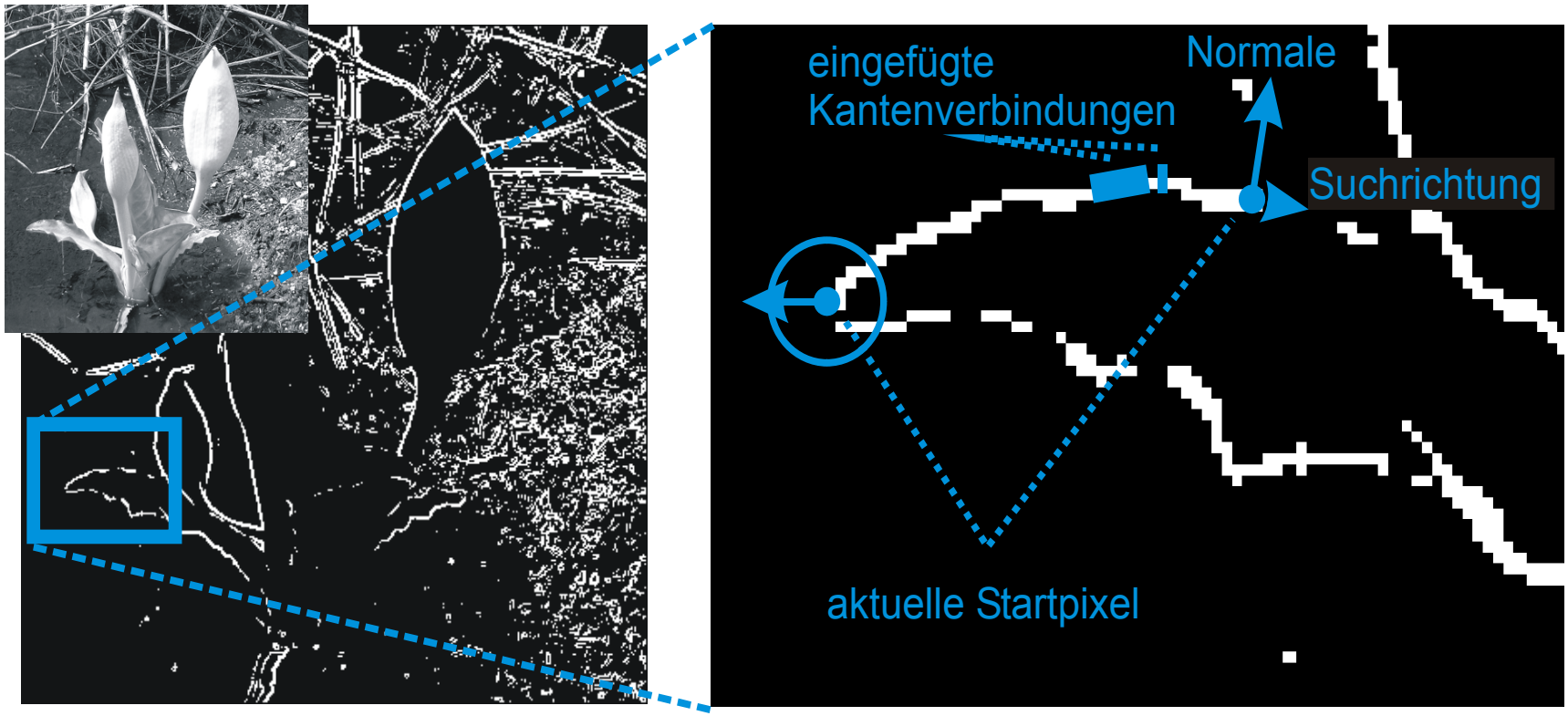
- Gradientenberechnung
- Kantenpunktdetektion (z.B. Schwelle auf Gradientenlänge)
- Region Labeling basierend auf Kantenpunkten.

# Segmentierung durch Kantenerkennung




**Problem**: Kantenpunkte sind nicht Ränder zusammenhängender Gebiete.

# Edge Linking




Anfangs sind alle Kantenpixel frei und nicht untersucht.

Edge Linking sucht sich das nächste nicht untersuchte und freie Kantenpixel und versucht es mit anderen Pixeln zu einem Kantenzug zu verknüpfen.

# Edge Linking

1. Suche das nächste Kantenpixel, welches noch nicht als „untersucht“ markiert wurde und erkläre es zum Startpixel eines Kantenzugs.
2. Falls sich in der Umgebung des Kantenpixels in einer der beiden Richtungen orthogonal zur Kantenrichtung unmarkierte Kantenpixel befinden, die eine ähnliche Gradientenrichtung und –stärke aufweisen:
   a. Markiere die Pixel als zum selben Kantenzug gehörend.
   b. Erkläre diese Pixel zu neuen Startpixeln.
   c. Gehe zu 2.
3. Falls sich in der Umgebung markierte Pixel befinden, die den obigen Bedingungen genügen, dann wurde eine Verzweigung von Kanten gefunden.
4. Falls ein Kantenpixel gefunden wurde, gehe zurück zu Schritt 1.

# Canny Edge Operator

Ziele:

- möglichst viele Kanten fehlerfrei vom Hintergrund unterscheiden zu können (niedrige Rate von Fehldetektionen).
- (unverzweigte) Kanten genau zu lokalisieren.
- für jede Kante genau eine Detektorantwort zu liefern.

Canny Operator besteht aus Kantenhervorhebung und Erzeugung von Kantenzügen.

Optimale Kantenhervorhebung ist eine Filterung mit einer 1-D abgeleiteten Gaußfunktion orthogonal zur Kante.

Geringfügig schlechtere Ergebnisse erzielt man mit einem 2-D Gradientenoperator auf der Basis abgeleiteter Gaußfunktionen.

# Canny Edge Operator

- Es wird immer das Pixel mit größter Gradientenlänge selektiert.
- Startpixel können nur Pixel sein, deren Gradientenlänge oberhalb einer Signifikanzschwelle T1 liegt.
- Neue Kantenpixel werden in den Kantenzug eingefügt, wenn ihre Gradientenlänge größer als T2 ist (T2<T1).
- Das Verfahren endet, wenn keine neuen Startpixel gefunden werden.

# Nulldurchgänge

Die Orte der Nulldurchgänge der zweiten Ableitung sind Ränder zusammenhängender Gebiete.

Methode:

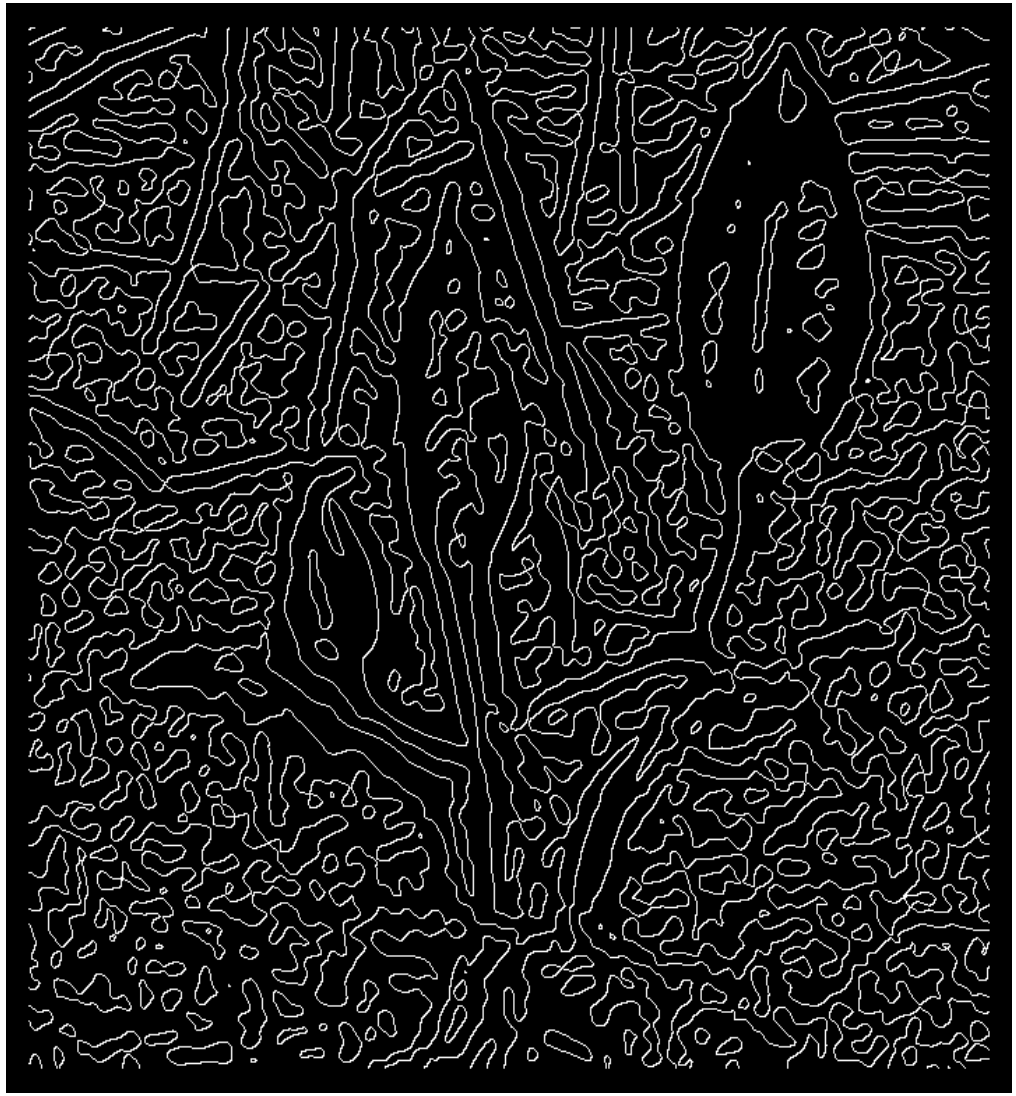
- Laplace-Operator
- Nulldurchgänge bestimmen:

  $\nabla^2(f(i,j)) \cdot \nabla^2(\text{shift}(f(i,j))) \leq 0$

  (shift: Verschiebung des Bilds um ein Pixel in jede Richtung)

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} + \frac{\partial^2 f(x,y)}{\partial x \partial y} + \frac{\partial^2 f(x,y)}{\partial y \partial x} : \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

# Nulldurchgänge



Kombination des Laplace-Operators mit Glättungsoperator (z.B. als LoG-Operator) reduziert die Anzahl der Nulldurchgänge

# Wasserscheidentransformation

- Wasserscheide: Menge aller Orte, die die Grenzen der Entwässerung in unterschiedliche Senken sind.
- Beispiel: Wasserscheide zwischen Nordsee und Mittelmeer verläuft entlang des Kamms der Berner Alpen.
- Wasserscheide in der Segmentierung: Generiere ein Höhenprofil so, dass die Wasserscheiden gerade die gesuchten Segmentgrenzen sind.

# Wasserscheiden

- Wasserscheiden sollen an Kanten verlaufen.
- Wasserscheiden sind „Gebirgskämme“

► Wasserscheiden sind die Längen der Grauwertgradienten.

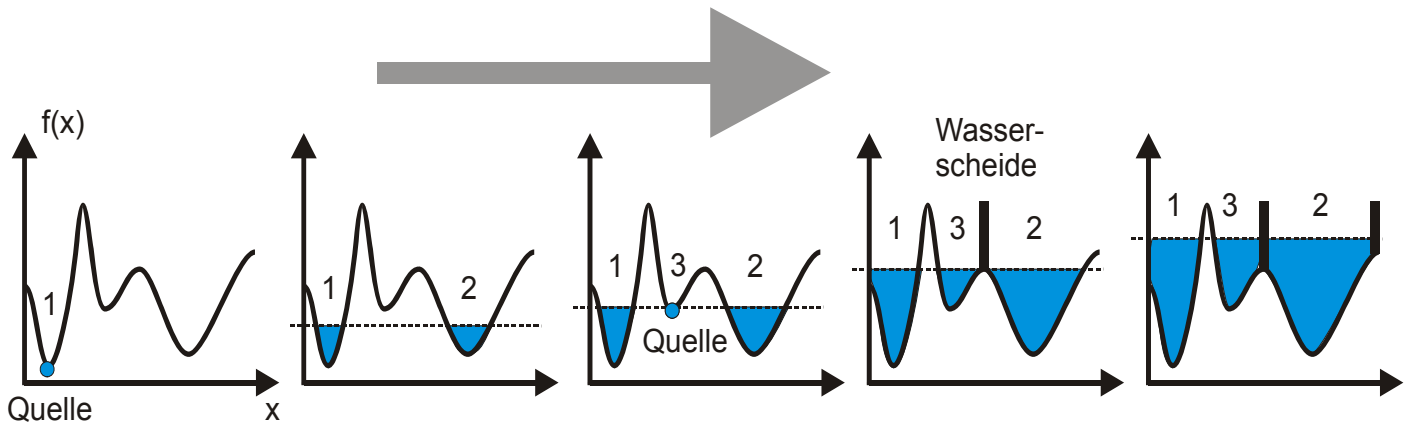
# Wasserscheidentransformation

- Beregnung

  Es fällt „Regen“ auf jedes Pixel. Anhand des Gradienten wird entschieden, wohin der Regen entwässert wird.

- Flutung

  Die „Welt“ wird von den Senken her geflutet. Immer wenn Wasser aus zwei Senken zusammen fließt, entsteht eine Wasserscheide.



# Flutungsalgorithmus (Skizze)

Jedes neu überflutete Pixel ($m_f$,$n_f$) ist

- **in Isolation**:

  Es nicht zu anderen überfluteten Pixeln der Höhen $h<h_{aktuell}$ benachbart.

  Isolierte Pixel sind Kerne von neuen Segmenten.

- **Erweiterung**:

  Es ist zu anderen überfluteten Pixeln der Höhen $h<h_{aktuell}$ mit gleichem Label benachbart.

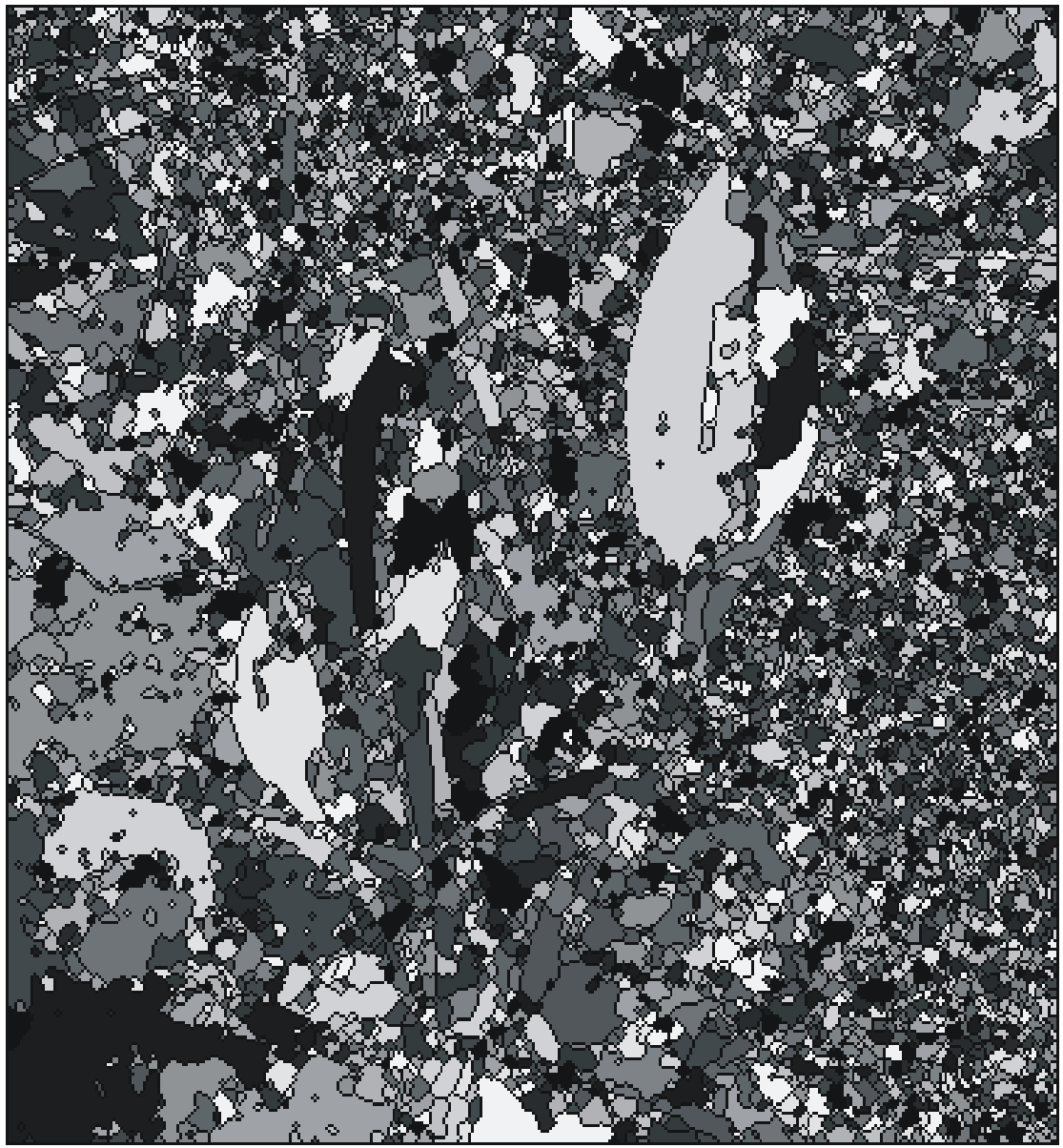
  Das Pixel wird dem Segment mit diesem Label zugeordnet.

- **Wasserscheide**:

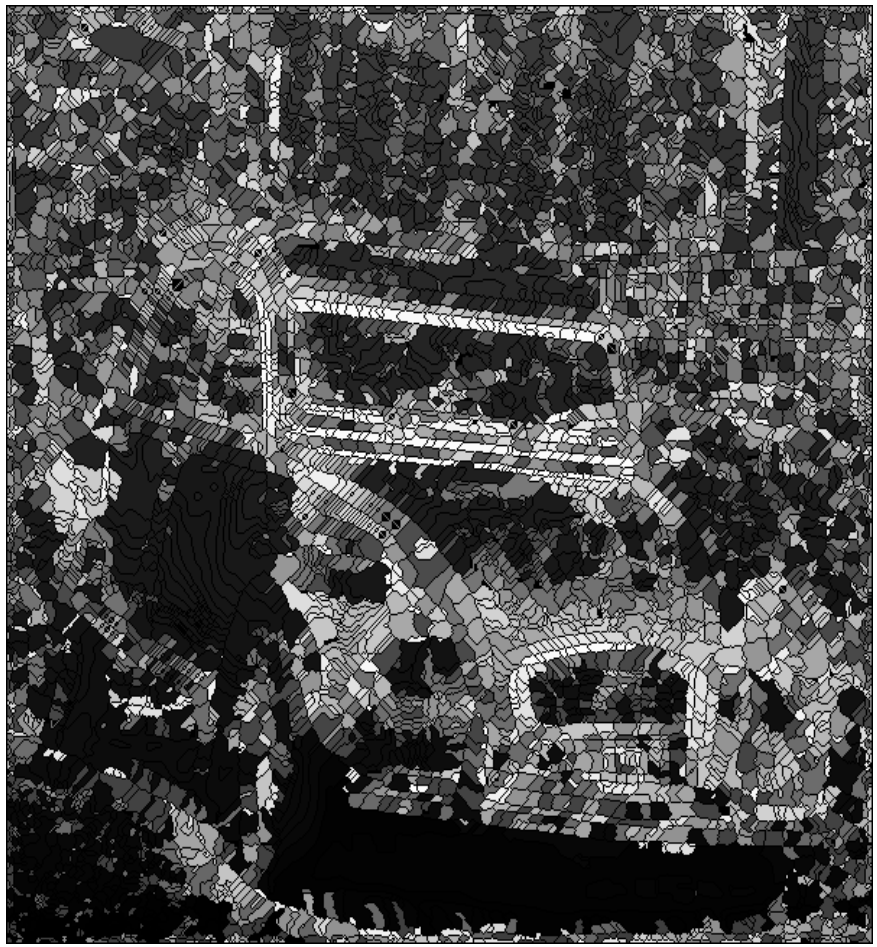
  Es ist zu überfluteten Pixeln von mindestens zwei Regionen benachbart.

  Dem Pixel wird das Label „Wasserscheide“ zugeordnet.

# Resultat der WST

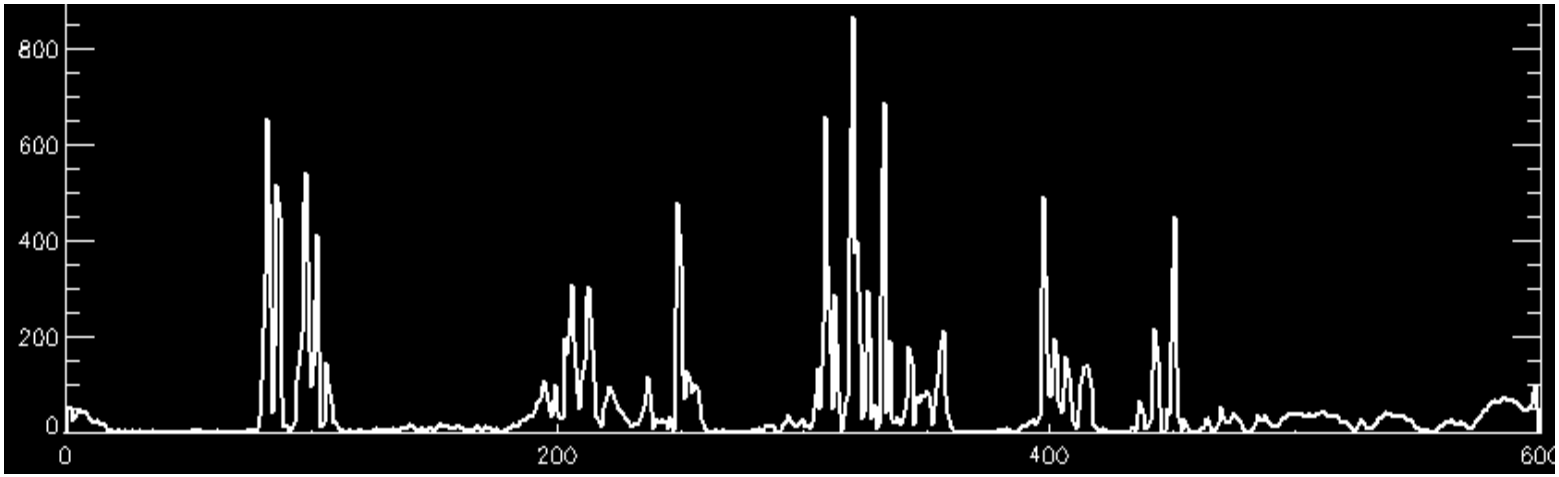


# Resultat



# Problem Übersegmentierung

- Für die WST ist jedes lokale Minimum eine Senke.
- Die meisten Senken werden durch Rauschen verursacht.
- Senken durch Rauschen sind weniger tief als die von Kanten.

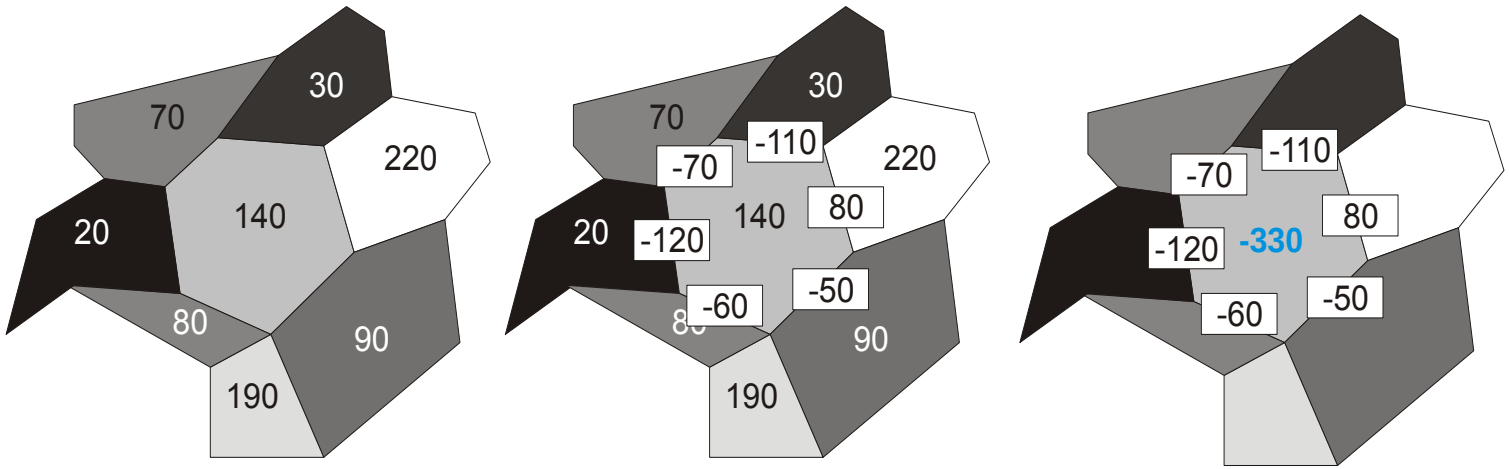


# Hierarchische WST

Multiskalenstrategie:

- Wasserscheidentransformation auf dem WST-Resultat.
- Jede Region erhält ihren durchschnittlichen Grauwert als Funktionswert.
- Die erste WST wird hauptsächlich durch Rauschen verursachte Senken finden.
- „Wahre“ Senken sollten über mehrere Stufen der Hierarchie erhalten bleiben.

# Gradienten für die hWST




- Zu jeder der benachbarten Regionen wird die Differenz berechnet.
- Die Länge des Gradienten ist die durchschnittliche Differenz zu allen Regionen.
- Die Richtung ergibt sich aus den (mit der Regionengröße gewichteten) Vektoren zwischen dem Schwerpunkt der Region und den Schwerpunkten aller benachb. Regionen.

# …nochmal zum langsam mitmeißeln…

- Edge Linking und Canny Edge Operator
- Nulldurchgänge zur Regionensegmentierung
- Wasserscheidentransformation
  - Ausführung
  - Probleme
  - Multiskalenstrategie

# Literatur

- Klaus D. Tönnies: "Grundlagen der Bildverarbeitung", ISBN 3-8273-7155-4
- http://www.pearson-studium.de/main/main.asp?page=booksites/selectchapter&isbn=3827371554&PSZielgruppe=Student

- Vorlesung am 23.12. fällt aus
  - Stoff wurde und wird auf andere Vorlesungstermine und Übungen verteilt
- Nächste Vorlesung am 13.01.06

# Übungsaufgabe über Weihnachten



- Finde Transformation von z1 nach z2
- Erweitere Wasserscheidentransf. auf t < 0°C
  - Suche nach Punkten maximaler Höhe
  - Bewegung in umgekehrter Richtung des Gradienten