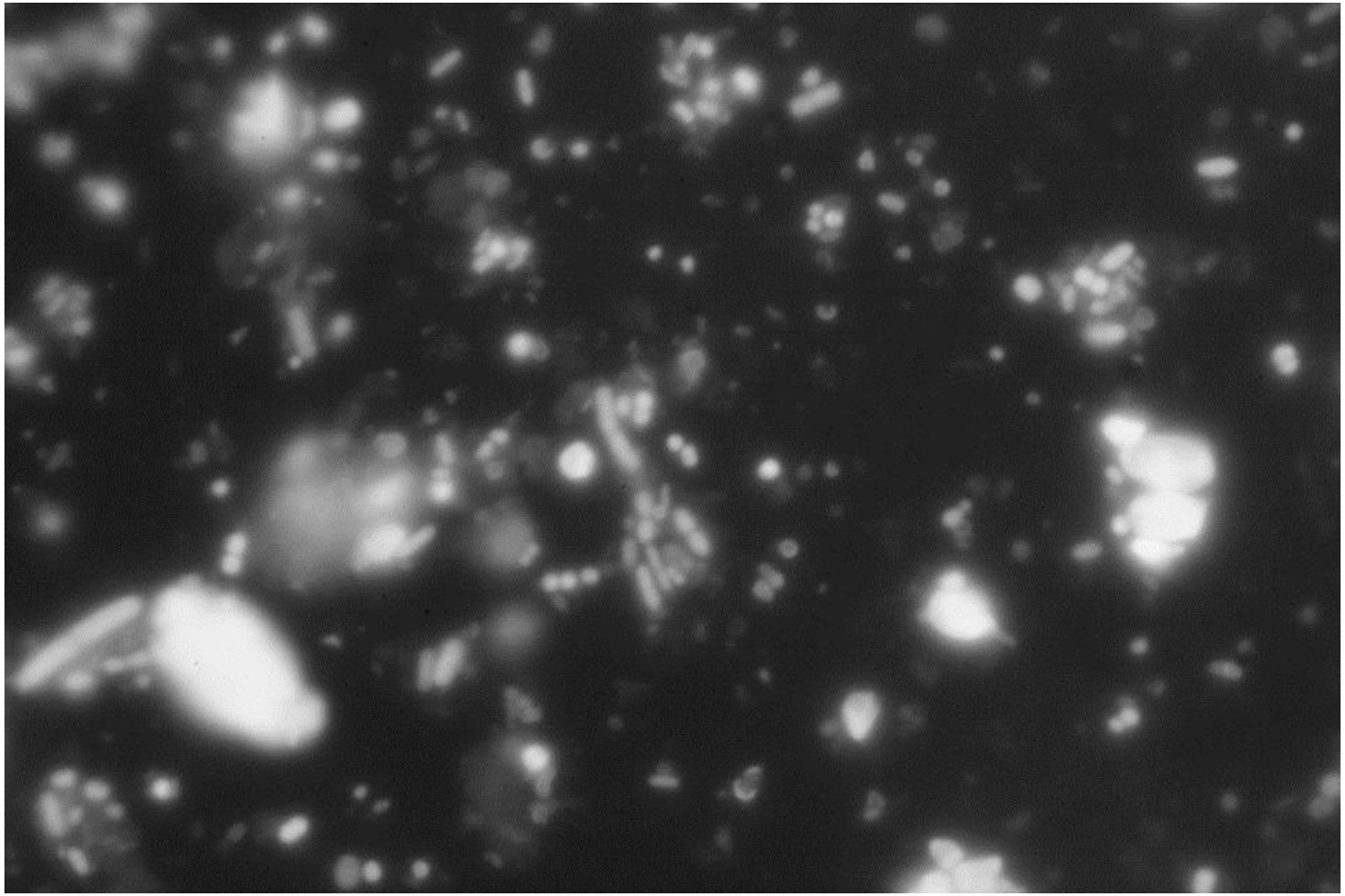
Famous Last Question

Wie könnte man diese Mikroorganismen segmentieren?



Kantenbasierte Segmentierung

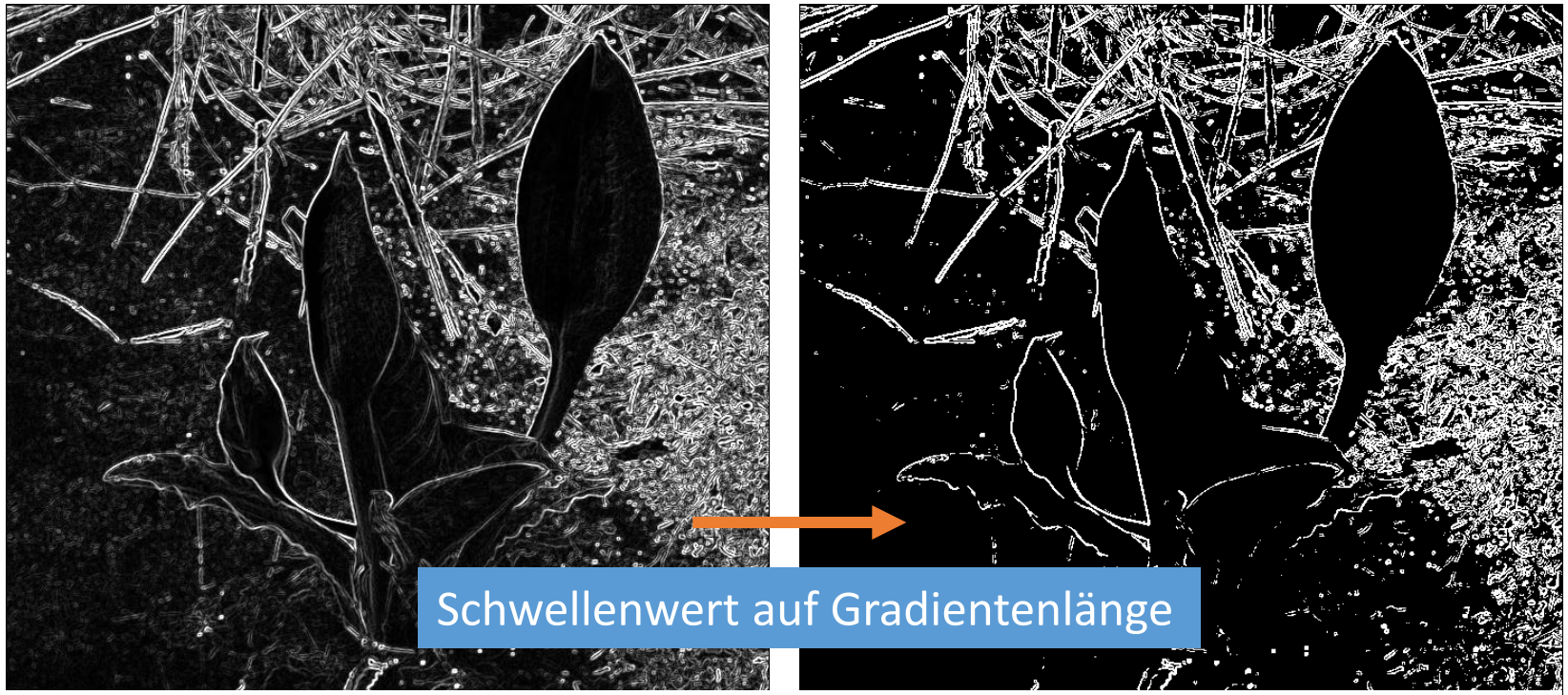
- Edge Linking und Canny Edge Operator
- Nulldurchgänge
- Wasserscheidentransformation
- Relaxation Labeling zur Nachverarbeitung

Segmentierung durch Kantenerkennung

- **Vorteil**: Kantenmerkmale sind robuster gegenüber Shading
- Einfache Methode
  - Gradientenberechnung
  - Kantenpunktdetektion (z.B. Schwelle auf Gradientenlänge)
  - Region Labeling basierend auf Kantenpunkten.

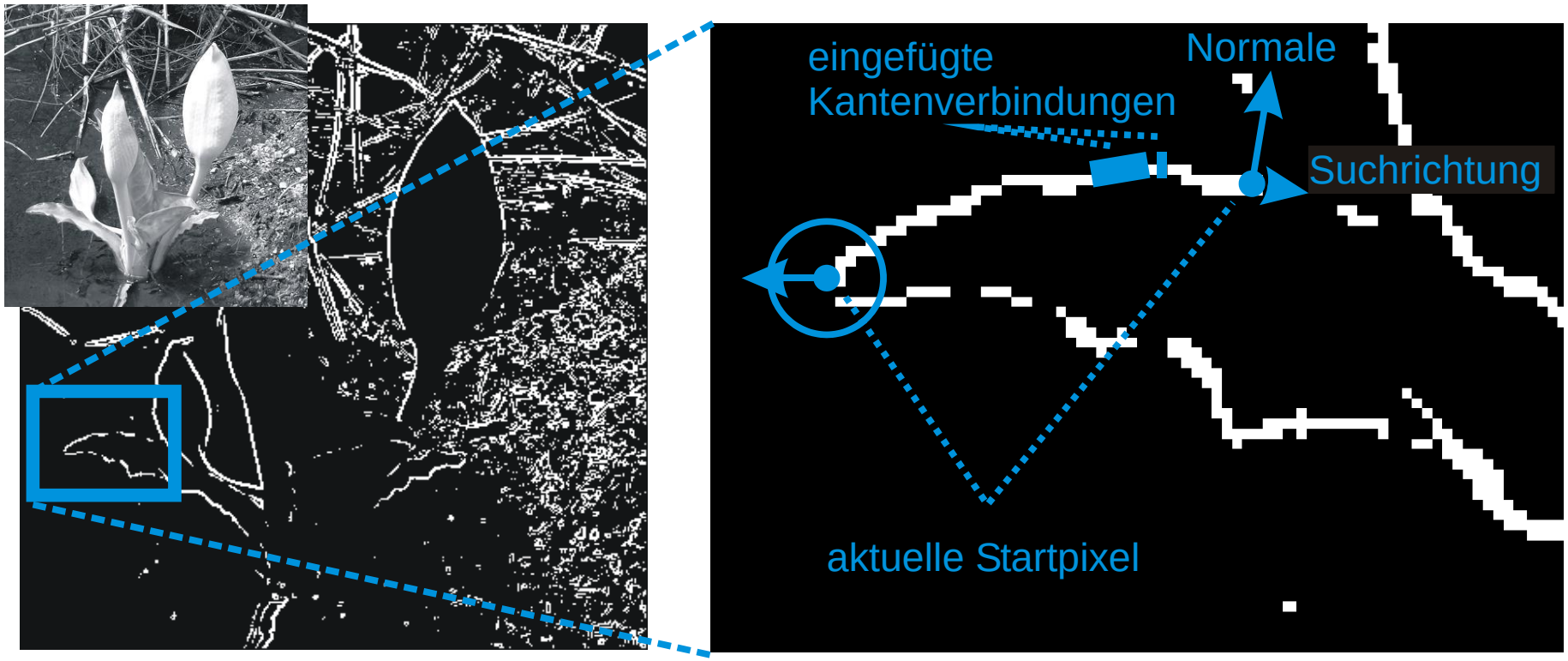
Länge des Gradienten

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 bereits 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 Kantenzug zugehörig.
   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 kein 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.

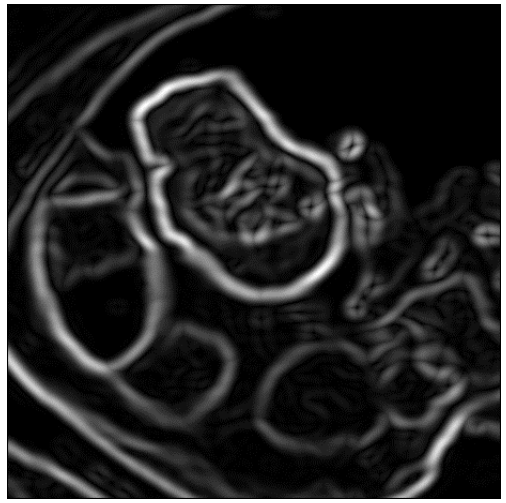
Canny Operator: *Kantenhervorhebung* und *Erzeugung von Kantenzügen*.

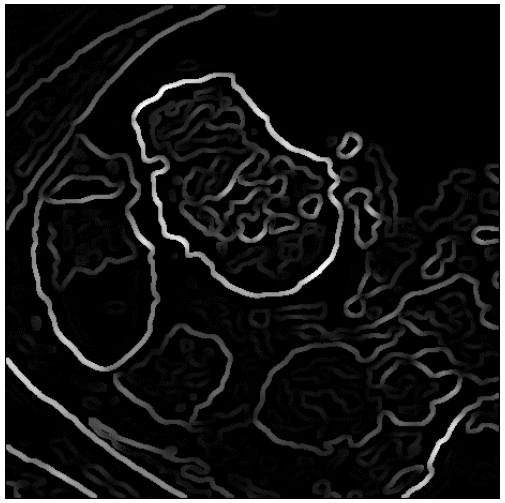
Optimale Kantenhervorhebung ist eine Filterung mit einer 1-D abgeleiteten Gaußfunktion orthogonal zur Kante.

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

Canny Edge Detection (Beginn)

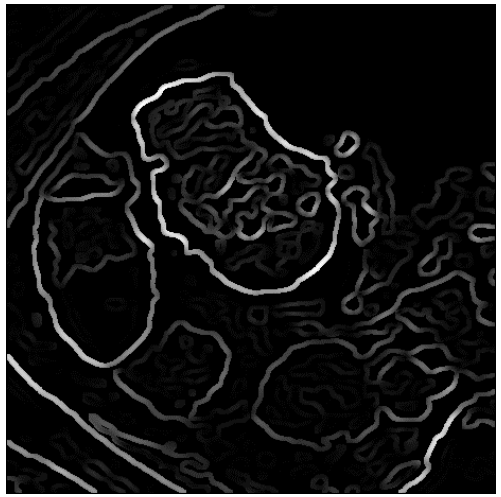
1. Anwendung eines Gradientenoperators (z.B. Sobel-Operator)
2. Non-Maximum Suppression (z.B. über Nulldurchgänge)





Canny Edge Detection (Fortsetzung)

3. Kantenverfolgung - Startpixelsuche
   - Es wird immer dasjenige Pixel mit größter Gradientenlänge selektiert.
   - Startpixel können nur Pixel sein, deren Gradientenlänge oberhalb einer Signifikanzschwelle T_1 liegt.



Canny Edge Detection (Fortsetzung)

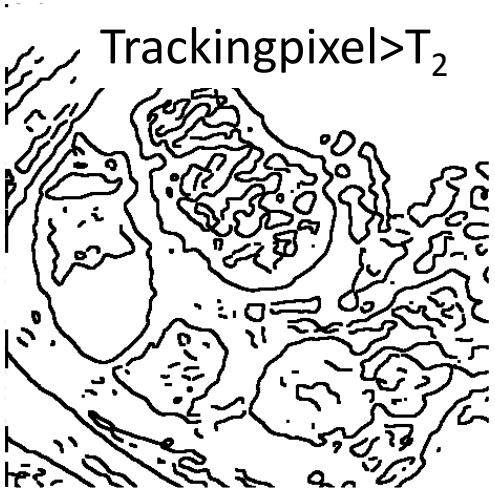
4. Kantenverfolgung - Tracking

   - Neue Kantenpixel haben eine $Gradientenlänge > T_2$ $(T_2 < T_1)$ und sind zu einem bereits gefundenen Kantenpixel benachbart.

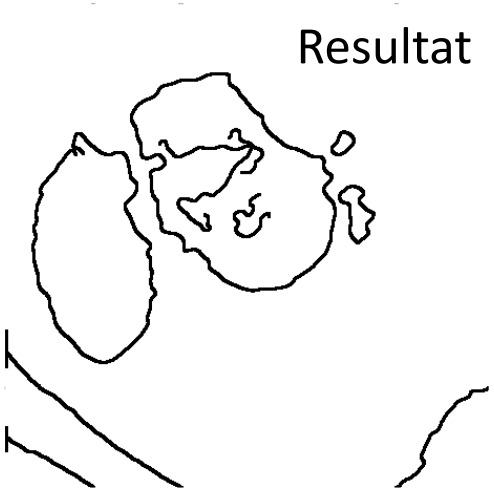
Verfahren endet, wenn keine neuen Startpixel gefunden werden.

Startpixel>T_1

Trackingpixel>T_2



Resultat



Nulldurchgänge zur Segmentierung

- Die Orte der Nulldurchgänge der zweiten Ableitung sind Ränder von zusammenhängenden Gebieten.
- Methode:
  - Laplace-Operator
  - Nulldurchgänge bestimmen:

$$\nabla^2 f(i,j) \cdot \nabla^2 shift[f(i,j)] \leq 0$$

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

Nulldurchgänge

- Die Menge der Nulldurchgänge bildet immer geschlossene Kurven = Segmente
- 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“

►„Geländehöhen“ sind die Längen der Grauwertgradienten.

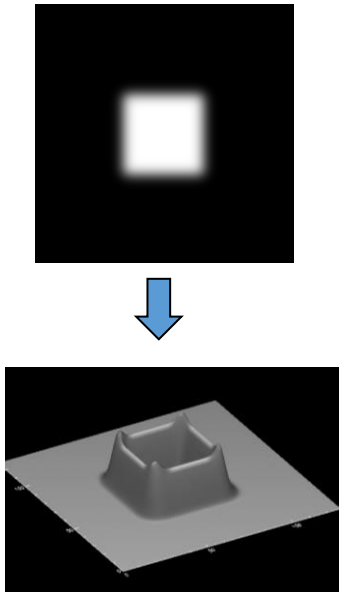
Wasserscheidentransformation

- Beregnung

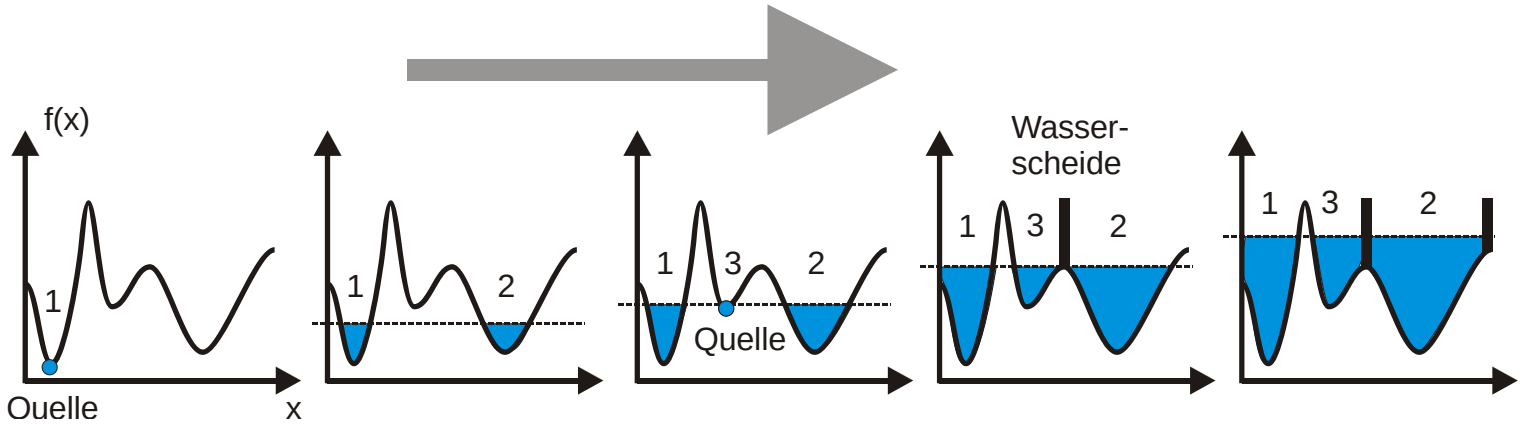
  Es fällt „Regen“ auf jedes Pixel. Anhand des Gradienten wird entschieden, wohin der Regen entwässert wird.

- Flutung

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



„Gradientengebirge“



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**:

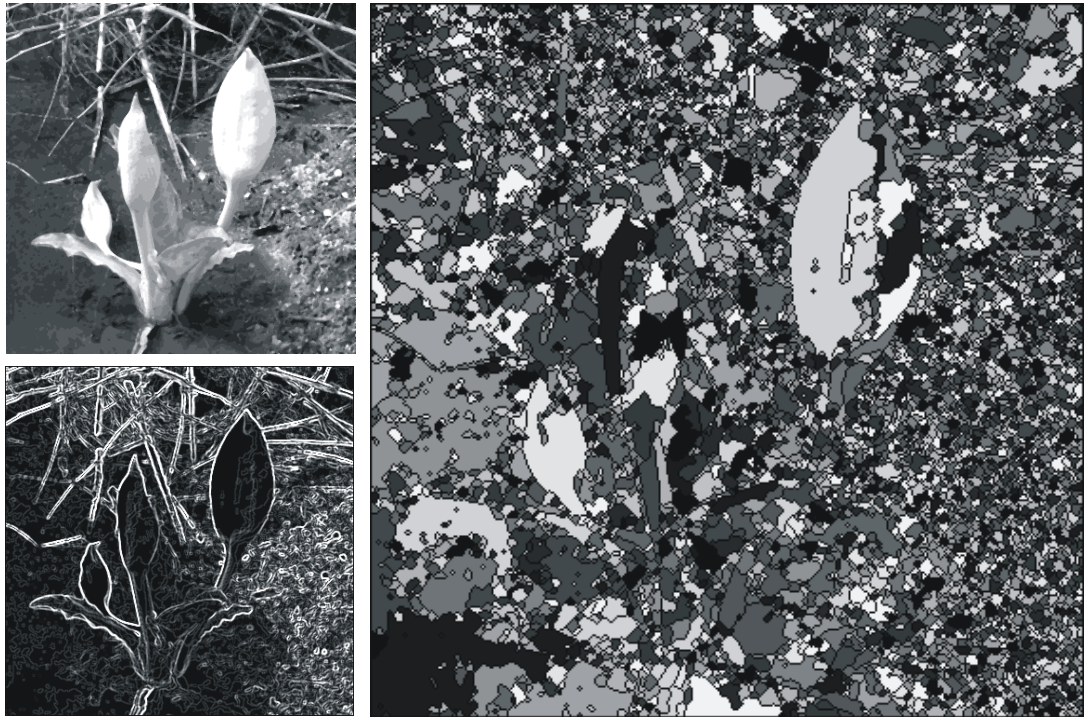
  Es zu überfluteten Pixeln von mindestens zwei Regionen benachbart.

  Dem Pixel wird das Label „Wasserscheide“ zugeordnet.

Resultat der WST

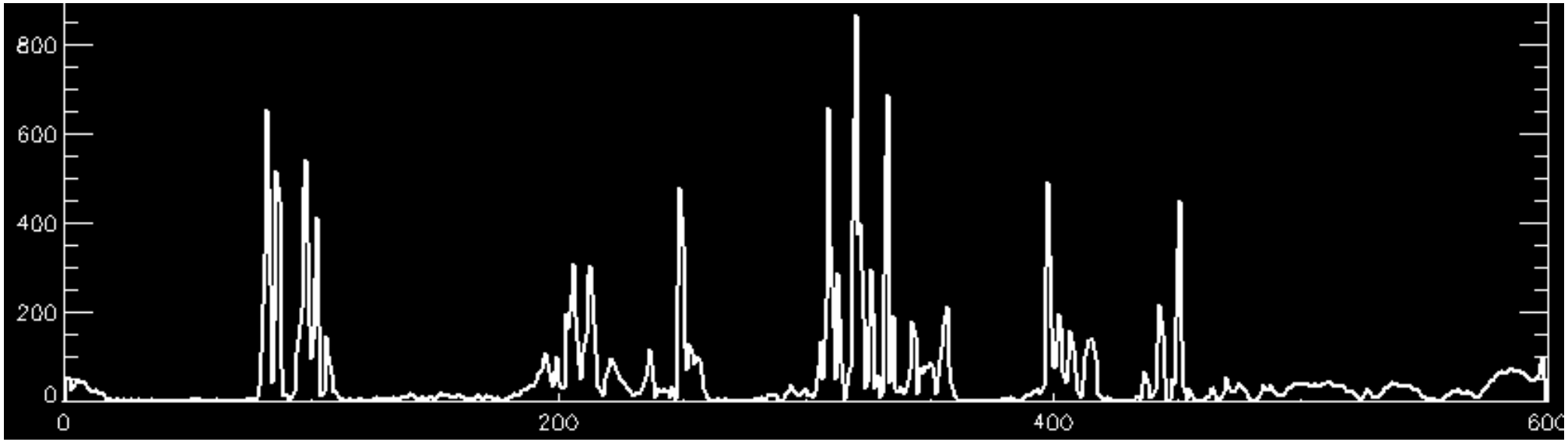
- WST auf Gradienten
- Segmentgrenzen sind lokale Maxima in Gradientenrichtung
- Maxima der ersten Ableitung = Nulldurchgänge der zweiten Ableitung

$\Rightarrow$ WST = Suche nach Nulldurchgängen



Problem Übersegmentierung

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



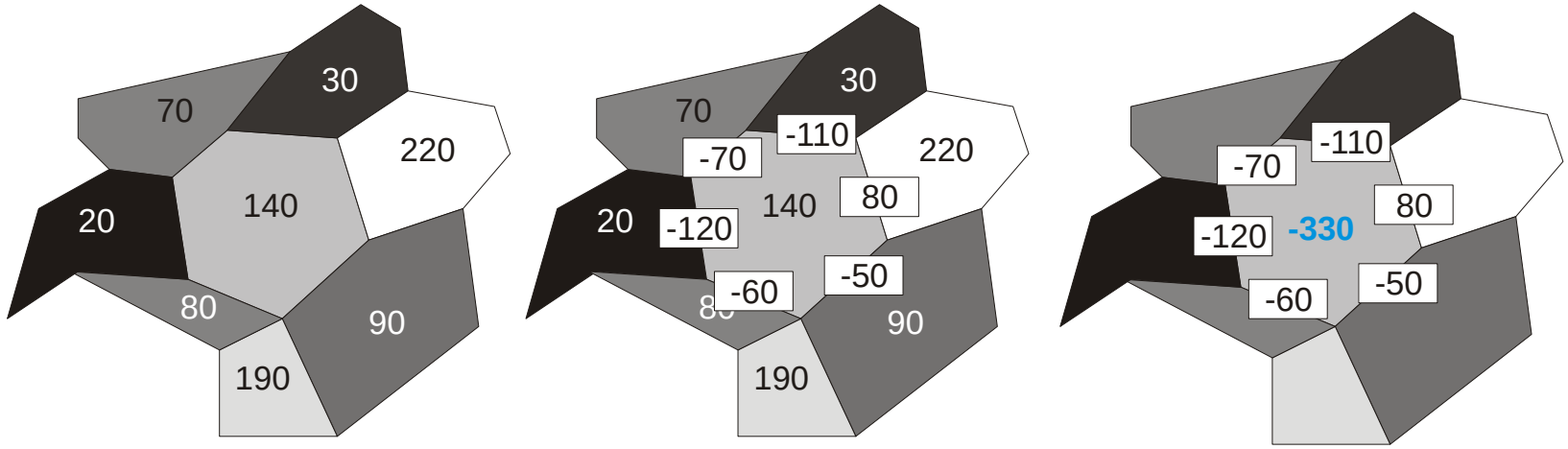
Hierarchische WST

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.

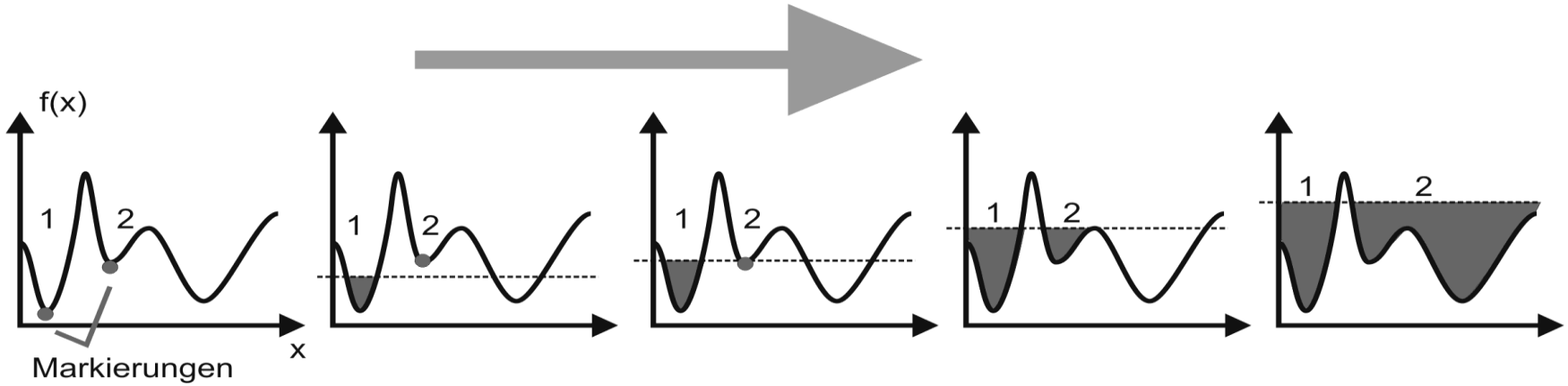
Gradienten für die hWST

- 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 der (mit der Regionengröße gewichteten) Vektoren zwischen dem Schwerpunkt der Region zu den Schwerpunkten aller benachbarten Regionen.



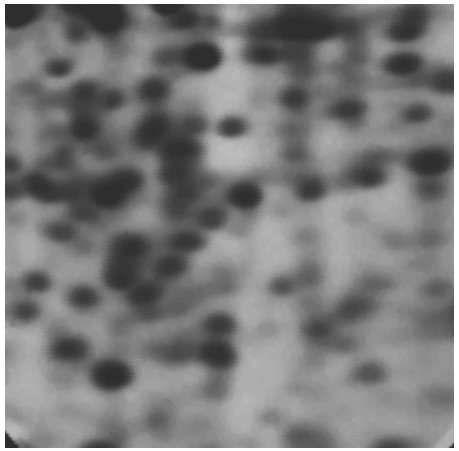
Markerbasierte WST

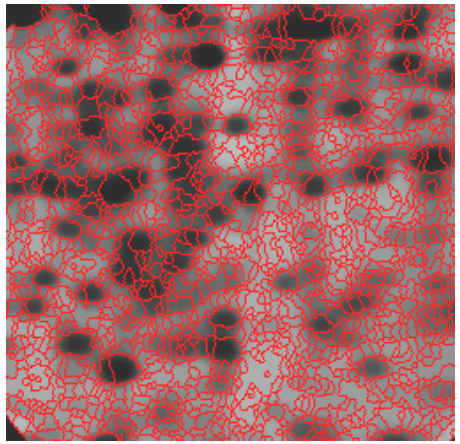
- Flutung erfolgt nur von vorher definierten Markierungen aus.
- Abwandlung des Flutungsalgorithmus:
  - Flutung erfolgt wie vorher von den Senken aus
  - Regionen erhalten das Label „undefined“, wenn sie nicht von einer Markierung aus geflutet werden
  - Wenn eine Region „undefined“ mit einer mit Label versehenen Region zusammenfließt, dann erhält sie dieses Label.



Markierungen suchen

- mWST ist im Gegensatz zur WST ein Segmentierungsverfahren, das Objektwissen (Objektorte) benötigt.
- Markierungen können z.B. erzeugt werden, falls die gesuchten Objekte mindestens in einem Punkt anhand der Helligkeit identifizierbar sind.
- Beispiel: Segmentierung in der Elektrophorese oder Zellsegmentierung.





Nachverarbeitung von Segmenten

- Übersegmentierung: Homogenität ist durch Rauschen gestört
- Charakterisierung der Störung: Rauschen verursacht sehr kleine Segmente
- Nachverarbeitung: kleinere Segmente werden benachbarten großen Segmenten zugeordnet
- Problem: Was ist das „Mastersegment“
- Lösung: relaxierendes Verfahren

Relaxation Labelling

- Jedes Pixel erhält für jedes Label eine 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

- Umsortierung aller Pixel in Liste $p_0, p_1, \ldots, p_N$.
- Initiale Labelwahrscheinlichkeit P^0 für jedes Pixel p_i 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
  - gibt an, wie sicher man sich nach der Segmentierung über das zugeordnete Label ist
  - 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:

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

- Kompatibilität für Binärbilder (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)

- Kompatibilität bei mehr als zwei Labeln kann auch bedeuten, dass sich bestimmte Labelpaare unterstützen *(auch wenn es unterschiedliche Label sind)*

Unterstützung eines Pixels

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

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

- Erinnerung: Pixel p_j hat für *jedes* Label eine von Null verschiedene Labelwahrscheinlichkeit
- Labelwahrscheinlichkeit für ein Label l_l wird mit der Kompatibilität zwischen l und dem zu unterstützenden Label l_k gewichtet
- Summe ist dann die Unterstützung durch alle Label

Unterstützung eines Pixels

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.

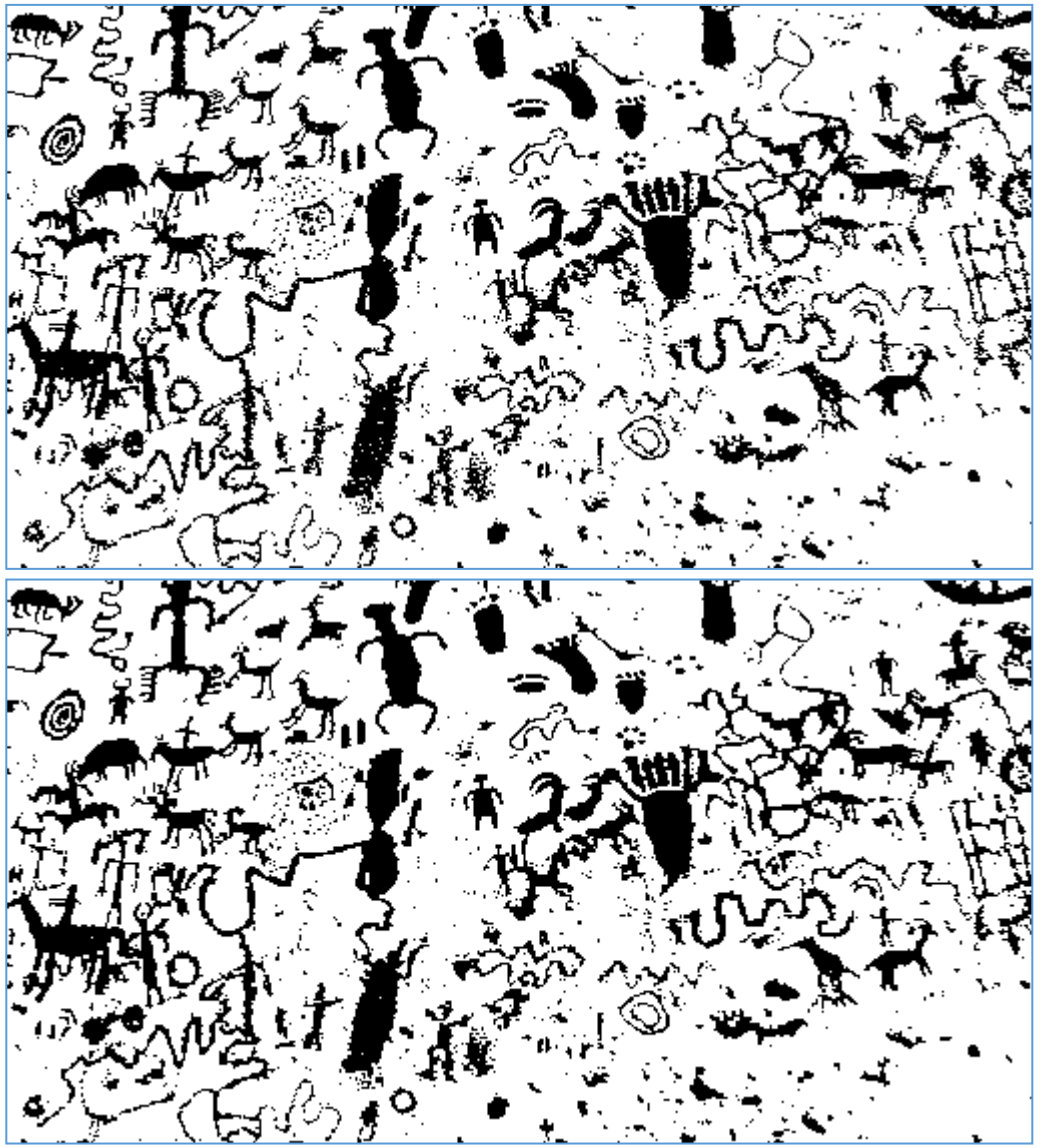
$$c_{ij} = \begin{cases} 1/8 & , \text{falls } p_j \in N_8(p_i) \\ 0 & , \text{sonst.} \end{cases}$$

Nachbarschaft kann sehr allgemein definiert sein und der „Grad der Nachbarschaft“ wird durch die Gewichtung c_{ij} berücksichtigt

Iterationsschritt

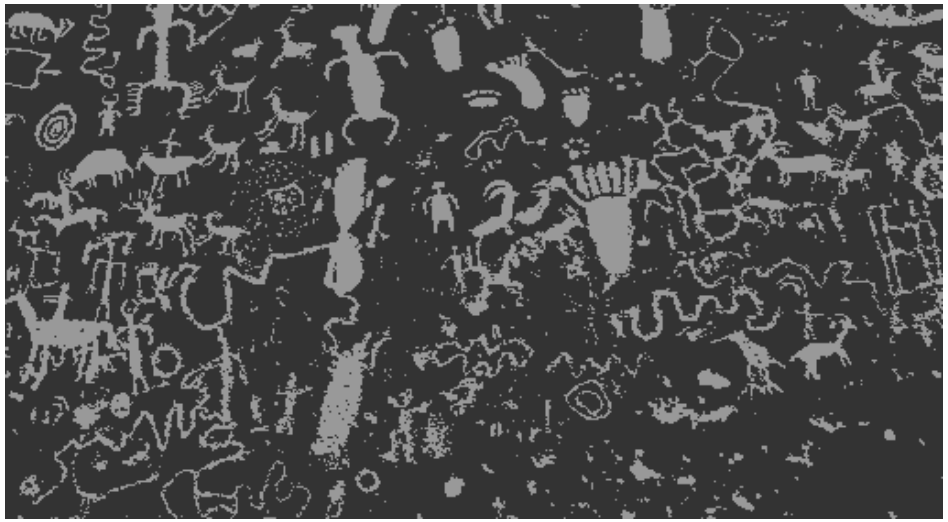
$$P^{(n+1)}(p_i, l_k) = \frac{P^{(n)}(p_i, l_k)\left[1 + Q^{(n)}(p_i, l_k)\right]}{\sum_{l=1}^{K-1} P^{(n)}(p_i, l_l)\left[1 + Q^{(n)}(p_i, l_l)\right]}.$$

- Unterstützung *Q* durch die umgebenden Pixel wird addiert.
- Division durch die Summe aller Labelwahrscheinlichkeiten dient der Normierung



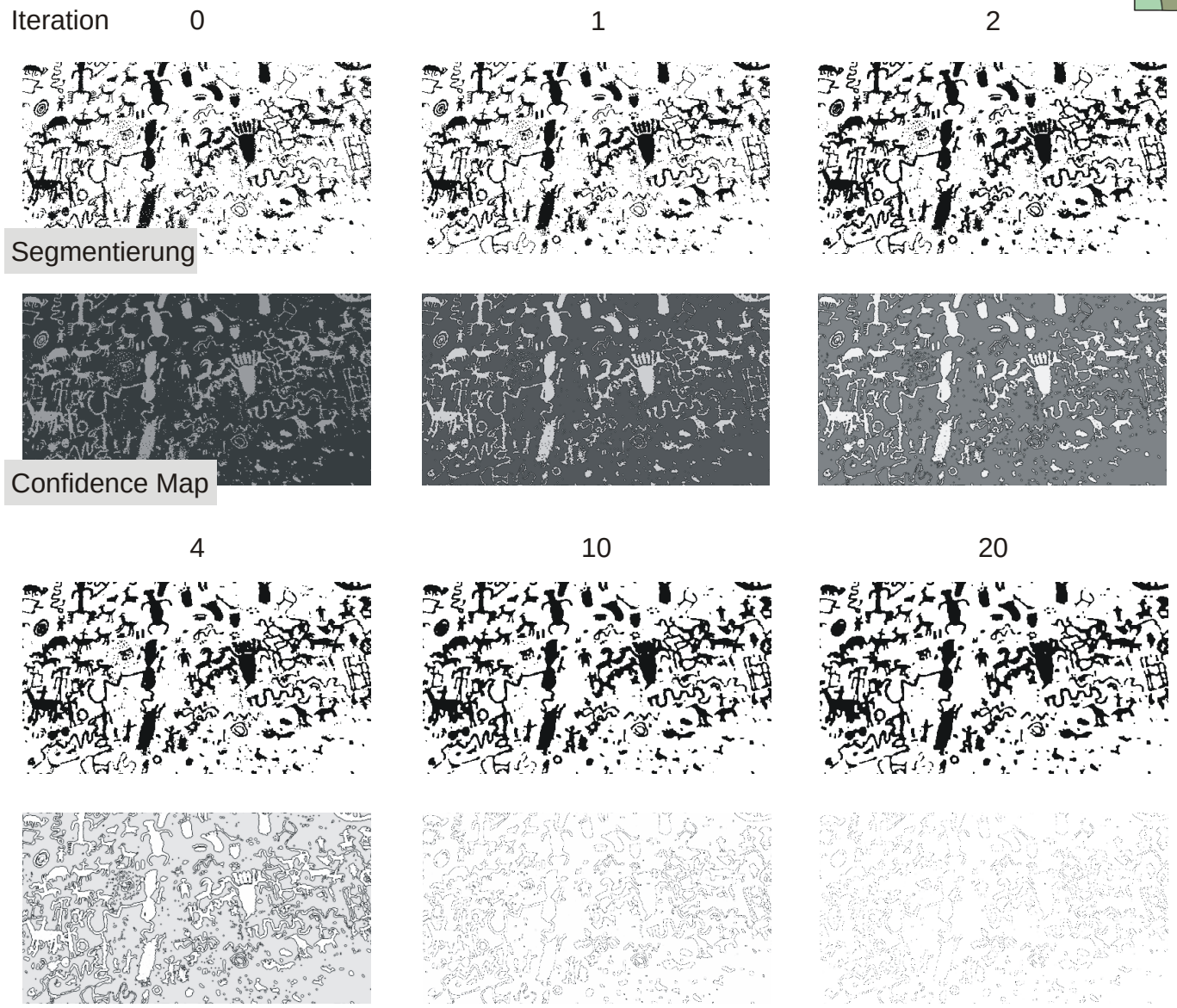
Confidence Map

- Um das Konvergenzverhalten zu beobachten, kann eine **Confidence Map** erzeugt werden.
- Confidence Map: Gibt für jedes Pixel die Zuverlässigkeit der derzeitigen Entscheidung an
- 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



Zwischenfazit Segmentierung

- Segmentierungskriterien
  - möglichst homogene Gebiete
  - möglichst kurze Segmentgrenzen
- Probleme
  - Homogenitätskriterium kann schwer zu definieren sein (vor allem bei Beleuchtungseinflüssen)
  - Homogenität kann in unterschiedlichen Bereichen etwas unterschiedliches bedeuten
  - Segmentgrenzen sind nicht immer Grauwertkanten

Berkeley Segmentation Dataset

Warum es sich dennoch lohnt

- Es gibt Anwendungen, bei denen die Bilder (die Bildklasse) nicht vorher bekannt ist, z.B.
  - Komprimierung (z.B. MPEG-7)
  - Suche in Bilddatenbanken
- Was hier benötigt wird
  - Segmente, die charakteristisch sind (d.h. in ähnlichen Bildern ähnlich gefunden werden)
  - Segmente, die mehr Bedeutung tragen als Pixel
- Man kann aber auch anders segmentieren, wenn ein bestimmtes Objekt gesucht ist $\Rightarrow$Segmentierung mit Objektwissen

Was Sie heute gelernt haben sollten

- Edge Linking und Canny Edge Operator
- Nulldurchgänge zur Regionensegmentierung
- Wasserscheidentransformation
  - Multiskalenstrategie
  - WST mit Markern
- Relaxation Labeling

Famous Last Question

Wie könnte man Segmentierung und Objektwissen verbinden, um die Figuren zu extrahieren?

Welche Art von Objektwissen?