

# Computer Vision

Kanten, Schwellen & Regionen - Segmentierung &  
Kantendetektion

Alexandra Posekany

December 2025

# Einführung Segmentierung: Vom Pixel zum Objekt

## Was ist Segmentierung?

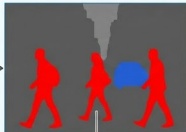
**Definition:** Der Prozess der Unterteilung eines digitalen Bildes in mehrere Segmente (Mengen von Pixeln, Superpixel).

**Ziel:** Vereinfachung der Darstellung eines Bildes hin zu etwas, das leichter zu analysieren ist.

Originalbild



Semantische Segmentierung



Alle Pixel der Klasse "Mensch" bekommen dieselbe Farbe.

Instanz-Segmentierung



Unterscheidung zwischen "Mensch 1" (Grün), "Mensch 2" (Gelb) und "Mensch 3" (Lila).

# Arten der Segmentierung

Bildsegmentierung ist die **Identifikation** und **Isolation** eines Bildes in unterschiedliche Teilbereiche (Regionen). Diese entsprechen **strukturellen Einheiten** (Segmente).

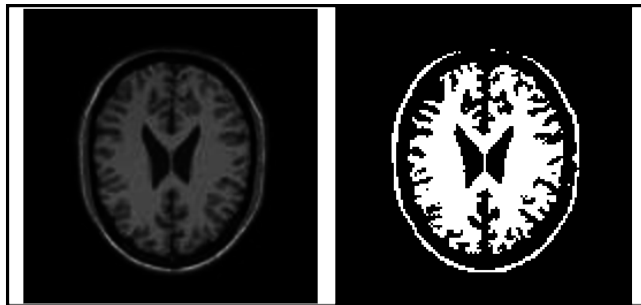


# Pixelbasierte Segmentierung - Thresholding

**Input:** Graustufenbild

**Output:** Binäres Bild (Schwarz-Weiß-Bild)

Schwellenwert-Methoden erstellen binäre Bilder und klassifizieren Pixel basierend darauf, ob ihre Intensität über oder unter einem bestimmten Grenzwert liegt. Histogramme, die die Häufigkeit bestimmter Pixelwerte in einem Bild darstellen, werden häufig zum Definieren von Schwellenwerten verwendet.



# Pixelbasierte Segmentierung - Thresholding

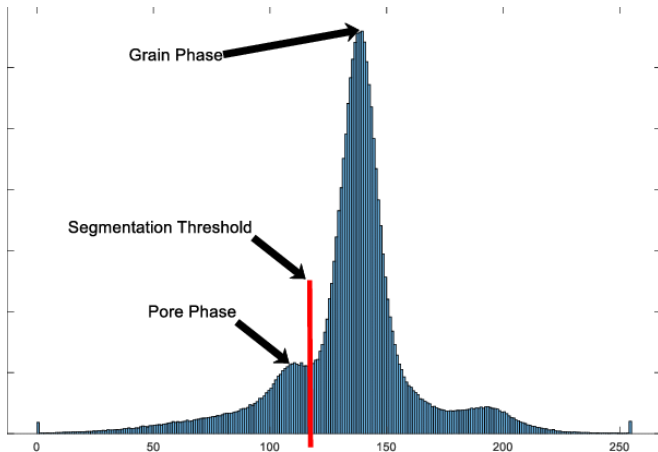
Filterung anhand der Helligkeit, indem ein Schwellwert (Threshold) überschritten oder unterschritten wird

$$\text{Formel: } g(x, y) = \begin{cases} 255 & \text{wenn } f(x, y) > T \\ 0 & \text{sonst} \end{cases}$$

Anwendung: Trennung von Vordergrund (Objekt) und Hintergrund, wenn der Kontrast hoch ist.

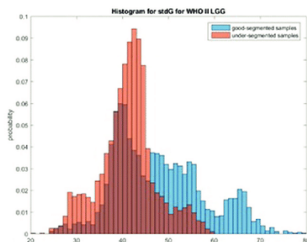
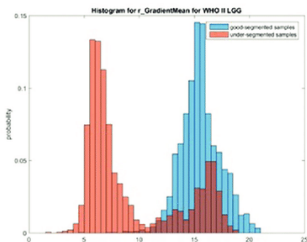
# Thresholding mithilfe des Histogramms

Histogramme können beispielsweise die Werte von Hintergrundpixeln ableiten und so zur Isolierung von Objektpixeln beitragen.



# Wann ist Thresholding sinnvoll?

- ▶ wenn die zu unterscheidenden Strukturen hinreichend weit von einander entfernt sind (Effektgröße)
- ▶ wenn das Hintergrundrauschen nicht zu stark ist (Variation, Noise)



## Thresholding 2.0 – Wenn Global versagt

Das Problem: Globales Thresholding (z.B.  $> 127$ ) versagt bei ungleichmäßiger Beleuchtung (Schatten auf einem Blatt Papier).

### **Adaptive Thresholding**

Der Schwellenwert wird für jeden Pixel individuell berechnet, basierend auf seinen Nachbarn (z.B. Mittelwert der 11x11 Umgebung).

Effekt: Funktioniert wie eine lokale Kontrastverstärkung.

### **Otsu's Methode**

Automatische Suche nach dem perfekten globalen Schwellenwert.

Prinzip: Suche im Histogramm das "Tal" zwischen zwei "Bergen" (Vordergrund/Hintergrund). Minimiert die Varianz innerhalb der Klassen.



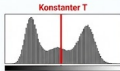
# Adaptives Thresholding

## ADAPTIVE THRESHOLDING: Die Lösung für ungleiche Beleuchtung

Der Schwellenwert  $T$  ist nicht konstant, sondern **variabel** für jeden Pixel  $T(x,y)$ .

### DAS PROBLEM: Globaler Schwellenwert ( $T$ ist konstant)

Sudoku im Schatten



Ergebnis (Global)  
Versagt bei Farbverläufen & Schatten

### DIE LÖSUNG: Adaptiver Schwellenwert $T(x,y)$ (variabel)

Sudoku im Schatten



Methode:  $T(x,y)$  Berechnung

$$T(x,y) = \text{mean}(\text{block}) - C \quad \begin{matrix} \text{C (Feinjustierung)} \\ \text{oder gewichteter Mittelwert (Gaussian)} \end{matrix}$$



Ergebnis (Adaptiv)  
Funktioniert auch bei Farbverläufen

### VORTEIL & ZUSAMMENFASSUNG

#### Methode

- $T(x,y)$  basiert auf lokaler Nachbarschaft (z.B. 11x11 Block)
- Berechnung via Mittelwert (Mean) oder Gaussian
- Formel:  $T = \text{mean}(\text{block}) - C$



#### Vorteil

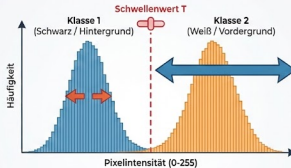
- Kompensiert lokale Helligkeitsunterschiede (Schatten, Verläufe) für eine robuste Segmentierung.

# Otsu's Methode

## Otsu's Methode (Automatisches T)

Ziel: Finde  $T$ , sodass die Trennung der Klassen optimal ist.

Minimiere  
Intra-Class Variance



Maximiere  
Inter-Class Variance



Algorithmus

1. Durchsuche alle Werte  $T$  (0-255)

2. Berechne für jedes  $T$  die Intra-Class Varianz

3. Wähle  $T$ , das die Varianz minimiert (Optimales  $T$ )

⚠ Einschränkung



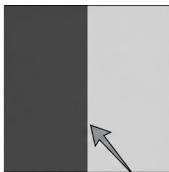
# Kantendetektion

Die kantenbasierte Segmentierung ist ein fundamentales Verfahren in der Bildverarbeitung, das darauf abzielt, Objekte in einem Bild zu isolieren, indem ihre Grenzen (Kanten) identifiziert werden.

Kantenerkennungsmethoden identifizieren die Grenzen von Objekten oder Klassen durch die Erkennung von Helligkeits- oder Kontrastunterbrechungen.

## Die Mathematik der Kante

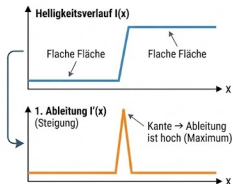
### Intuition



Eine Kante ist dort, wo sich die Helligkeit drastisch ändert.

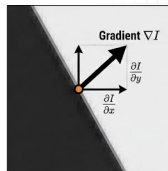


### Mathematisch



Die 1. Ableitung (Steigung) der Helligkeitsfunktion  $I(x)$ .  
Flache Fläche  $\rightarrow$  Ableitung  $\approx 0$ .  
Kante  $\rightarrow$  Ableitung ist hoch (Maximum).

### Der Gradient



Ein Vektor, der in Richtung der steilsten Änderung zeigt.

-  Magnitude (Stärke der Kante)
-  Richtung (Ausrichtung der Kante)

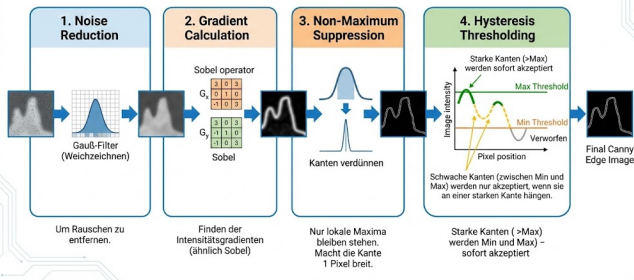
# Ablauf des Canny-Algorithmus

1. **Bildglättung** (Rauschunterdrückung, Noise Reduction): Gauß-Filter (Weichzeichnen), um Rauschen zu entfernen.
2. Berechnung der **Gradienten** (Kantenfilterung, Gradient Calculation): Finden der Intensitätsgradienten (ähnlich Sobel).
3. **Non-Maximum Suppression** (Kantenverdünnung): Nur lokale Maxima bleiben stehen. Macht die Kante 1 Pixel breit.
4. **Schwellenwertverfahren** (Hysteresis, Hysteresis Thresholding): Zwei Schwellenwerte (Min/Max).
  - ▶ Starke Kanten ( $> \text{Max}$ ) werden sofort akzeptiert.
  - ▶ Schwache Kanten (zwischen Min und Max) werden nur akzeptiert, wenn sie an einer starken Kante hängen.
5. **Kantenverbindung** (Edge Linking)

# Canny-Algorithmus

## Canny Edge Detector (Der Goldstandard)

Canny – Der intelligente Algorithmus (1986)



# Kantenbasierte Segmentierung

## 2. Schritt: **Kantenfindung** (Sobel filter, Laplacian, Canny filter)

### Diskrete Ableitung & Faltung

**Wir haben keine Funktionen, sondern Pixel.**  
**Finite Differenz:**  $f'(x) \approx f(x+1) - f(x-1)$

Dies wird durch Faltung (Convolution) mit einem Kernel realisiert.  
Beispiel Kernel:  $[-1, 0, 1]$

**Der Sobel-Operator**  
Kombiniert Glättung (gegen Rauschen) und Differenzierung.

**Sobel X (Vertikale Kante):**  
Betont Unterschiede links/rechts.

**Sobel Y (Horizontale Kante):**  
Betont Unterschiede oben/unten.

**Laplacian Operator**  
Basiert auf der 2. Ableitung.

Eigenschaft: Isotrop (Richtungsunabhängig).  
Reagiert auf jede Änderung.

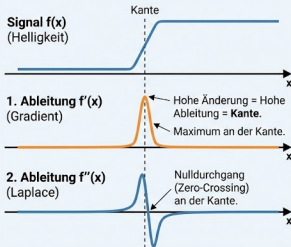
Nachteil: Extrem empfindlich gegenüber Rauschen. Findet oft "zu viel".

# Gradienten

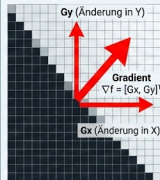
## Die Mathematik der Kante und der Bild-Gradient

### WAS IST EINE KANTE MATHEMATISCH?

Intuition: Helligkeitswechsel.



### DER BILD-GRADIENT



Magnitude (Stärke):

$$|G| = \sqrt{G_x^2 + G_y^2}$$
$$\approx |G_x| + |G_y| \text{ (für Speed)}$$

Richtung (Winkel):

$$\theta = \arctan(G_y/G_x)$$

**Die Kantenrichtung steht senkrecht zum Gradienten!**

**Bedeutung:** Wir müssen Änderungen in X-Richtung und Y-Richtung getrennt berechnen.

# Kantenbasierte Segmentierung

## 3. Schritt: **Non-Maximum Suppression** (Kantenverdünnung)

Das Ergebnis aus Schritt 2 liefert oft breite, verschwommene Kanten. Wir wollen aber dünne, präzise Linien.

Der Algorithmus wandert entlang der Gradientenrichtung. Wenn ein Pixel nicht das lokale Maximum (der stärkste Wert) in seiner Nachbarschaft ist, wird es auf Null gesetzt (unterdrückt). Das Resultat sind "dünne" Kanten (1 Pixel breit).



# Kantenbasierte Segmentierung

## 4. Schritt: **Schwellenwertverfahren (Hysterese)**

Entscheidung: Was ist eine “echte” Kante und was ist nur eine leichte Helligkeitsschwankung?

Man definiert zwei Schwellenwerte (Thresholds):  $T_{high}$  und  $T_{low}$ .

- ▶ Werte über  $T_{high}$  sind sichere Kanten.
- ▶ Werte unter  $T_{low}$  werden verworfen.
- ▶ Werte dazwischen sind “schwache Kanten”. Sie werden nur dann akzeptiert, wenn sie direkt mit einer “sicheren Kante” verbunden sind.

# Kantenbasierte Segmentierung

## 5. Schritt: **Kantenverbindung** (Edge Linking)

Nach der Filterung hat man oft noch Lücken in den Konturen. Um ein Objekt wirklich zu segmentieren (also vom Hintergrund zu trennen), muss die Kante geschlossen sein, also einzelne Kanten zu einer **Verkettung** (Kantenzug, Polygon) erweitern.

- ▶ Edge Relaxation ('erleichtert' die Kantenzugbedingung, indem die lokale Umgebung berücksichtigt wird)
- ▶ Graph searching (sucht Pixel, die 'Kandidaten' sind, um einzelne Kanten zu Pfaden zu verbinden, wobei Kosten minimiert werden)
- ▶ Hough transformation (robustes globales Verfahren zur Erkennung von Geraden, Kreisen oder beliebigen anderen parametrisierbaren geometrischen Figuren)

# Hough Transformation

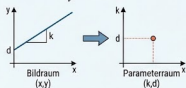
## HOUGH TRANSFORMATION

Transformation in eine parametrisierbare Notation der geometrischen Figur

### 1. Transformation in Parameterraum

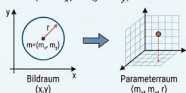
Gerade (Line)

$$y = k \cdot x + d$$



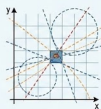
Kreise (Circles)

$$(x - m_x)^2 + (y - m_y)^2 = r^2$$



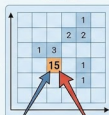
### 2. Der Voting-Prozess (Voting)

Alle Pixel werden bzgl. Dazupassen zu einer solchen Figur geprüft



Ein Pixel kann zu vielen möglichen Geraden/ Kreisen gehören!

Akkumulator-Array (Voting)



wenn sie auf eine Gerade passen, wird k und d geschätzt und der Geradencounter erhöht

Lokales Maximum = Erkannte Figur

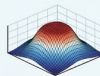
### 3. Nachteile (Herausforderungen)



"Brute-Force-Ansatz", daher sehr rechenaufwändig



wenn mehrere Geraden sich ein Pixel "teilen", kommt es zu Häufungsplateaus



Häufungsplateau (Unschärfe Erkennung)

# Vergleich Sobel vs. Canny

## Sobel vs. Canny Edge Detection: Ein direkter Vergleich

### SOBEL: Schnell & Einfach

- Schnell, einfach, aber Kanten sind dick und oft unterbrochen.
- Gut als Input für weitere Algorithmen.



Sobel-Ergebnis

### CANNY: Präzise & Sauber

- Langsamer, aber liefert saubere, dünne, zusammenhängende Linien.
- Gut für menschliche Interpretation.



Canny-Ergebnis

### GRAFIK: DIREKTER BILDVERGLEICH



Originalbild

Originalbild



Sobel-Ergebnis

Sobel-Ergebnis

Dick, unterbrochen, rauschig



Canny-Ergebnis

Canny-Ergebnis

Dünn, zusammenhängend, sauber

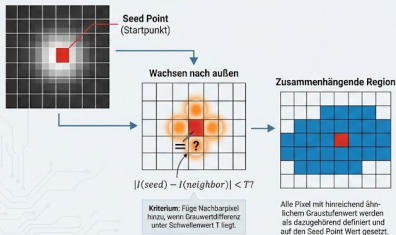
# Region Growing

Ausgehend von einem **Seed Point** wird eine gesamte zusammenhängende Region als identifiziert, indem alle Pixel mit hinreichend ähnlichem Graustufenwert als dazugehörend definiert und auf den Seed Point Wert gesetzt werden.

## Region Growing: Agglomerative Image Segmentation

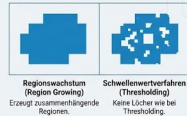
Ausgehend von einem oder mehreren „Seed-Pixeln“ gruppieren Regionswachstumsalgorithmen benachbarte Pixel mit ähnlichen Eigenschaften. Algorithmen können agglomerativ (wachsend) oder spaltend sein.

### 1. Der Prozess (Agglomerativ)



### 2. Vor- und Nachteile

#### ✓ Vorteil (Pro)



#### ✗ Nachteil (Con)



# Herausforderungen von Region Growing

- ▶ **Ähnlichkeitsbereich zu klein** nicht die gesamte gesuchte Region wird erkannt und markiert
- ▶ **Ähnlichkeitsbereich zu groß** Überlaufen des segmentierten Bereichs in Nachbarregionen, die eigentlich abgetrennt werden sollen (Problem von Fehldiagnosen von Tumoren etc. )

similarity range  
too small

similarity range  
too large



# Ablauf von Region Growing

## Initialisierung

1. Wahl vom Startpunkt (Seed): Manuell oder automatisch wird die initiale Region gewählt.

## Iterativer Prozess für jede bestehende Region:

2. Untersuche benachbarte Pixel oder Pixelgruppen, etwa die 4 oder 8 Nachbarn.
3. Vergleiche die Eigenschaften (z.B. Grauwerte) der Nachbarn mit der Region.
4. Füge Nachbarn zur Region hinzu, wenn sie ein Homogenitätskriterium erfüllen (innerhalb des Graustufenintervalls liegen).
5. Aktualisiere die Eigenschaften der Region (z.B. Durchschnittswert, Indikatormatrix der inneren Punkte).

Wiederhole den Prozess, bis kein weiteres Wachstum mehr möglich ist (while Schleife)

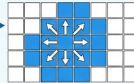
# Abbruchkriterien

- ▶ Vorgegebene Anzahl an Iterationen: Ein einfaches Kriterium ist die Festlegung einer maximalen Anzahl von Durchläufen, z.B. 5 oder 10 Iterationen. Dies garantiert eine begrenzte Rechenzeit, kann aber zu suboptimalen Ergebnissen führen.
- ▶ Ausschluss aller Nachbarpixel: Der Algorithmus endet, wenn keine Nachbarpixel mehr die Kriterien für eine Aufnahme in die Region erfüllen

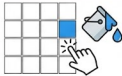


# Flood Fill

## The Flood Fill Algorithm: Filling Connected Areas



### 1. Select Start & Target Color



User chooses a starting pixel and the fill color.

### 2. Check & Change Color



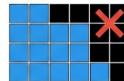
If pixel matches target color, change to new color.

### 3. Spread to Neighbors (Recursion/Queue)



Algorithm checks adjacent pixels and repeats the process.

### 4. Stop at Boundary



Stops when a different color (boundary) or already filled pixel is hit.



### Real-World Example

Used in image editing for filling shapes, mazes, and game development.

Before



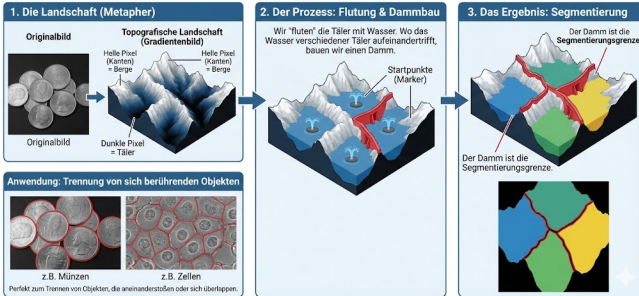
After



# Watershed Algorithmus

## Watershed Algorithmus (Wasserscheide)

Metapher: Betrachtet das Bild (Gradientenbild) als topografische Landschaft.



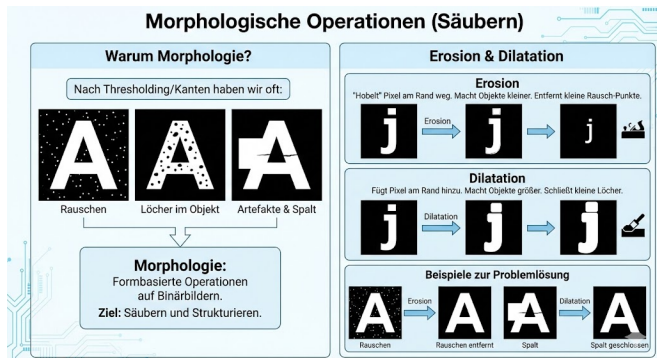
# Nachbearbeitung nach der Segmentierung

Die Nachbearbeitung dient dazu, Fehler zu korrigieren, die Genauigkeit zu erhöhen und die Segmentierungsergebnisse für die weitere Analyse oder klinische Anwendung zu optimieren, um folgende Probleme zu beheben:

- ▶ **Unvollständige Segmentierung:** Viele Segmentierungsverfahren, etwa Schwellenwertverfahren, erzeugen keine vollständig zusammenhängenden Segmente, besonders bei stärkerem Rauschen.
- ▶ **Randungenauigkeiten:** Die Genauigkeit an den Rändern der segmentierten Objekte lässt oft zu wünschen übrig, besonders bei komplexen anatomischen Strukturen.
- ▶ **Eckige Kanten:** Einige Verfahren wie Erosion und Dilation erzeugen teilweise eckige Kanten, die eine Glättung erfordern.
- ▶ **Trennung überlappender Objekte:** Besonders bei der Zellsegmentierung kann die Trennung einzelner Zellen eine aufwändige Nachbearbeitung erfordern.

# Morphologische Operationen

Morphologische Operationen sind eine Reihe von Algorithmen, die zur Analyse und Bearbeitung der Form und Struktur von Objekten in Bildern verwendet werden. Sie nutzen spezielle Strukturelemente, um Pixel in Bildern zu beeinflussen, für Anwendungen wie Objekterkennung, Rauschunterdrückung und Bildsegmentierung.



# Grundoperationen der Morphologie

- ▶ Dilatation (Erweiterung): Fügt Pixel zu den Grenzen von Objekten hinzu. Diese Operation vergrößert die weißen Regionen eines binären Bildes.
- ▶ Erosion (Abtragung): Entfernt Pixel von den Grenzen von Objekten. Sie verkleinert die weißen Regionen, indem sie kleinere Strukturen entfernt.
- ▶ Öffnen: Eine Sequenzoperation, die zuerst eine Erosion gefolgt von einer Dilatation ausführt. Diese Technik kann kleinere Objekte oder Störungen aus dem Bild entfernen.
- ▶ Schließen: Das Gegenteil von Öffnen, beginnend mit einer Dilatation gefolgt von einer Erosion. Diese Methode füllt kleinere Löcher innerhalb von Objekten.

# Dilation, Erosion

## ► **Dilatation** (Ausdehnung, Erweiterung)

Das aktive Pixel wird auf den **Maximumwert** der Umgebungspixel gesetzt. Die **Region** wird dadurch **erweitert**

$$\begin{bmatrix} 11 & 45 & 23 \\ 108 & ? & 0 \\ \mathbf{210} & 71 & 98 \end{bmatrix} \Rightarrow \begin{bmatrix} 11 & 45 & 23 \\ 108 & \mathbf{210} & 0 \\ \mathbf{210} & 71 & 98 \end{bmatrix}$$

## ► **Erosion** (Abtragung, Verkleinerung)

Das aktive Pixel wird auf den **Minimumwert** der Umgebungspixel gesetzt. Die **Region** wird dadurch **verkleinert**

$$\begin{bmatrix} 11 & 45 & 23 \\ 108 & ? & \mathbf{0} \\ 210 & 71 & 98 \end{bmatrix} \Rightarrow \begin{bmatrix} 11 & 45 & 23 \\ 108 & \mathbf{0} & 0 \\ \mathbf{210} & 71 & 98 \end{bmatrix}$$

# Kombination der Methoden

- ▶ **Opening**, erst Erosion, dann Dilation

kleine Objekte und Hintergrundrauschen werden entfernt, große Objekte bleiben unbeeinträchtigt

- ▶ **Closing**, erst Dilation, dann Erosion

verbindet große Strukturen, indem 'Löcher' oder 'Spalten' geschlossen werden, die morphologisch nicht relevant sind

# Opening - Strukturauftrennung

Original

0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	0	0	255	255	0	0	0	0
0	0	0	0	255	0	0	0	0	0
0	0	0	0	255	0	0	0	0	0
0	0	0	0	255	0	0	0	0	0
0	0	0	0	255	0	0	0	0	0
0	0	0	0	255	0	0	0	0	0
0	0	0	255	255	0	0	0	0	0
0	0	255	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0

Schritt 1

0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	255	255	255	0	0	0	0
0	0	0	255	255	255	0	0	0	0
0	0	0	255	255	255	0	0	0	0
0	0	0	255	255	255	0	0	0	0

Schritt 2

0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	255	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0



# Opening - Artefaktentfernung

Original

0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	0	0	255	255	255	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	255	0	0	0	0	0	0	0
0	0	255	0	0	0	0	255	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	255	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0

Schritt 1

0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	0	0	255	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	255	255	255	0	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0

Schritt 2

0	0	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0
0	0	255	255	255	255	255	255	0
0	0	0	0	255	255	255	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0
0	0	255	255	255	255	255	255	0
0	0	255	255	255	255	255	255	0
0	0	255	255	255	255	255	255	0

## Closing - Spaltenschließung

**Original**

[illegible]

### Schritt 1

[illegible]

## Schritt 2

[illegible]