

Objektbasierte Klassifikation

Christian Wilms

Computer Vision Group
Universität Hamburg

Wintersemester 17/18

14. November 2016

Übersicht

1 Binarisierung

2 Bildtransformation

3 Merkmale

Wir wollen Bilder auf Basis eines Objekts klassifizieren



Wir können wir das Objekt zur Klassifikation nutzen?



- Objekt (Vordergrund) und Hintergrund trennen (Binarisierung)
 - nur das Objekt oder die Box mit dem Objekt betrachten
 - der Hintergrund ist einfarbig und sehr konstant ⇒ Zerlegung anhand von Grauwert/Farbe
 - Maske für Vordergrund bzw. Hintergrund (Binärbild)
 - Form der Objekte kann ein Merkmal werden

Binärbild

- ein Binärbild hat nur zwei mögliche Pixelwerte
 - 0 = schwarz = False
 - 1 = weiß = True
 - nützlich zur Maskierung des Bildes
 - Vordergrund vs. Hintergrund
 - interessante Regionen definieren

Binarisieren

Binarisieren

Binarisieren ist die Zuordnung aller Pixel in einem Bild zu den Klassen 0 oder 1.

- einseitiges Schwellenwertverfahren

$$I_{result}(x, y) = \begin{cases} 1 & \text{if } I(x, y) < \tau \\ 0 & \text{else} \end{cases}$$

- zweiseitiges Schwellenwertverfahren

$$I_{result}(x, y) = \begin{cases} 1 & \text{if } \tau_1 < I(x, y) < \tau_2 \\ 0 & \text{else} \end{cases}$$

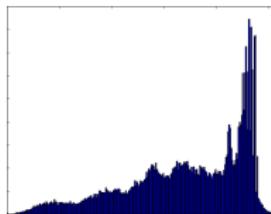
- globale Methode \Rightarrow beide Klassen müssen sich global trennen lassen
 - Ermittlung des Schwellenwerts über Histogramm (manuell oder automatisch), Kontextwissen oder Trial and Error

Binarisieren - schlechtes Beispiel

Region-based segmentation

Let us first determine markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

```
>>> markers = np.zeros_like(coins)
```

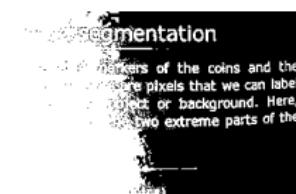
 I


Histogramm

Region-based segmentation

Let us first determine markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

```
markers = np.zeros_like(coins)
```

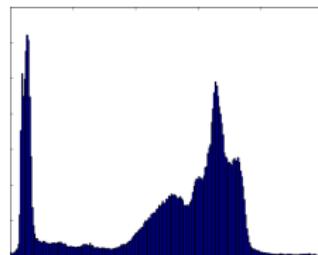
 $I < 100$

 $I < 200$

Schlechte Ergebnisse, da Schrift und Seite sich nicht global trennen lassen.

Binarisieren - gutes Beispiel



I



bimodales Histogramm



$I < 80$

- gute Ergebnisse bei bimodalem Histogramm (zwei Hügel und ein Tal dazwischen)
- Mögliche Probleme: Vordergrund hat Löcher, im Hintergrund werden einige Bereiche als Vordergrund erkannt \Rightarrow Morphologische Operationen anwenden

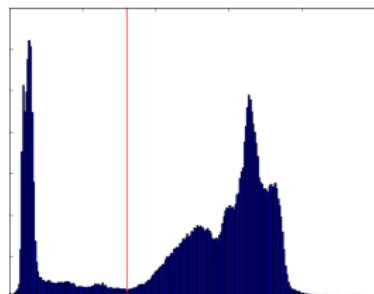
Otsu-Verfahren

Problem: Schwellenwert manuell finden ist unpraktisch!

Prinzip

- als Ergebnis wollen wir zwei Klassen haben, die innerhalb eine möglichst kleine Varianz aufweisen
- identisch zur maximalen Varianz zwischen den Klassen
- Grundlage ist wieder ein Histogramm
- Otsu-Verfahren probiert alle möglichen Schwellenwerte aus
- berechnet zu jedem Schwellenwert die Varianz zwischen Klassen
- Schwellenwert mit maximaler Varianz bietet beste Binarisierung

Otsu-Verfahren - Beispiel I

 I 

bimodales Histogramm

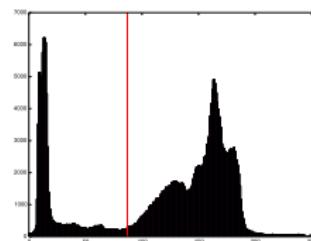
 $I < 80$

Vorher: Manuelle Auswahl des Schwellenwertes anhand des Histogramms.

Otsu-Verfahren - Beispiel II



I



bimodales Histogramm



$I < 87$

Nachher: Das Otsu-Verfahren wählt selbstständig 87 als Schwellenwert aus.

Otsu-Verfahren macht nur bei bimodalem Histogramm Sinn!

Binarisieren mit Farbe

Problem

Die Relationen $<$ und $>$ sind nicht eindeutig auf 3D-Vektoren ($[R, G, B]$) definiert.

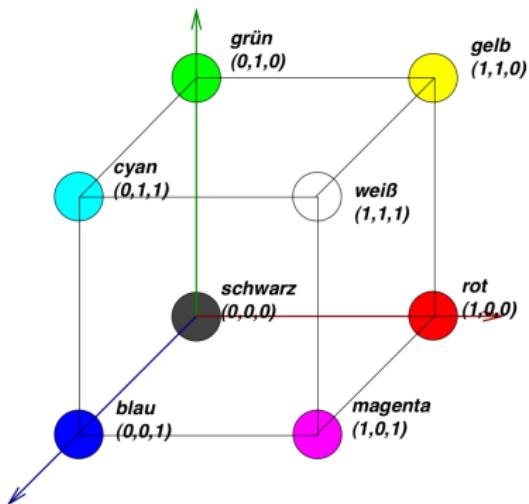
Lösungen

- 1 Bild in Graustufen umrechnen \Rightarrow Farbinformationen gehen verloren
- 2 Menge von RGB-Farbwerten definieren und nach Mitgliedschaft in dieser Menge S filtern

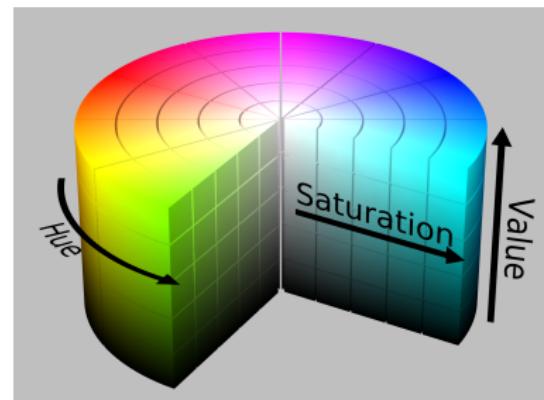
$$I_{result}(x, y) = \begin{cases} 1 & \text{if } I(x, y) \in S \\ 0 & \text{else} \end{cases}$$

- 3 Farbraum wechseln! \rightarrow HSV

HSV-Farbraum



RGB-Einheitswürfel

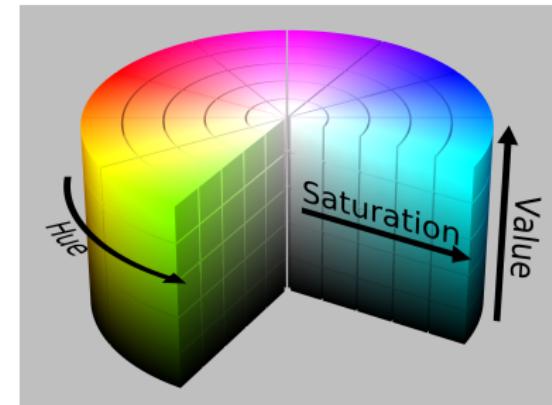


HSV-Zylinder¹

¹https://en.wikipedia.org/wiki/File:HSV_color_solid_cylinder_alpha_lowgamma.png

HSV-Farbraum

- Farben werden über Farbton, Sättigung und Helligkeit beschrieben
- der Farbton wird als Winkel im Farbkreis angegeben
- die Entfernung von der mittleren Achse gibt die Sättigung an
- die Höhe im Zylinder die Helligkeit
- intuitivere Farbbezeichnung
- erlaubt einfachen Vergleich von Farbtönen



HSV-Zylinder¹

¹https://en.wikipedia.org/wiki/File:HSV_color_solid_cylinder_alpha_lowgamma.png

Binarisieren - Reales Beispiel



Ausgangsbild



mask = (img>40) * (img<60)

Binarisieren - Reales Beispiel



Ausgangsbild



mask = $(img > 40) * (img < 60)$

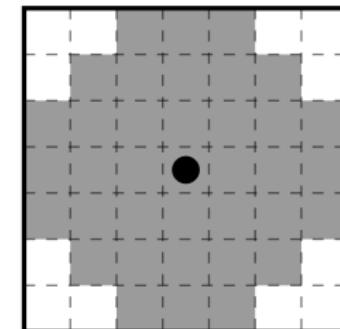
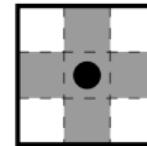
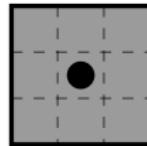
Da ist noch ziemlich viel 'Rauschen' im Hintergrund. Was nun?

Morphologische Operatoren

Morphologie: die Lehre der Form oder Gestalt

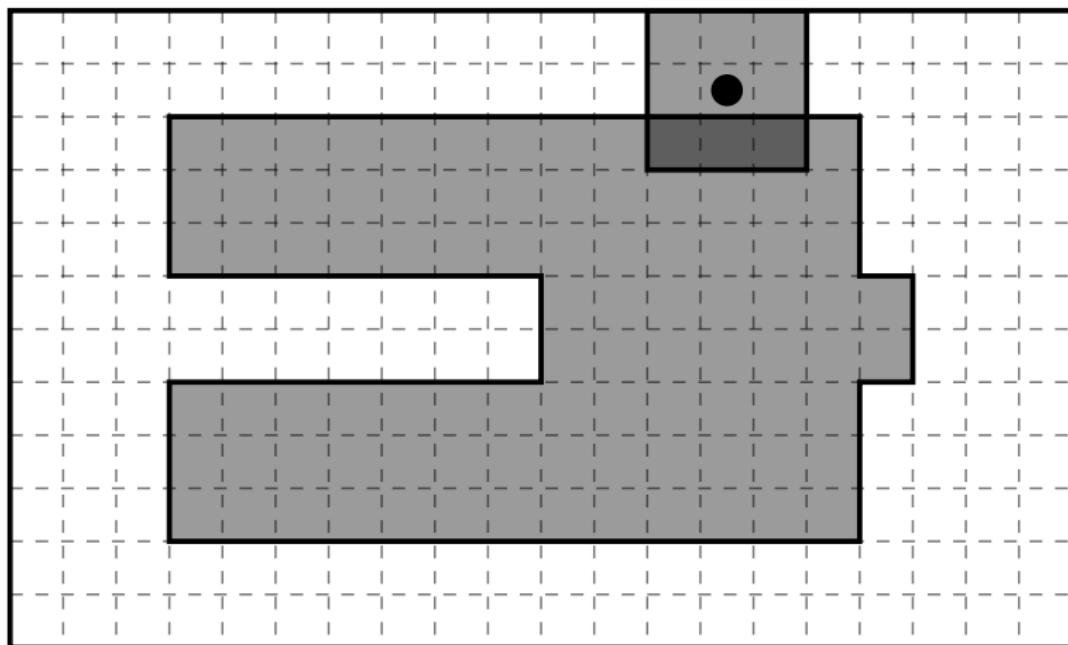
Prinzip

Der Vordergrund eines Binärbildes wird mit einem Strukturelement umfahren und dabei die Mengenoperationen Vereinigung oder Differenz auf den Vordergrund und das Strukturelement angewendet.



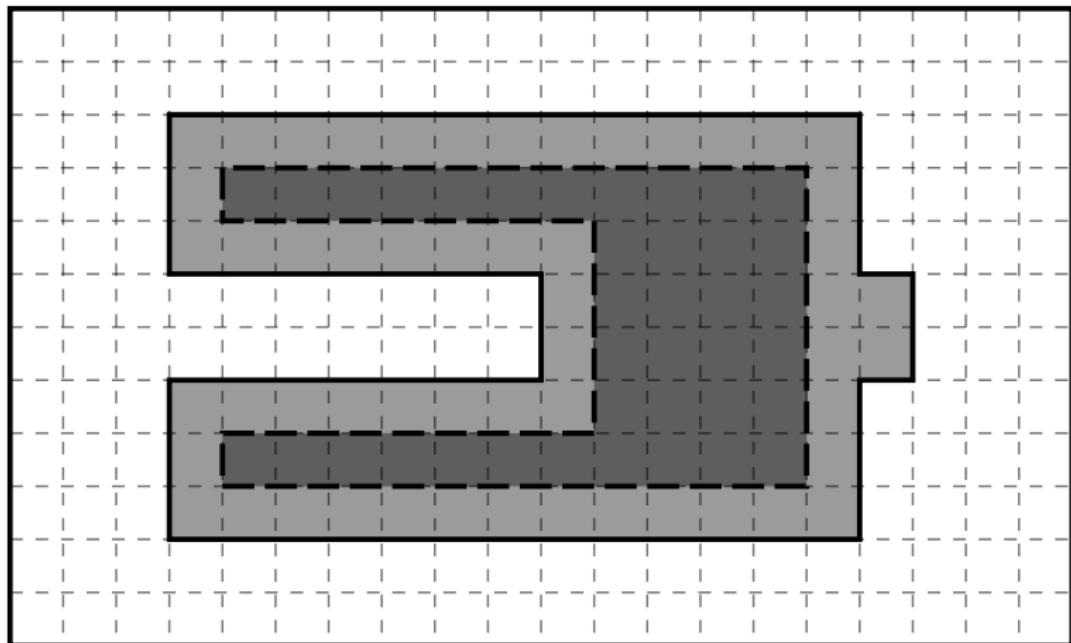
Strukturelemente

Erosion



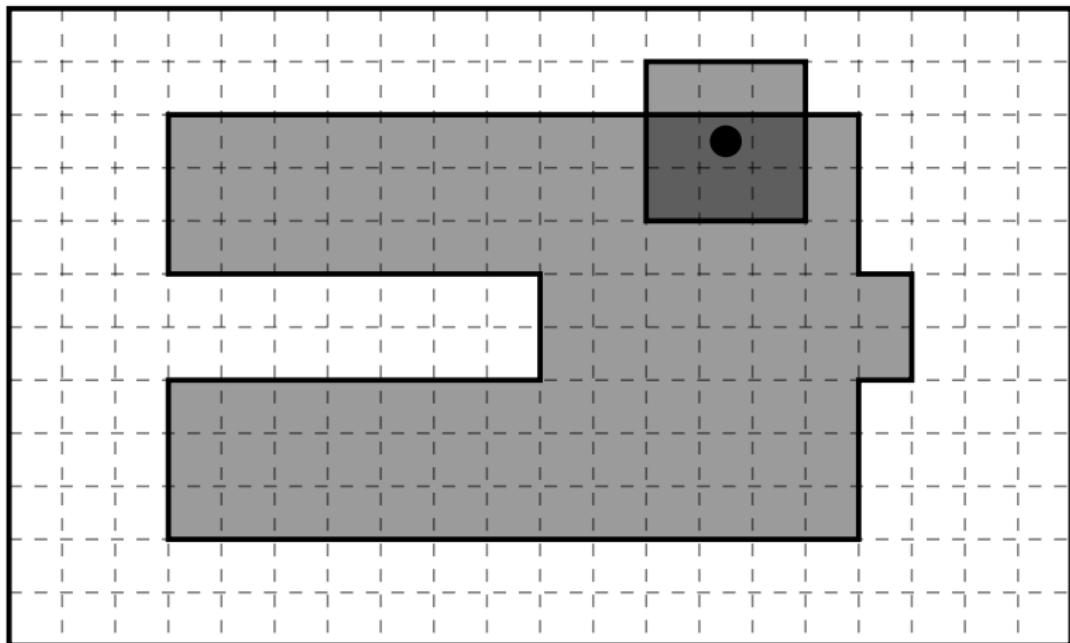
$A \ominus S$: alles, was das Strukturelement überstreicht, abknabbern

Erosion



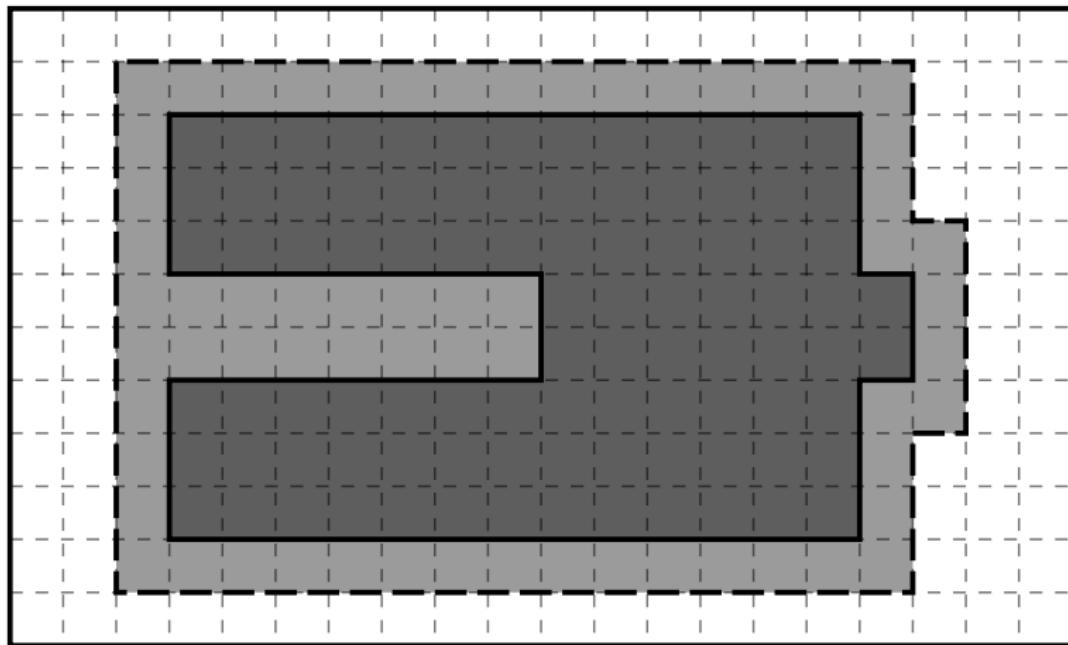
$A \ominus S$: alles, was das Strukturelement überstreicht, abknabbern

Dilatation



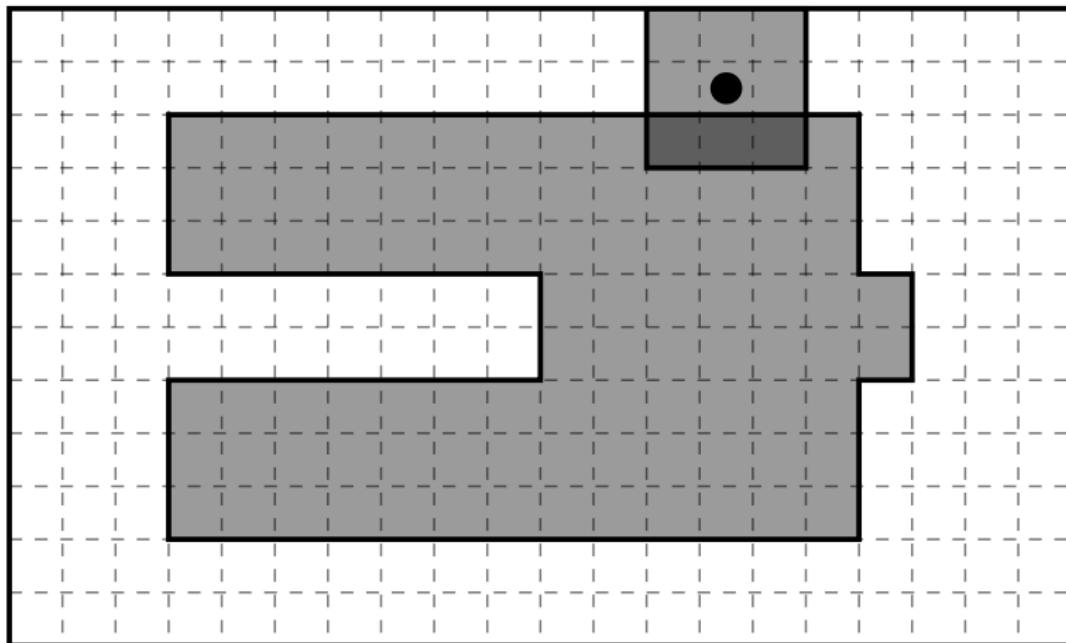
$A \oplus S$: alles, was das Strukturelement überstreicht, hinzufügen

Dilatation



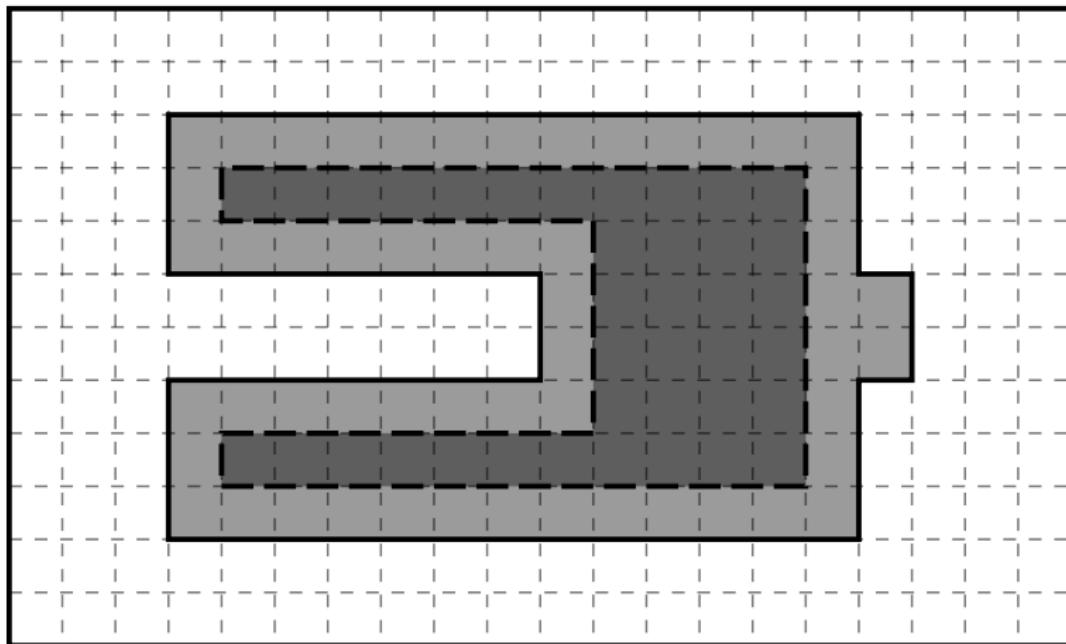
$A \oplus S$: alles, was das Strukturelement überstreicht, hinzufügen

Opening



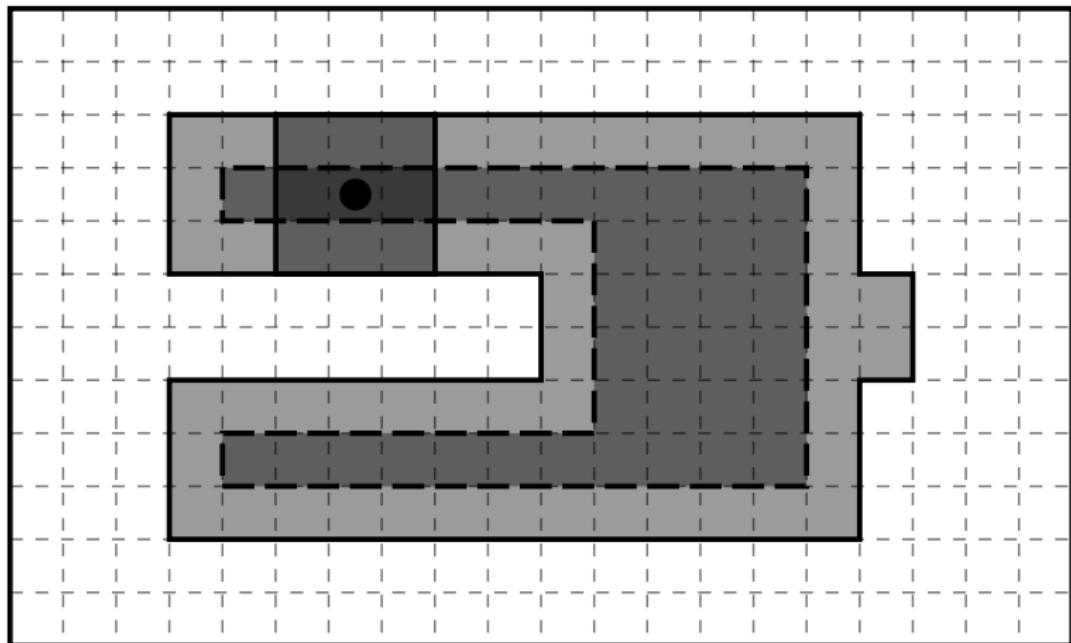
$$A \circ S = (A \ominus S) \oplus S: \text{erst Erosion, dann Dilatation}$$

Opening



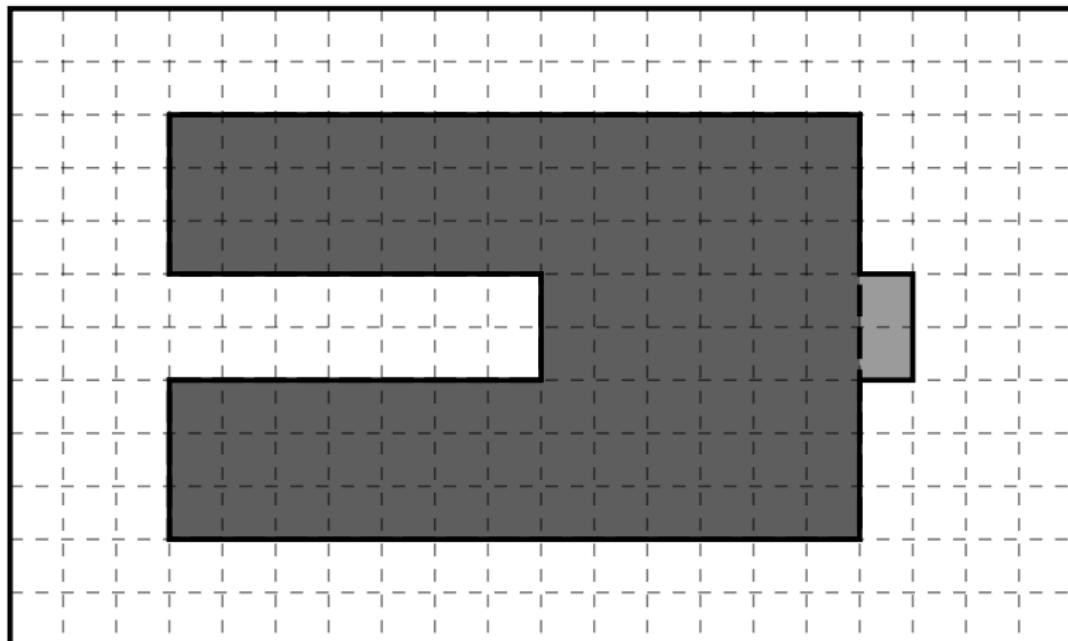
$$A \circ S = (A \ominus S) \oplus S: \text{erst Erosion, dann Dilatation}$$

Opening



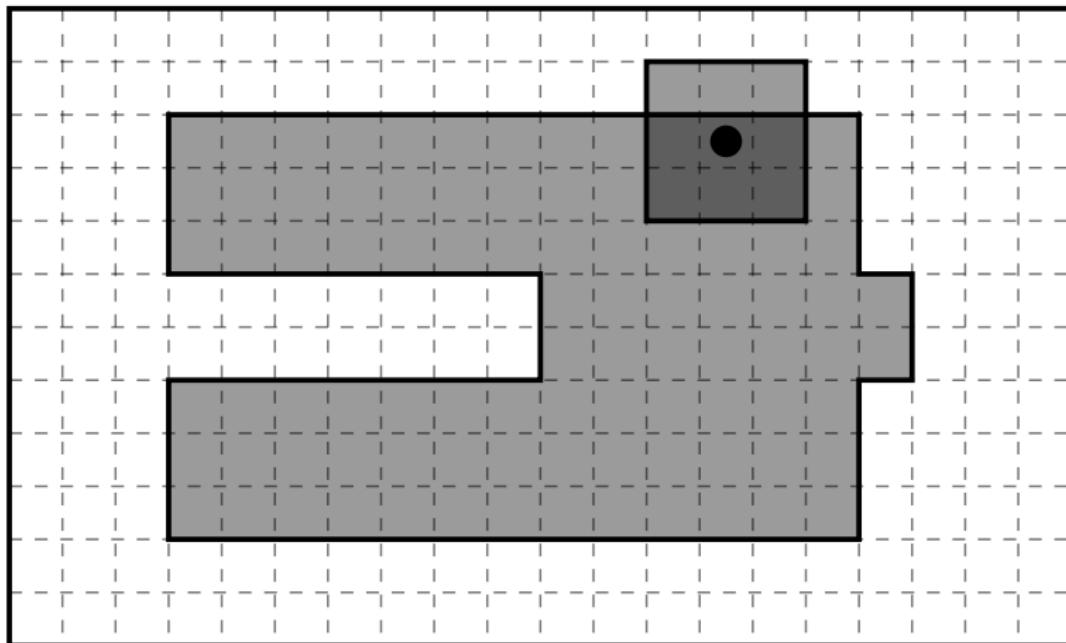
$A \circ S = (A \ominus S) \oplus S$: erst Erosion, dann Dilatation

Opening



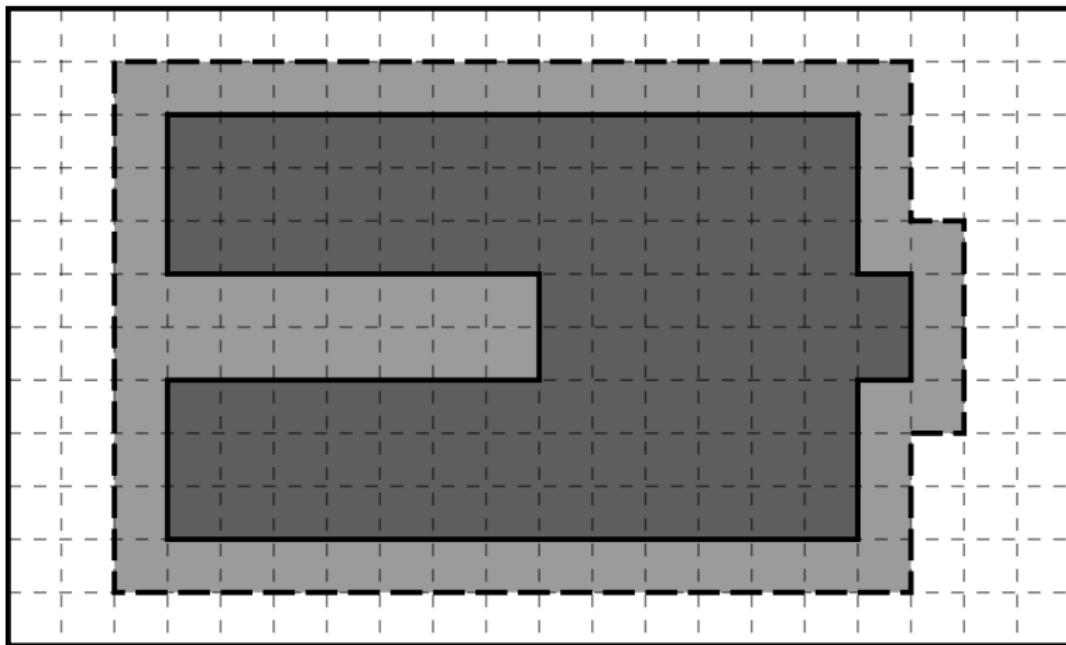
$$A \circ S = (A \ominus S) \oplus S: \text{erst Erosion, dann Dilatation}$$

Closing



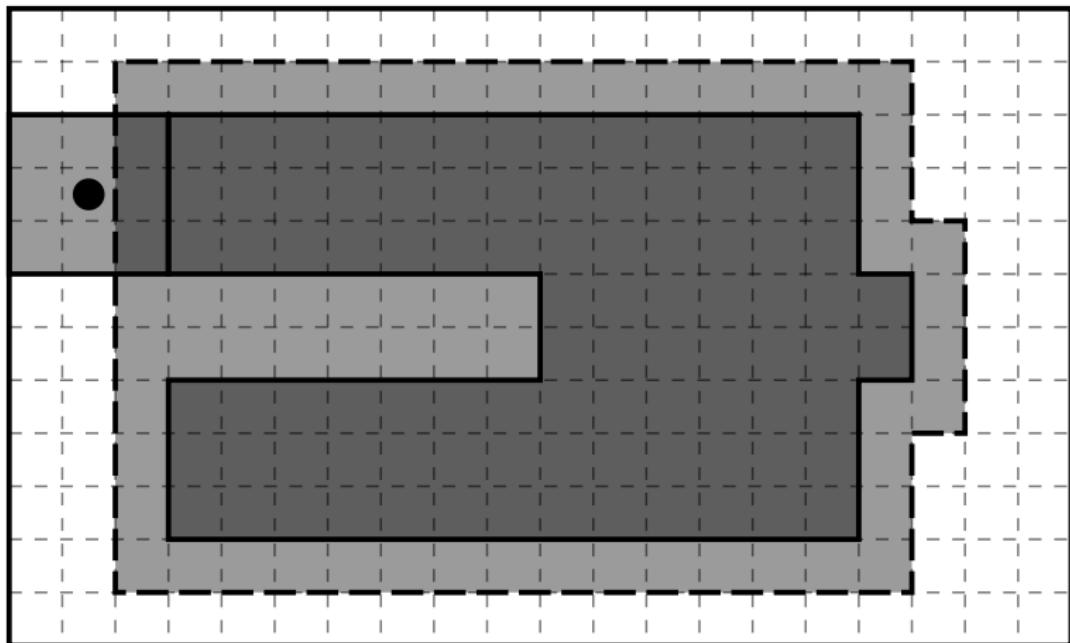
$$A \bullet S = (A \oplus S) \ominus S: \text{erst Dilatation, dann Erosion}$$

Closing



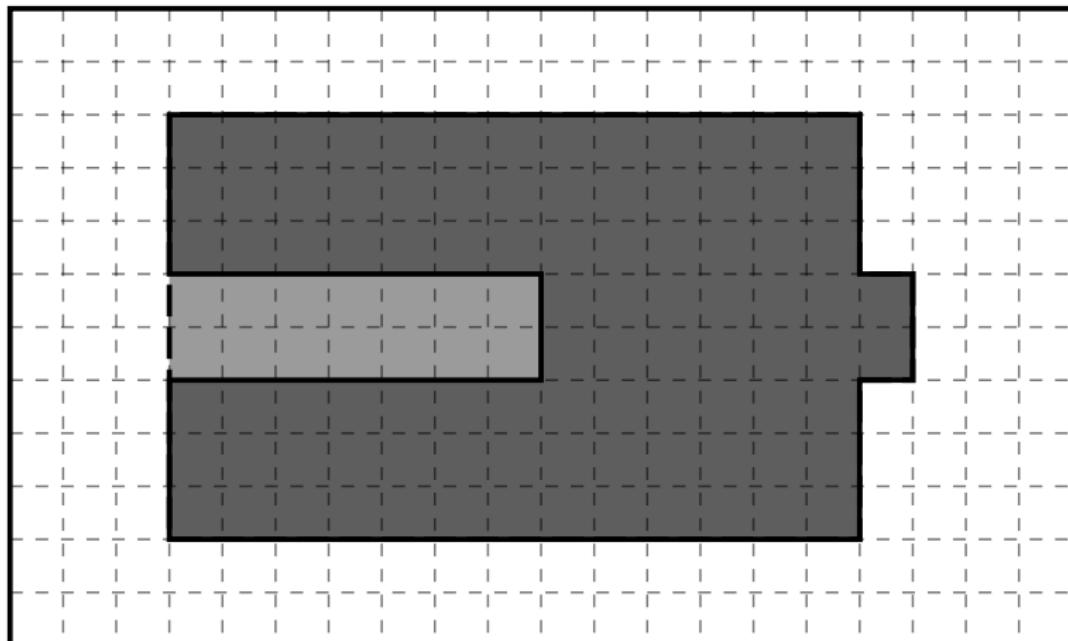
$$A \bullet S = (A \oplus S) \ominus S: \text{erst Dilatation, dann Erosion}$$

Closing



$$A \bullet S = (A \oplus S) \ominus S: \text{erst Dilatation, dann Erosion}$$

Closing



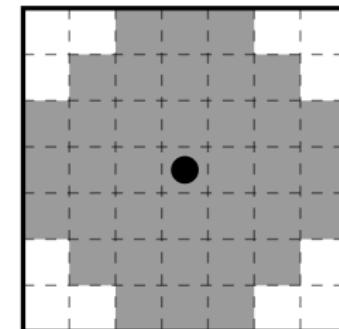
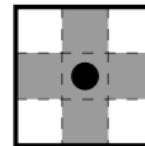
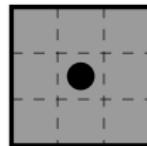
$$A \bullet S = (A \oplus S) \ominus S: \text{erst Dilatation, dann Erosion}$$

Morphologische Operatoren

Morphologie: die Lehre der Form oder Gestalt

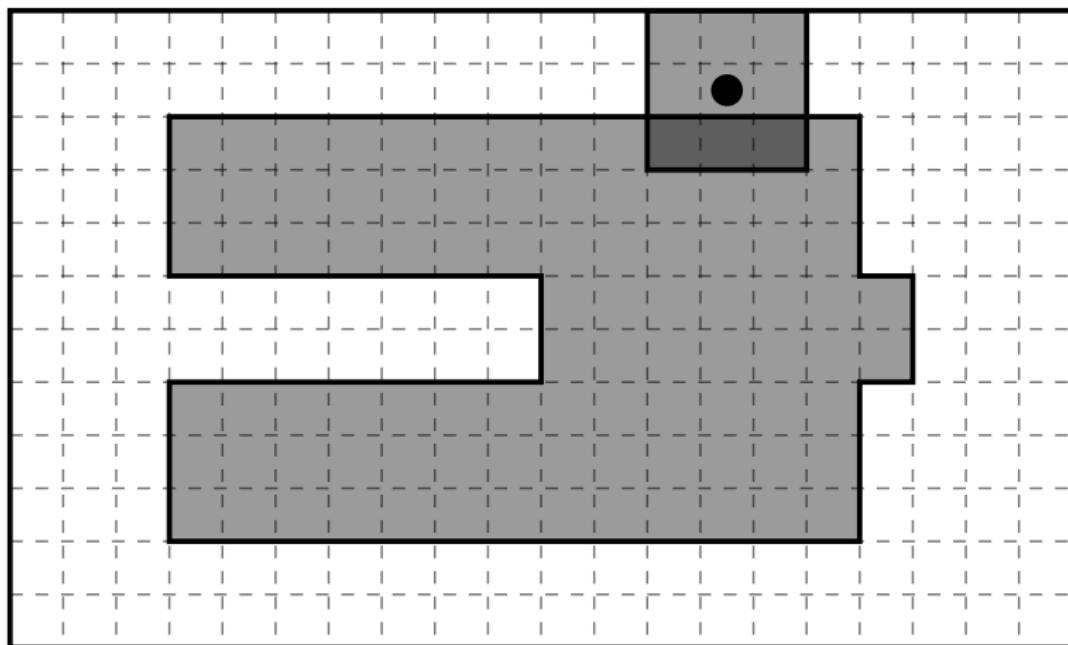
Prinzip

Der Vordergrund eines Binärbildes wird mit einem Strukturelement umfahren und dabei die Mengenoperationen Vereinigung oder Differenz auf den Vordergrund und das Strukturelement angewendet.



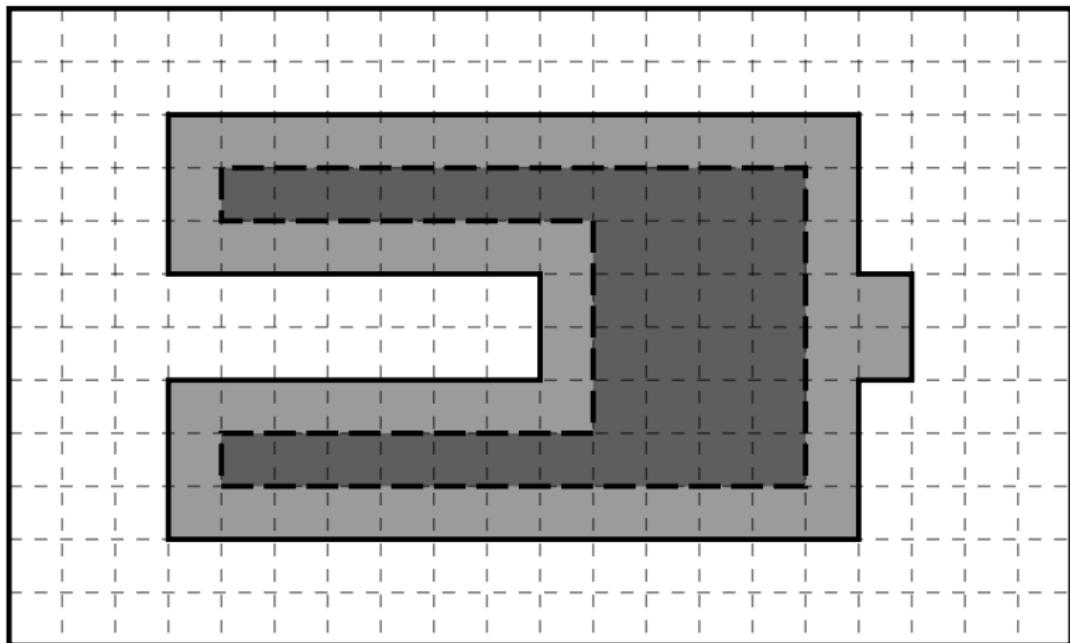
Strukturelemente

Erosion



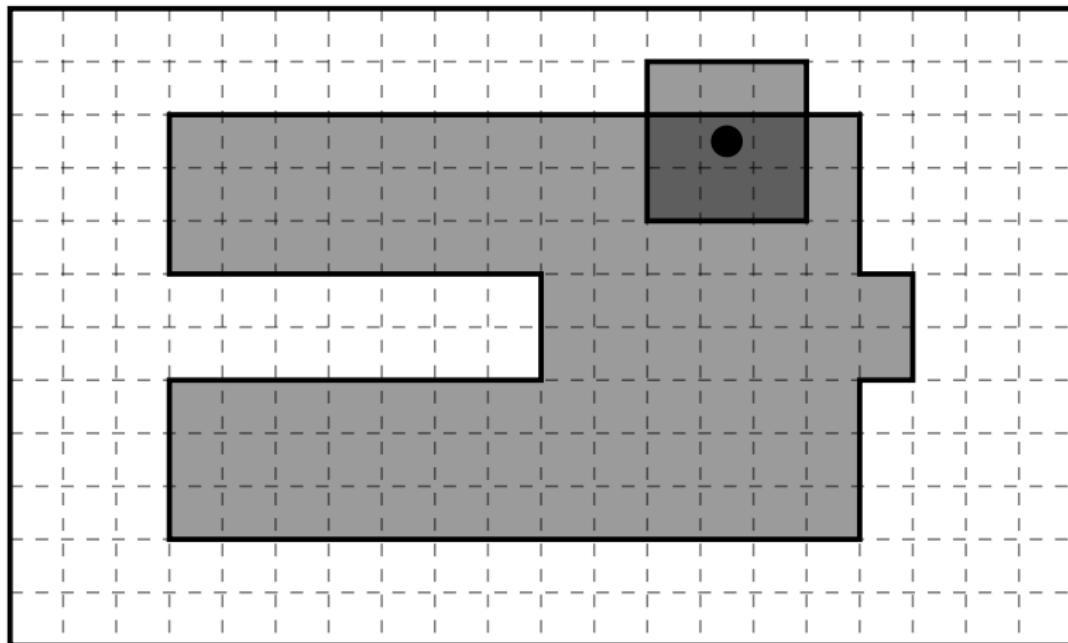
$A \ominus S$: alles, was das Strukturelement überstreicht, abknabbern

Erosion



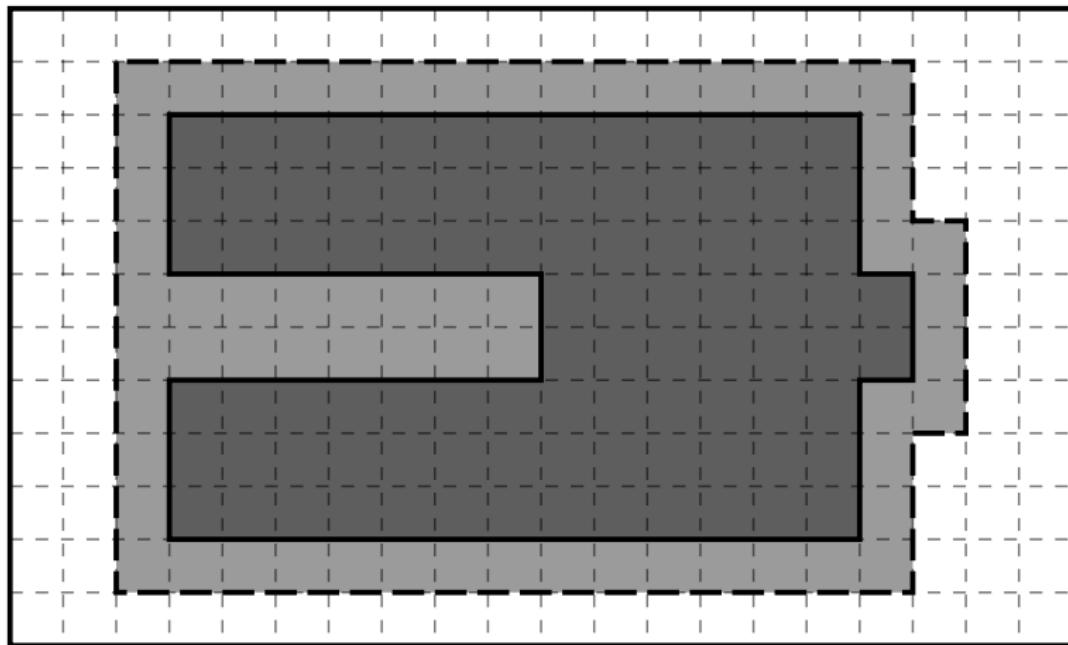
$A \ominus S$: alles, was das Strukturelement überstreicht, abknabbern

Dilatation



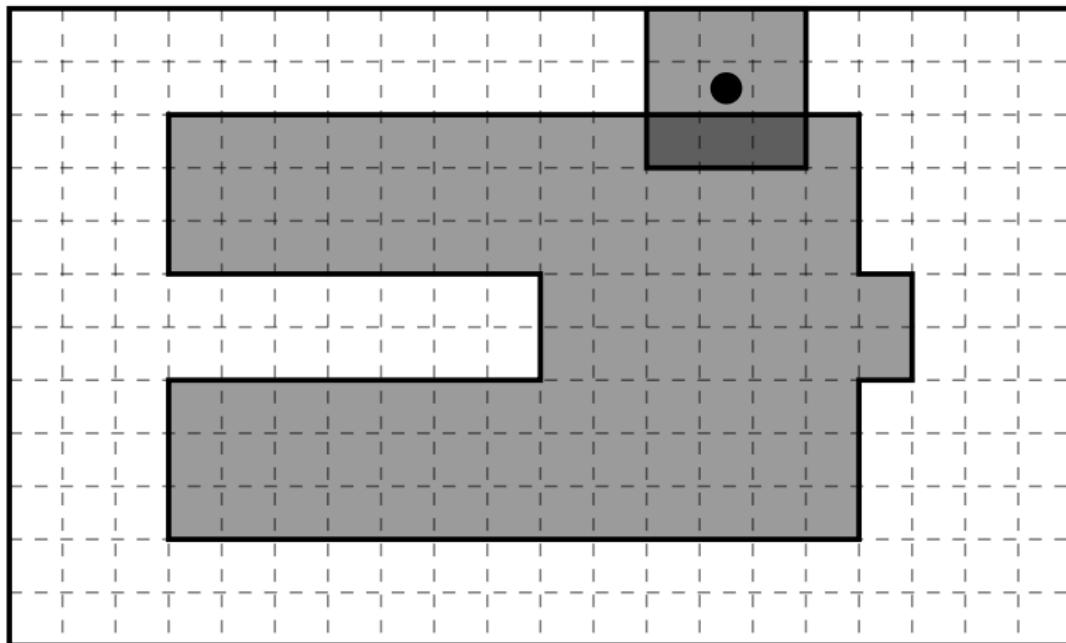
$A \oplus S$: alles, was das Strukturelement überstreicht, hinzufügen

Dilatation



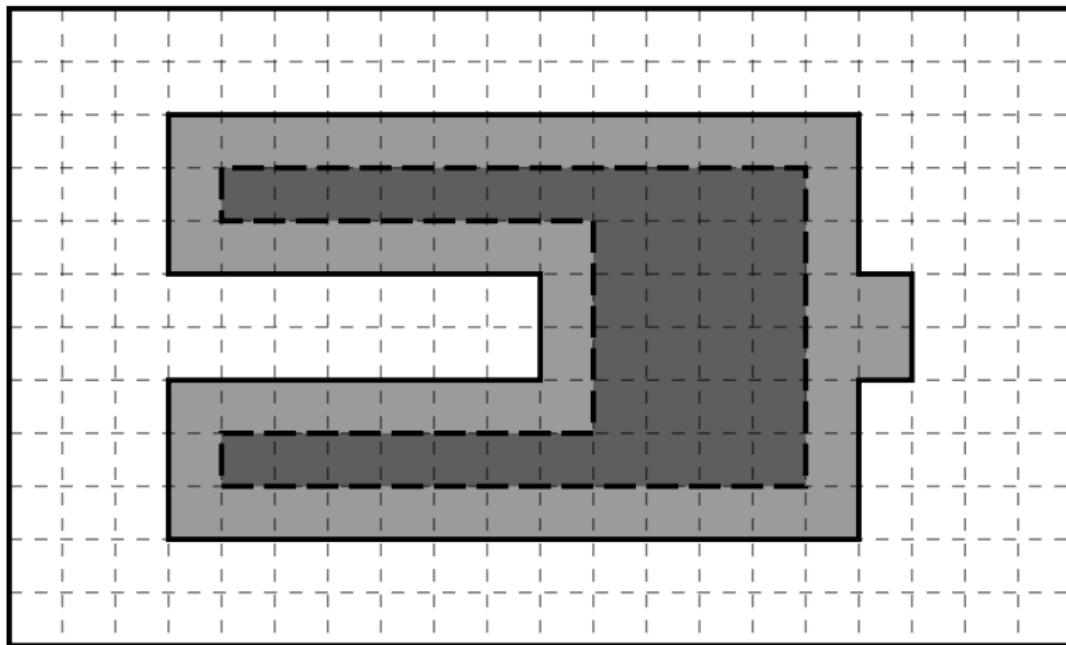
$A \oplus S$: alles, was das Strukturelement überstreicht, hinzufügen

Opening



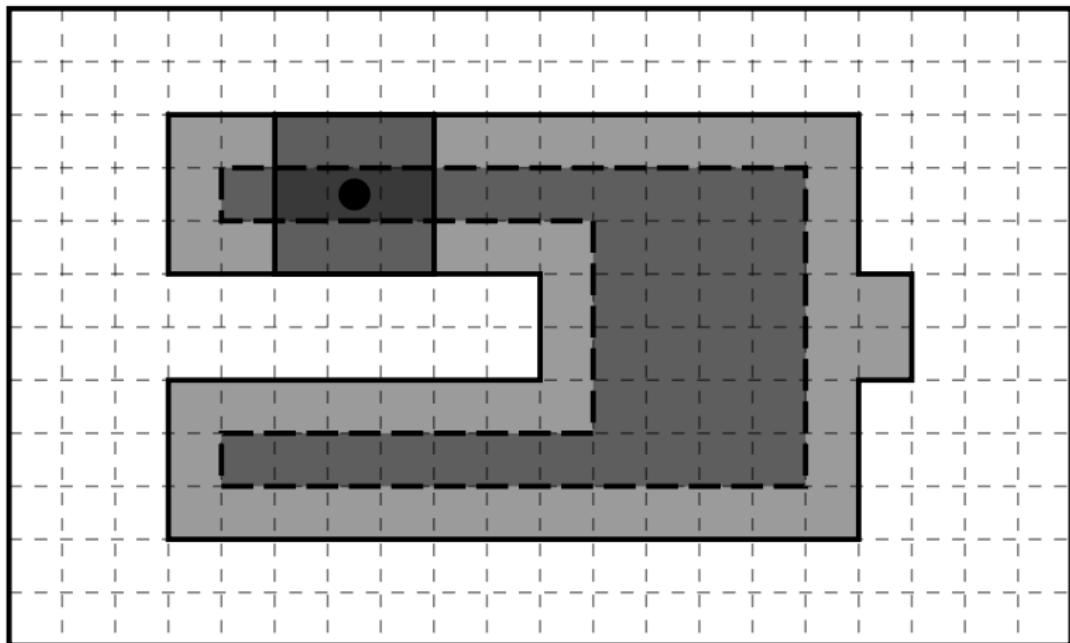
$$A \circ S = (A \ominus S) \oplus S: \text{erst Erosion, dann Dilatation}$$

Opening



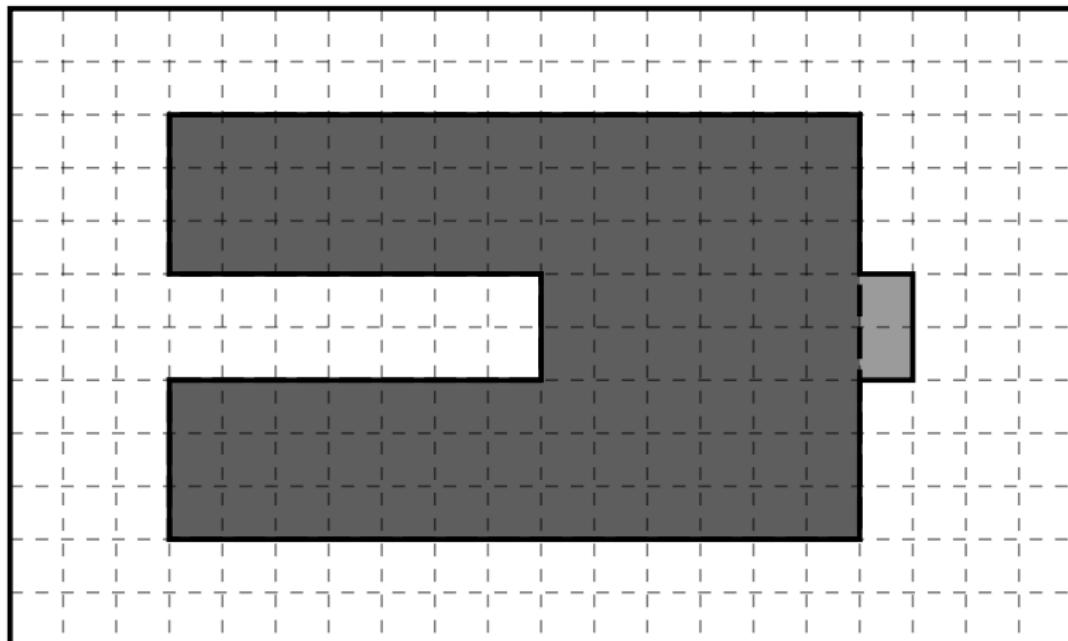
$$A \circ S = (A \ominus S) \oplus S: \text{erst Erosion, dann Dilatation}$$

Opening



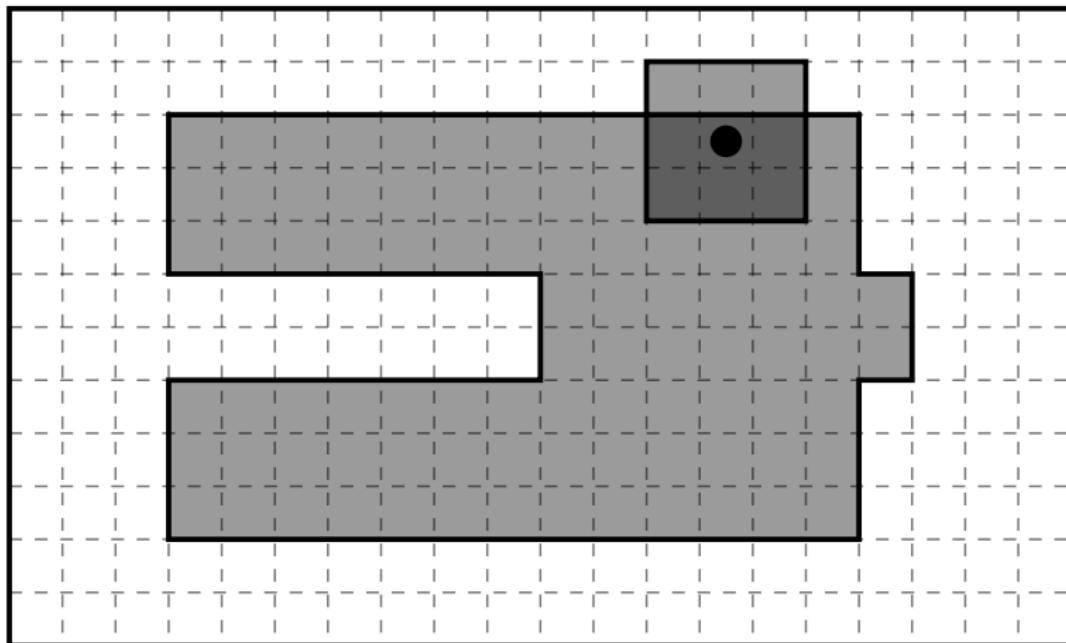
$$A \circ S = (A \ominus S) \oplus S: \text{erst Erosion, dann Dilatation}$$

Opening



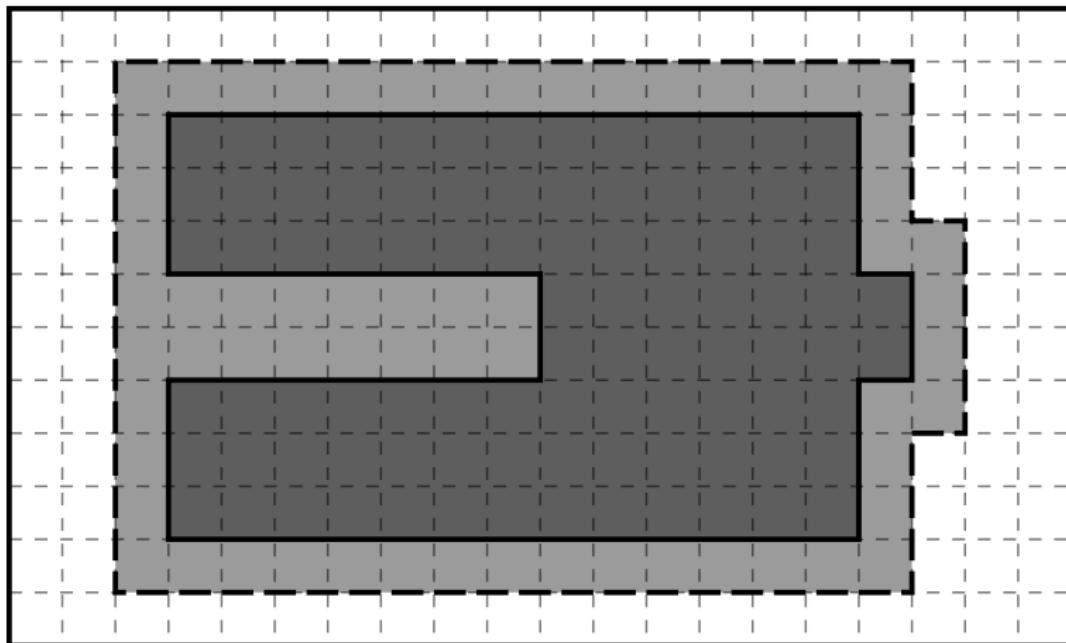
$$A \circ S = (A \ominus S) \oplus S: \text{erst Erosion, dann Dilatation}$$

Closing



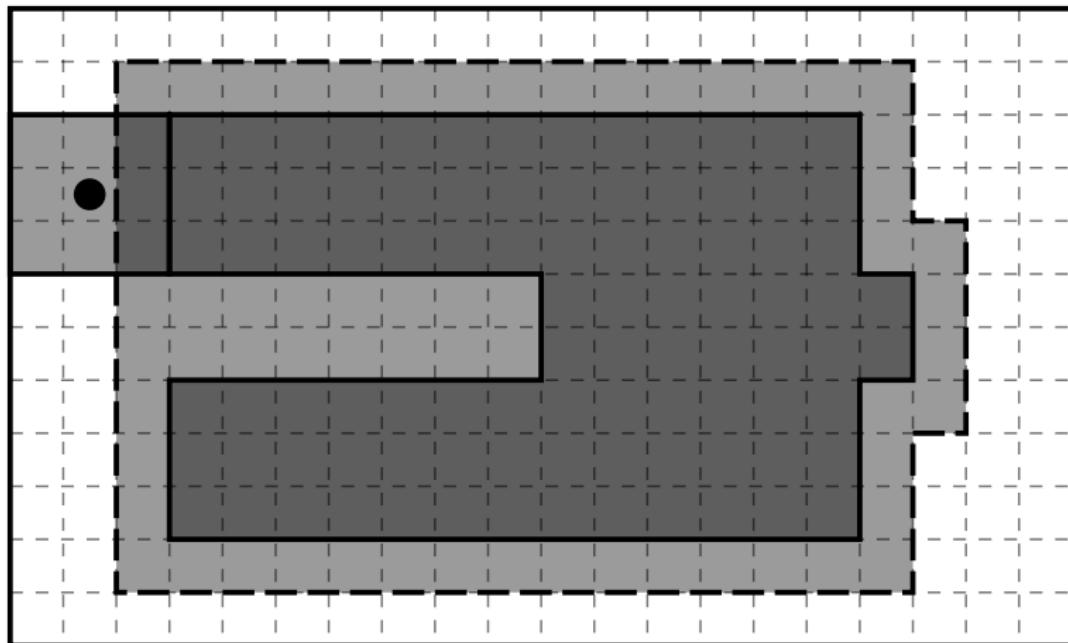
$$A \bullet S = (A \oplus S) \ominus S: \text{erst Dilatation, dann Erosion}$$

Closing



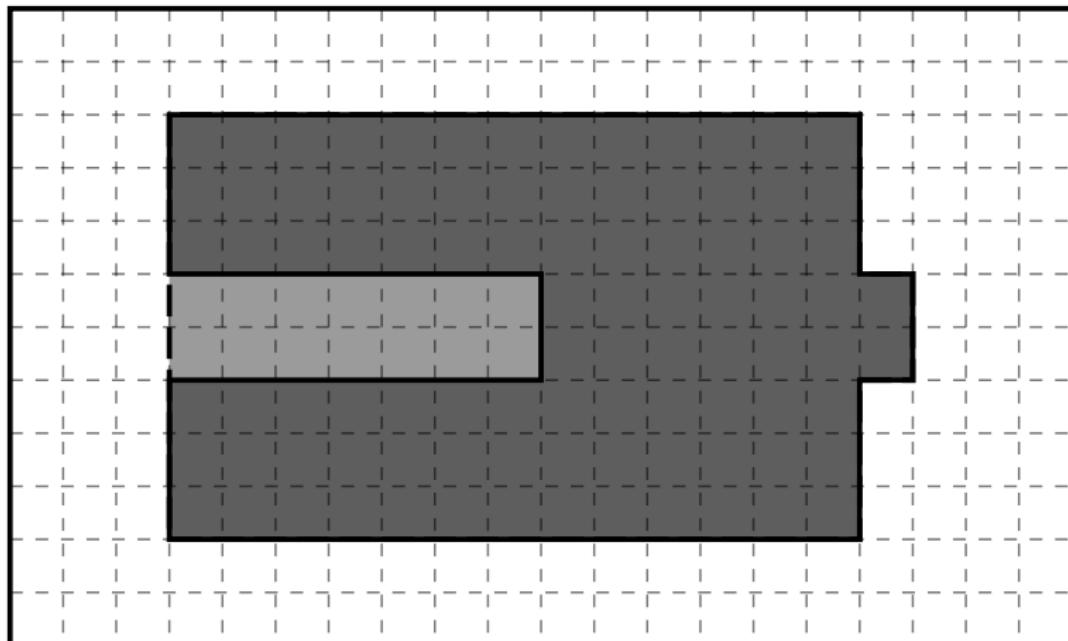
$$A \bullet S = (A \oplus S) \ominus S: \text{erst Dilatation, dann Erosion}$$

Closing



$$A \bullet S = (A \oplus S) \ominus S: \text{erst Dilatation, dann Erosion}$$

Closing



$$A \bullet S = (A \oplus S) \ominus S: \text{erst Dilatation, dann Erosion}$$

Mögliche Lösung



```
mask = opening(mask, disk(5)).astype(np.uint8)
```

Was sind Zusammenhangskomponenten?

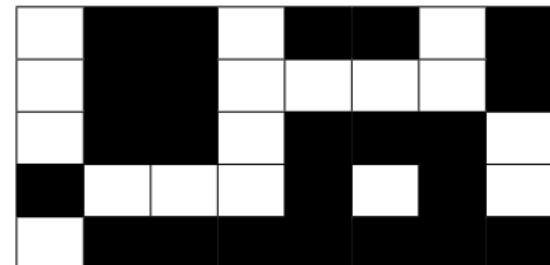
- ein Binärbild besteht aus mehreren Einzelteilen bzw. Inseln
- jedes dieser Einzelteile wird Zusammenhangskomponente genannt
- durch ein Connected Components Labeling kann man jedem dieser Einzelteile eine eindeutige Nummer/Graustufe/Farbe/... zuordnen
- Binärbild → Labelbild



Was sind Zusammenhangskomponenten?

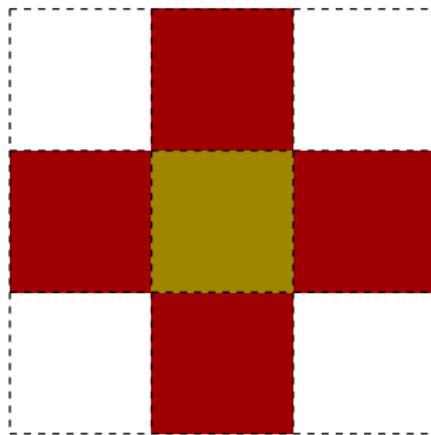
- ein Binärbild besteht aus mehreren Einzelteilen bzw. Inseln
- jedes dieser Einzelteile wird Zusammenhangskomponente genannt
- durch ein Connected Components Labeling kann man jedem dieser Einzelteile eine eindeutige Nummer/Graustufe/Farbe/... zuordnen
- Binärbild → Labelbild

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

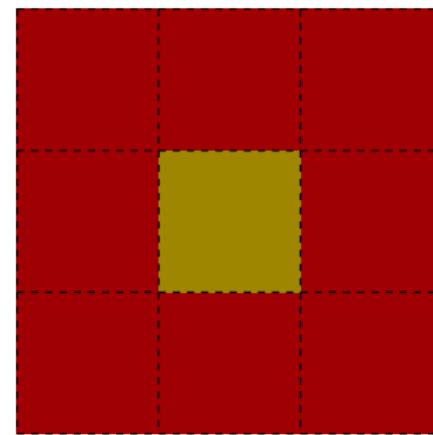


Wie viele Zusammenhangskomponenten gibt es hier?

Alles eine Frage der Nachbarschaft



4er-Nachbarschaft
oder
4er-Zusammenhang



8er-Nachbarschaft
oder
8er-Zusammenhang

Nach dem Labeling

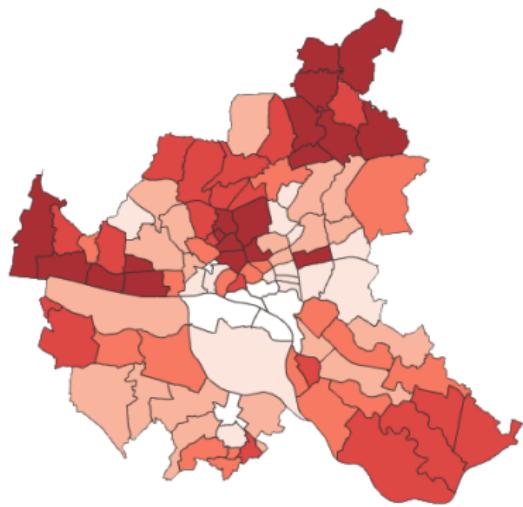


größte Komponente



Originalbild ausgeschnitten mit
der Bounding Box der größte
Komponente

Probleme mit globalem Schwellenwert



Es können Löcher im Objekt entstehen.

Region Growing

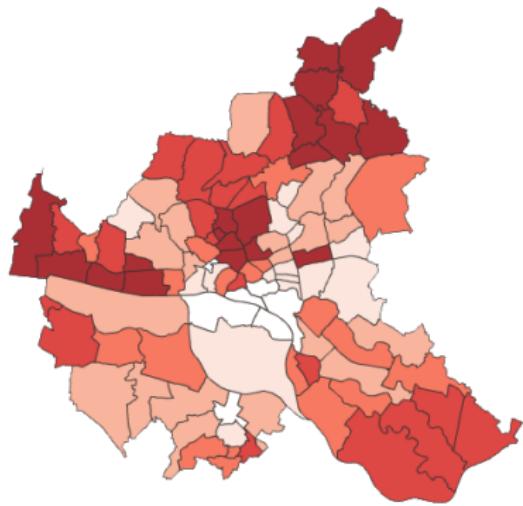
Problem

- Schwellenwert-basierte Verfahren binarisieren das Bild global
- Teile eines Objekts können dem Hintergrund ähneln
- Objekt/Region bekommt Löcher

Eine Lösung: Region Growing

- ein oder mehrere Pixel als Saatpunkte für den Hintergrund aussuchen
- Nachbarn der Saatpunkte ermitteln und auf Ähnlichkeit testen
- falls ähnlich, werden die Nachbarn Teil des Hintergrunds
- rekursiv weiter vorgehen (praktisch: besser iterativ mit Queue)

Ergebnis mit Region Growing



Es können Löcher im Objekt entstehen.

Übersicht

1 Binarisierung

2 Bildtransformation

3 Merkmale

scikit-image

- Eure erste richtige Bildverarbeitungsbibliothek!
- speziell konzipiert, um mit NumPy und SciPy zusammen zu arbeiten
- hat eine Schwesternbibliothek fürs maschinelle Lernen: scikit-learn

Wertvolle Links...

- Dokumentation
- viele gute und anschauliche Beispiele

Für alle Funktionen, die es nicht in scikit-image gibt und die man auch nicht selbst bauen will, nimmt man OpenCV.

Warum Bilder transformieren?

- Bildinhalte werden manchmal achsenparallel oder in bestimmten Größen benötigt
- ermöglicht einfacheren Vergleich (invariant gegen Rotation und Skalierung)
- Bilder der gleichen Szene können so exakt aufeinander gelegt werden (Registrierung)

Transformationen

- Rotation
- Skalierung

Problem

Durch die Transformationen fallen die Pixel auf Koordinaten außerhalb von $\mathbb{N}^2 \rightarrow$ Interpolation nötig.

Bilder skalieren

- Bild wird um gegebenen Faktor skaliert (meist uniform, `rescale`) oder auf eine bestimmte Größe (`resize`)
- bei einer Vergrößerung werden keine Informationen gewonnen
- bei einer Verkleinerung gehen Informationen verloren
- je nach Interpolationsmethode können Bilder beim Vergrößern pixelig werden



NN-Interpolation



Spline-Interpolation

Bilder rotieren

- Bild wird um seinen Mittelpunkt rotiert
- Ergebnisbild ist zumeist größer
- je nach Interpolationsmethode können Bilder beim Rotieren pixelig werden

```
>>> rotate(image, 90, order=0)
```



NN-Interpolation



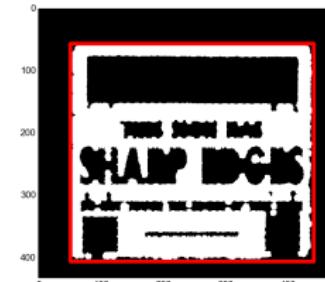
Spline-Interpolation

Bounding Box

Bounding Box

Eine Bounding Box ist das kleinste ein Objekt vollständig umhüllende Rechteck.

- gut geeignet als Grundlage zum Ausschneiden/Betrachten von Objekten → Kasten um das Objekt herum
- die Größe der Bounding Box ist (meist) abhängig von der Rotation des Objekts → minimale Bounding Box bedeutet Ausrichtung des Objekts an den Bildachsen



Bounding Box in Python

Gegeben ein Binärbild (auch Maske genannt):

Möglichkeit 1 - np.where

- die Koordinaten aller Vordergrundpixel mit `np.where` auswählen
- Minimum und Maximum der x- sowie der y-Koordinaten ermitteln

Möglichkeit 2

Bounding Box in Python

Gegeben ein Binärbild (auch Maske genannt):

Möglichkeit 1

Möglichkeit 2 - skimage.measure.regionprops

- stellt vorher sicher, dass in der Maske nur Zahlen vom Typ np.int stehen
- Rückgabe als Liste von Props-Objekten → hier nur eins, da Nullen in der Maske ignoriert werden
- Props-Objekte haben Felder für verschiedene Eigenschaften

```
>>> mask = mask.astype(np.int)
>>> props = regionprops(mask)[0]
>>> props.bbox#xMin, yMin, xMax, yMax
```

Übersicht

1 Binarisierung

2 Bildtransformation

3 Merkmale

Was wir heute machen wollen...



- neuer Datensatz mit Objekten auf homogenem Hintergrund und drei Klassen
- Bilder anhand der Objekte klassifizieren
- Objekte durch Binarisierung ausschneiden
- Merkmale an Objekten oder den Bounding Boxen berechnen

Was für Merkmale von Objekten könnten wir gebrauchen?

Fläche

Anzahl der Pixel der Region.

Seitenverhältnis

Quotient aus Höhe und Breite der an den Bildachsen ausgerichteten Region. (**nicht rotationsinvariant** → Exzentrizität)

Exzentrizität

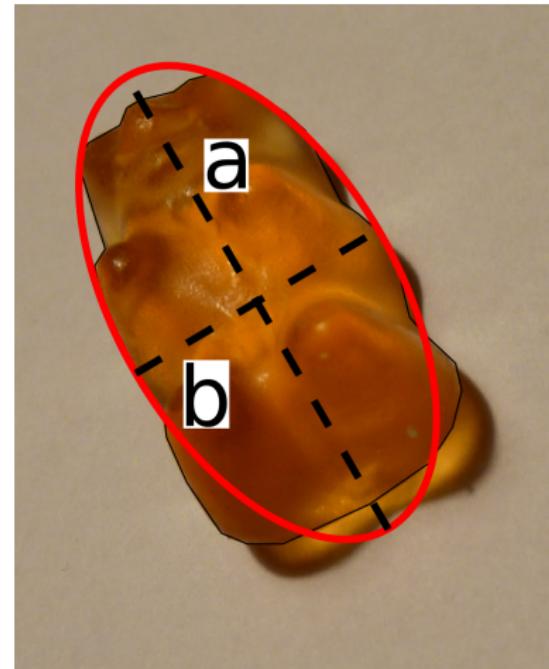
Die Exzentrizität e einer Region lässt sich aus der Länge der beiden Achsen a und b einer Ellipse berechnen, welche die selbe Verteilung an Pixeln aufweist wie die Region:

$$e = \sqrt{1 - \frac{b^2}{a^2}}$$

Beispiele - (quasi) rotationsinvariant

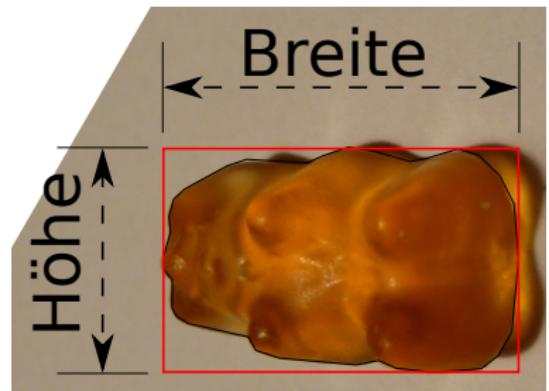
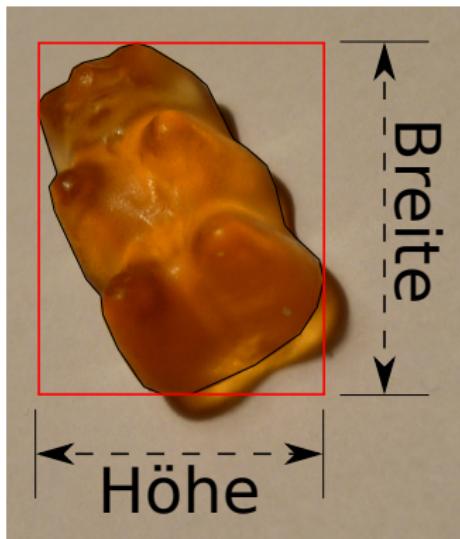


Fläche



$$\text{Exzentrizität } e = \sqrt{1 - \frac{b^2}{a^2}}$$

Beispiele - nicht rotationsinvariant



Statistiken über die Verteilung der Region

Wie ist die Region räumlich innerhalb der Bounding Box verteilt?

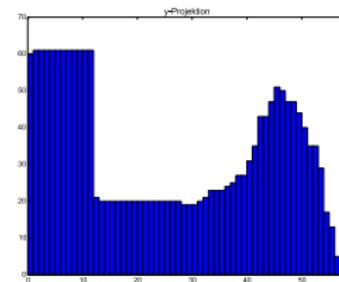
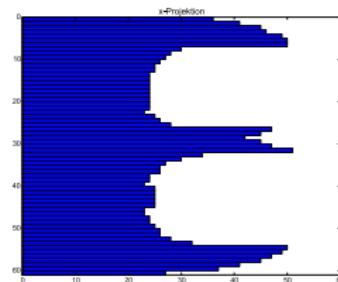
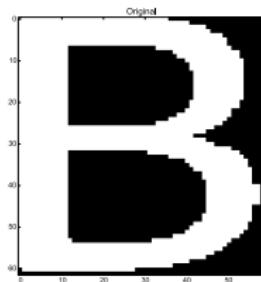
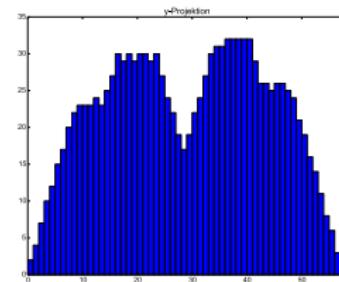
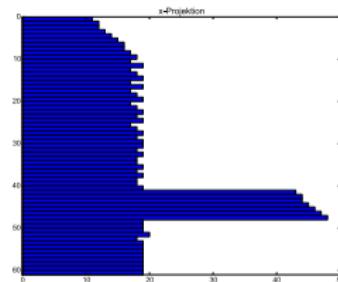
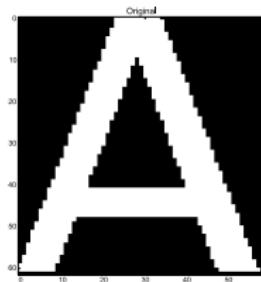
→ Region und Referenzregion haben die gleiche Größe.

→ Es wird nur auf den Vordergründen binarierter Bilder gearbeitet.

- Mittelwert der x- und y-Koordinaten (Schwerpunkt)
- Standardabweichung der x- und y-Koordinaten
- Kombinationen
- Histogramme über x- und y-Koordinaten (Projektionen)

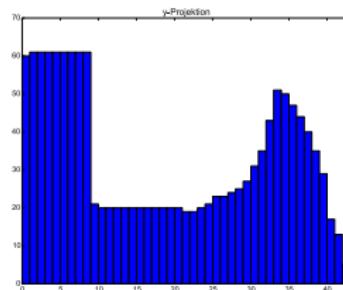
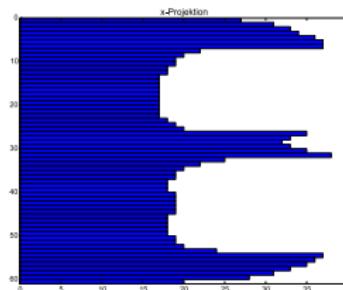
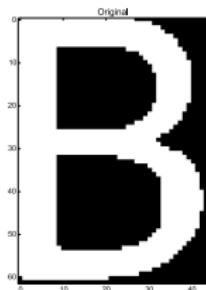
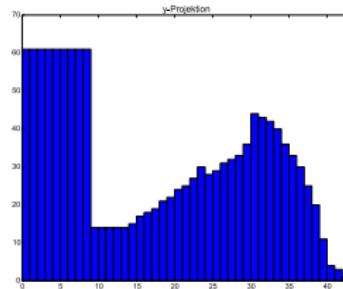
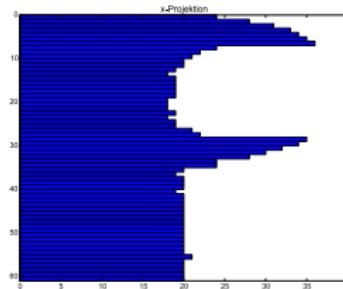
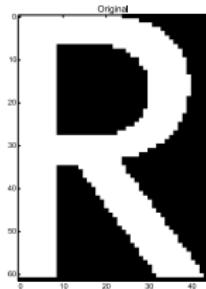
x- und y-Koordinaten werden hier jeweils unabhängig voneinander betrachtet!

Statistiken über die Verteilung der Region - Beispiel I



Mittlere Koordinaten A: (33.16, 28.57), B: (29.79, 18.28)

Statistiken über die Verteilung der Region - Beispiel II



Mittlere Koordinaten R: (28.48, 17.21), B: (29.79, 18.28)

Praxistipps Python - Alle Bilder eines Ordners einlesen

```
>>> import glob  
>>> glob.glob('./dataset/*.png')  
 ['./dataset/1.png', './dataset/2.png', './dataset/3.  
 png', './dataset/4.png']  
>>> glob.glob('./dataset/*.png')[0].split('/')[-1].  
 split('.') [0]  
'1'
```

- die Sortierung der Dateien kann unterschiedlich sein
- Labels können in den Dateinamen integriert werden
- der Dateiname lässt sich aus dem Pfad über zwei
split-Operationen extrahieren ('/','.')

Praxistipps Python - Einbinden von Bibliotheken

- die meisten Bibliotheken bestehen aus mehreren Paketen, daher sollte man nur das gewünschte Paket oder sogar nur die gewünschte Funktion importieren
- die Bibliothek scikit-image nennt sich kurz skimage; scikit-learn nennt sich kurz sklearn
- Bevor ihr eine Bibliotheksfunktion nutzt, schaut in die Doku und achtet auf die richtige Version!!!

```
>>> from skimage import transform  
>>> transform.rotate(img,20) #oder:  
>>> from skimage.transform import rotate  
>>> rotate(img,20)
```