# Computergraphik

## Vorlesung im Wintersemester 2013/14
## Übung zu Kapitel 2: Farben & Filter

Prof. Dr.-Ing. Carsten Dachsbacher
Lehrstuhl für Computergrafik
Karlsruher Institut für Technologie

# Organisatorisches

- Ab und zu auf die Homepage schauen!
- Wir können nicht für jedes System Makefiles pflegen

- Übung von 14:45h auf 17:30h in HS -102 verschoben

# Besprechung ÜB 1

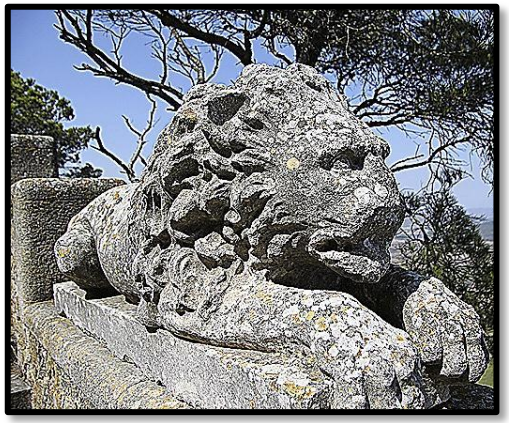
- Das Problem des Rundens
  - Wir wollten alle Abgaben einfach und konsistent halten, indem wir das Runden übernehmen
  - Schaut bitte auch in den existierenden Code, um herauszufinden, welche Funktionalität schon implementiert ist

# Bildmanipulationen

## Wir betrachten 3 Arten von Manipulationen

- Operationen im *Farbraum* („pro Pixel“)
- *lineare Filter*: gewichtete Summe benachbarter Pixel-Werte
- *morphologische Filter*: strukturverändernde Operationen

Helligkeit (Farbraum)



Scharfzeichnung (linearer Filter)

Entfernen kleiner Strukturen (morphologischer Filter)

# Bildmanipulationen

## Bildoperationen im Farbraum

- Änderung der Farbe eines Pixels basierend nur auf dem aktuellen Farbwert an dieser Stelle
  - keine Betrachtung anderer Pixel

- Formell: Hintereinanderausführung von Funktionen
  - Bildfunktion $f: \mathbb{R}^2 \rightarrow \mathbf{C}$ weist einer Position $\mathbf{u} \in \mathbb{R}^2$ im Bild eine Farbe zu: $f(\mathbf{u}) \in \mathbf{C}$
  - die Bildoperation ist eine Funktion $g: \mathbf{C} \rightarrow \mathbf{C}'$
  - das Bild wird jetzt durch die Funktion $g \circ f$ beschrieben

# Beispiel: Änderung der Helligkeit

- doppelte Helligkeit (hier: Funktion für ein Graustufenbild): $g(x) = 2x$

# Beispiel: Änderung des Kontrasts

- Kontraständerung (hier: Graustufenbild): $g(x) = c(x - 0.5) + 0.5$

$c < 1$

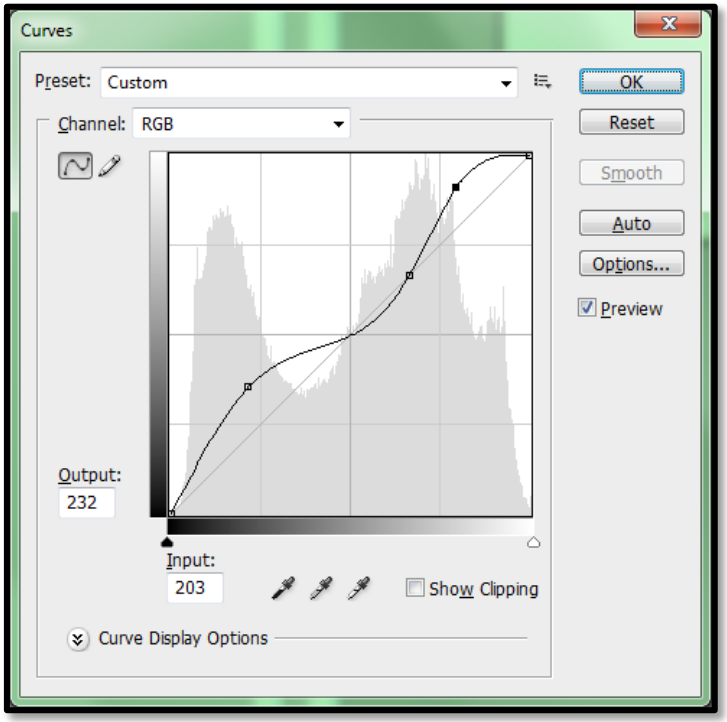
$c > 1$

# Beispiel: Desaturierung

- häufig verwendete Abbildung: $Y = 0.3R + 0.59G + 0.11B$

# Gradationskurven, Farbkurven

- visuelle/interaktive Manipulation der Funktion $g$
- präzise Kontrolle




# Operationen im Farb- *und* Bildraum

- die neue Farbe eines Pixels hängt ab von
  - seiner aktuellen Farbe
  - den Farbwerten der Pixelnachbarschaft

- viele Bildoperationen (sog. Filter) lassen sich durch **Faltung** ausdrücken
  - Bsp. Unschärfe, Schärfe, Kantendetektion, Embossing, …

# Faltung – lineare Filter

- das Bild ist eine Funktion $f$, die Farben zu Positionen zuordnet

- eine Faltung berechnet die „Überlappung" einer Funktion $f$ mit einer zweiten Funktion $g$, die (an der $y$-Achse gespiegelt) über $f$ geschoben wird

$$[f * g](t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$$

- $g$ ist die Filterfunktion
- die Faltung zweier Funktionen ist wieder eine Funktion

# Faltung

- Faltung zweier Rechteckfunktionen $f$ und $g$
- das Resultat der Faltung ist eine Dreiecksfunktion: der Flächeninhalt des Produkts von $f$ und $g$ für jede Verschiebung

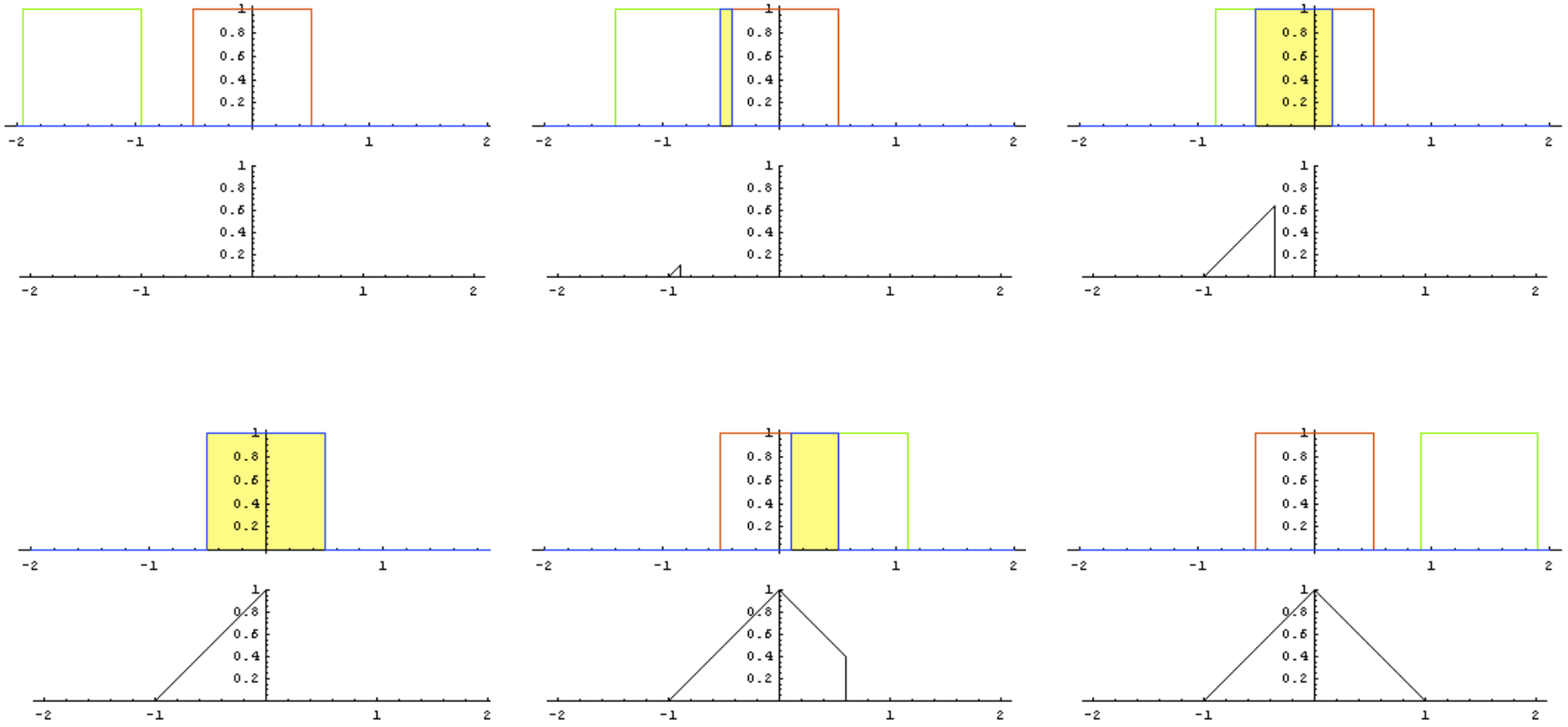


Bild: Wikipedia

# Diskrete Faltung

- wenn $f$ und $g$ diskret sind (z.B. die Funktionen nur an ganzzahligen Stellen definiert), dann ist die diskrete Faltung

$$[f * g](n) = \sum_{i=-\infty}^{\infty} f(i)g(n-i)$$

- Prinzip
  - die Filterfunktion („Kernel“) $g$ wird mittig um den n-ten Pixel ausgerichtet
  - Gewichtung jedes Pixels des Bildes gemäß des Werts von $g$ an dieser Stelle
  - Aufsummieren der gewichteten Farbwerte ergibt die neue Farbe des n-ten Pixels

# Faltung in 2D

- kontinuierlich

$$[f * g](s,t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\sigma,\tau) g(s-\sigma, t-\tau) d\sigma d\tau$$

- diskret

$$[f * g](m,n) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(i,j) g(m-i, n-j)$$

- oft sind Filter $g(.,.)$ nur in einem (kleinen) Intervall ungleich 0

$$[f * g](m,n) = \sum_{i=m-a}^{m+a} \sum_{j=n-b}^{n+b} f(i,j) g(m-i, n-j)$$

  - mit $-a \leq m-i \leq a$ und $-b \leq n-j \leq b$

# Bsp. diskrete Faltung in 2D

```
#define WIDTH  128
#define HEIGHT 128

// ein weniger häufiger Floating-Point Frame Buffer
float img[ WIDTH * HEIGHT ];

// Graustufen Bild in Floating Point (0.0 = schwarz, 1.0 = weiß)
float f[ WIDTH * HEIGHT ];

// diskrete Filterfunktion mit a=b=3
float g_table[ 7 * 7 ];
#define g( i, j ) g_table[ ((i)+3) + ((j)+3) * 7 ]
int a = 3, b = 3;

for ( n = 0; n < HEIGHT; n++ ) {
  for ( m = 0; m < WIDTH; m++ ) {

    float res = 0.0f;

    for ( int i = m - a; i <= m + a; i++ )
      for ( int j = n - b; j <= n + b; j++ )
        res += f[ i + j * WIDTH ] *
               g( m - i, n - j );

    img[ x + y * WIDTH ] = res;
}  }
```

Achtung:
in diesem Beispiel wird nicht auf Bereichsüberschreitungen geachtet!

# Bsp. diskrete Faltung in 2D

- Annahme: alle Werte außerhalb des Definitionsbereichs von $f$ sind 0
- der hier verwendeten Kernelmatrix sieht man nicht an, dass sie gespiegelt ist (siehe Definition diskrete Faltung)

$$[f * g](m,n) = \sum_{i=m-a}^{m+a} \sum_{j=n-b}^{n+b} f(i,j)g(m-i,n-j)$$

| | | |
|---|---|---|
| 2 | 3 | 1 |
| 0 | 5 | 1 |
| 1 | 0 | 8 |

$*$

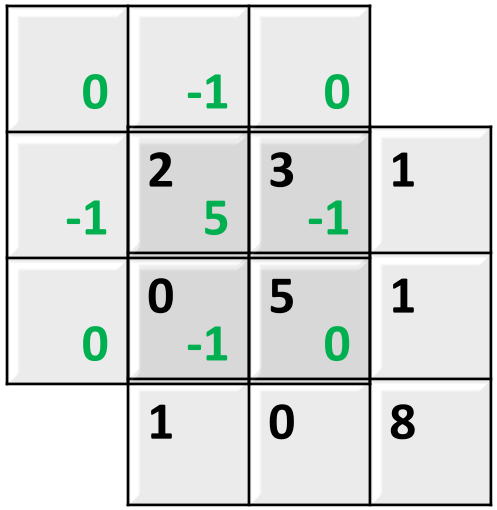
| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

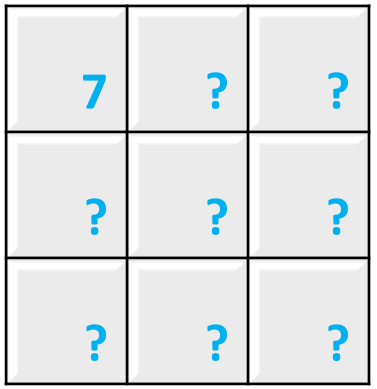
$=$

| | | |
|---|---|---|
| 7 | 7 | 1 |
| -8 | 21 | -9 |
| 5 | -14 | 39 |

# Bsp. diskrete Faltung in 2D

- Werte außerhalb des Bildes werden oft als 0 angenommen
  - man kann aber auch definieren, dass das Bild wiederholt wird, oder
  - man auf den jeweils nahsten Pixelwert zugreift



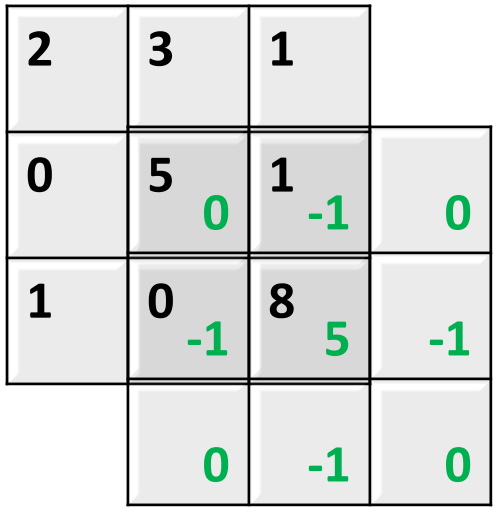



# Bsp. diskrete Faltung in 2D

| | | |
|---|---|---|
| 2 0 | 3 -1 | 1 0 |
| 0 -1 | 5 5 | 1 -1 |
| 1 0 | 0 -1 | 8 0 |

| | | |
|---|---|---|
| 7 | 7 | 1 |
| -8 | 21 | ? |
| ? | ? | ? |

# Bsp. diskrete Faltung in 2D




| | | |
|---|---|---|
| 7 | 7 | 1 |
| -8 | 21 | -9 |
| 5 | -14 | 39 |

# Filter: Unschärfe (Blur)

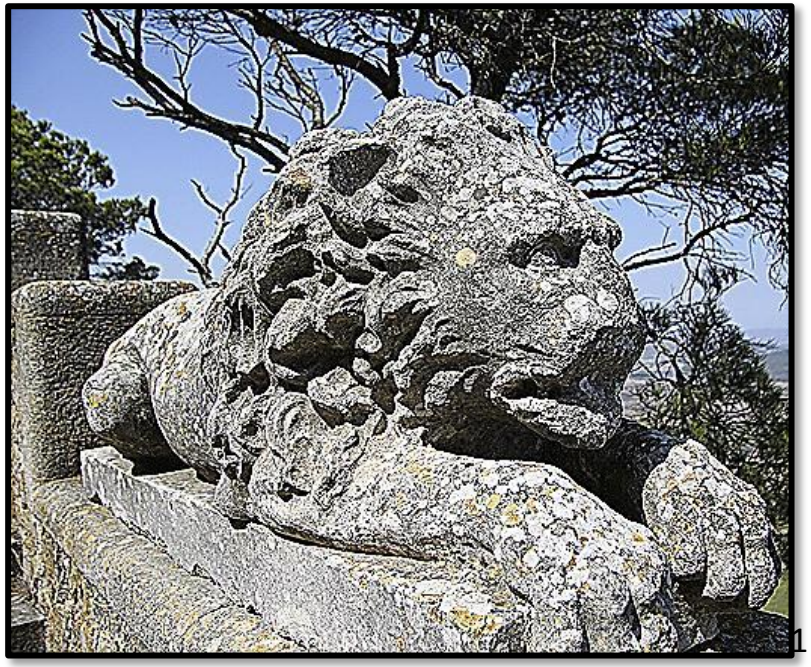
- normalisierter Kernel: die Einträge summieren sich zu 1
  - die Gesamthelligkeit des Bildes bleibt erhalten

$$* \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \cdot 1/9$$

# Filter: Schärfe (sharpen)

*

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |



# Filter: Kantendetektion

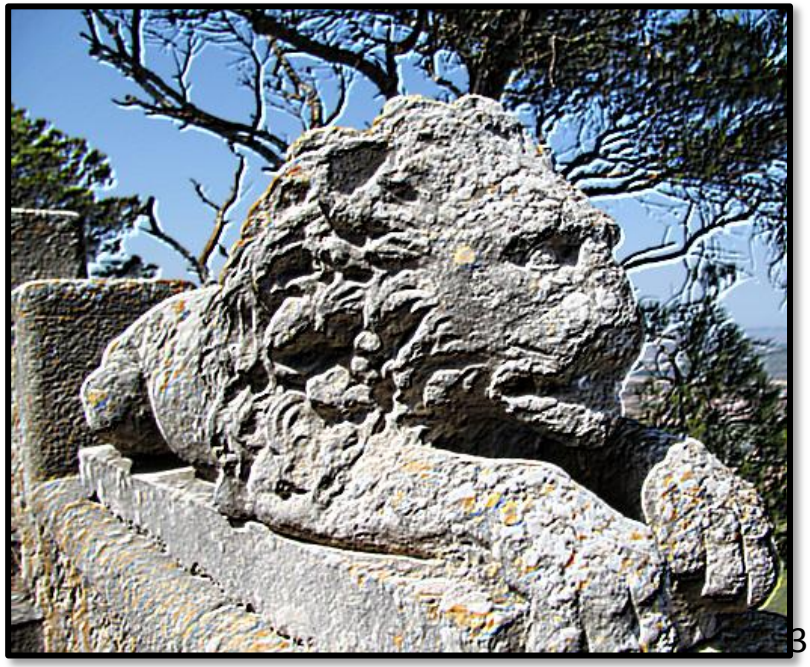
*

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

# Filter: Emboss

*

| | | |
|---|---|---|
| -1 | -1 | 0 |
| -1 | 1 | 1 |
| 0 | 1 | 1 |



# Morphologische Operationen

- strukturverändernde Operation (ein nicht-linearer Filter)
  - betrachte Menge der Nachbarpixel $A$ (typ. in einem Binärbild)
  - Strukturelement $B$
  - Operation: setze oder lösche Pixel

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

$B$

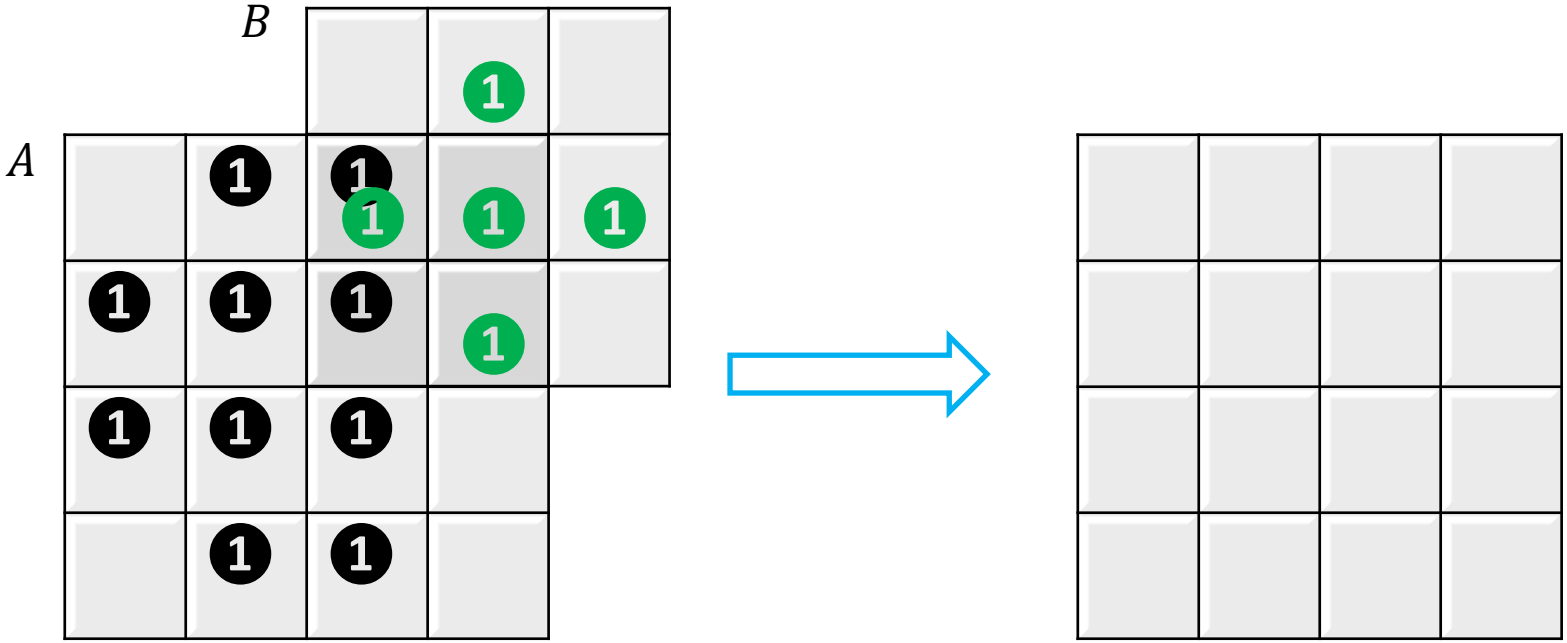
- **Dilatation** (Erweiterung)
  - Menge aller Pixel $z$, bei denen eine „1“ im Strukturelement mit mind. einer „1“ in $A$ überlappt

- **Erosion** (Abtragung)
  - Menge aller Pixel $z$, bei denen alle „1“ im Strukturelement mit einer „1“ in $A$ überlappen

# Dilatation

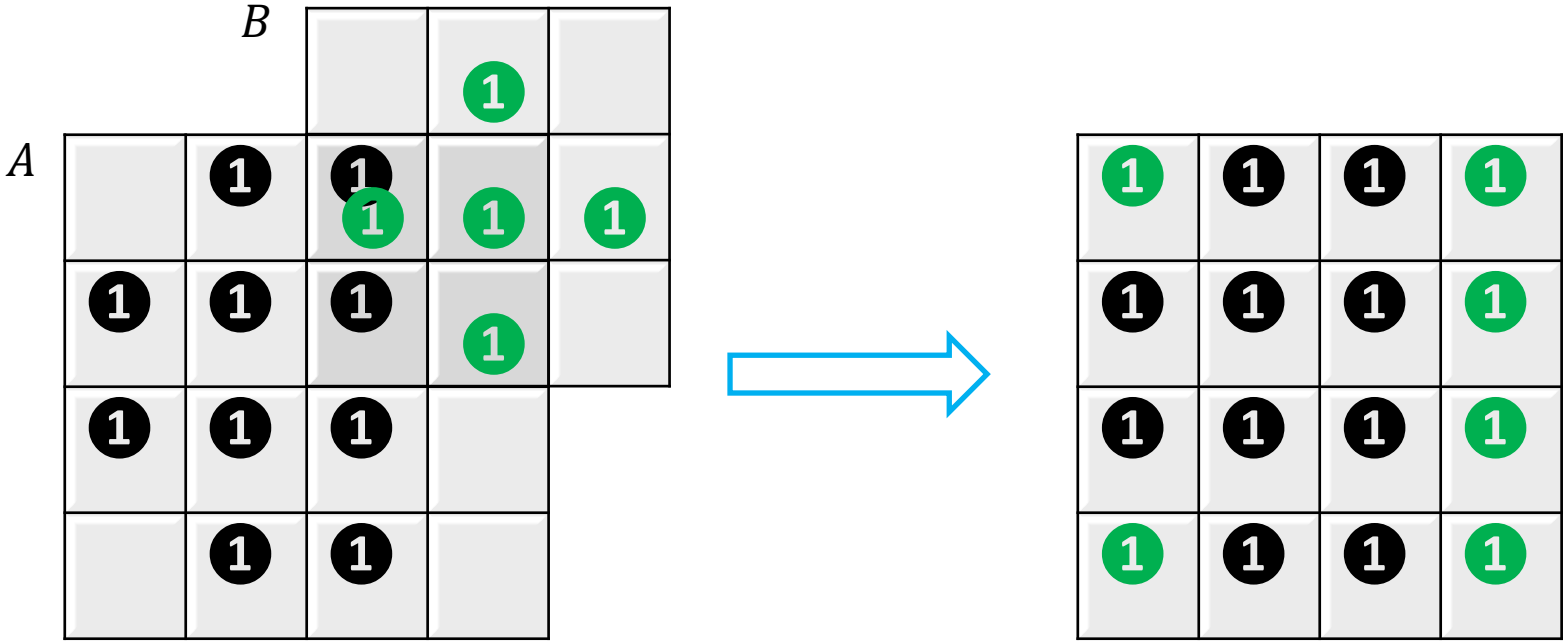
- Menge aller Pixel $z$, bei denen eine „1“ im Strukturelement mit mind. einer „1“ in $A$ überlappt

- Frage: Welche Pixel sind nach der Dilatation „1“?



# Dilatation

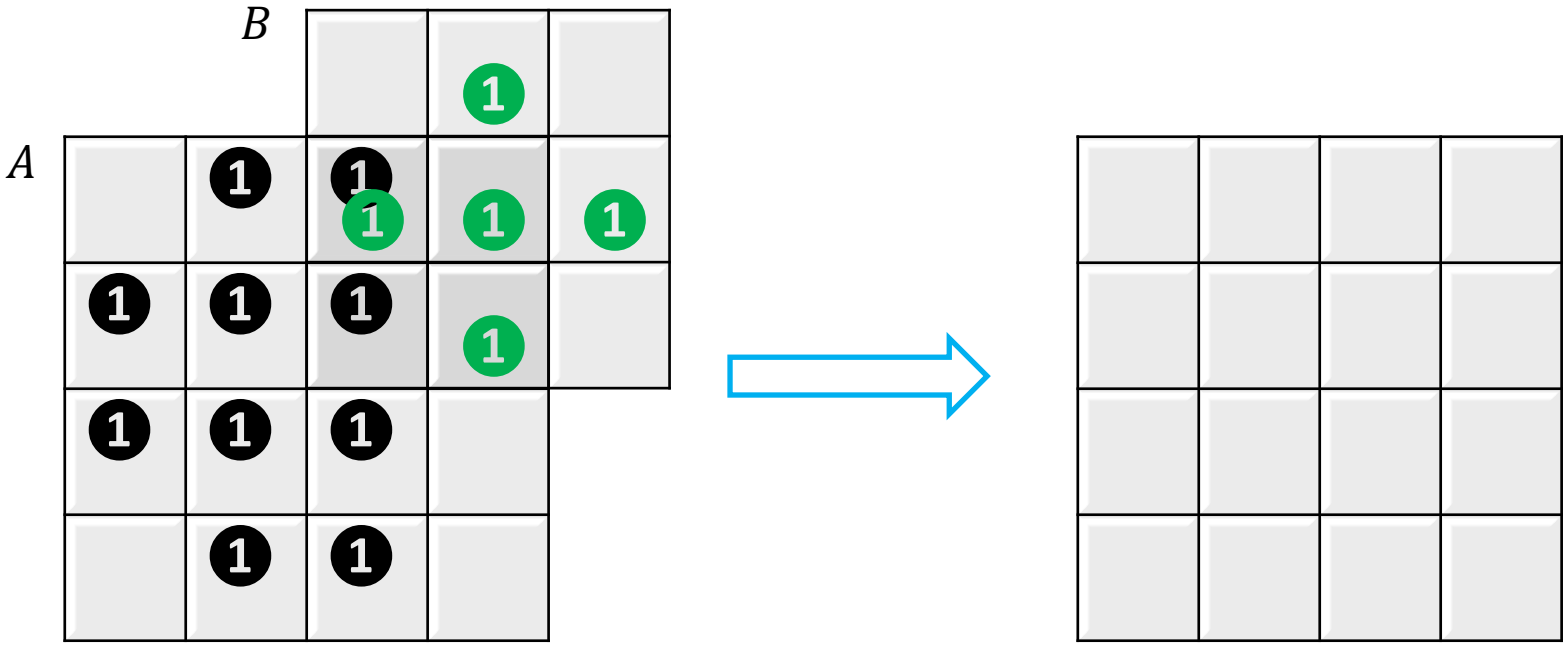
- Ergebnis:



# Erosion

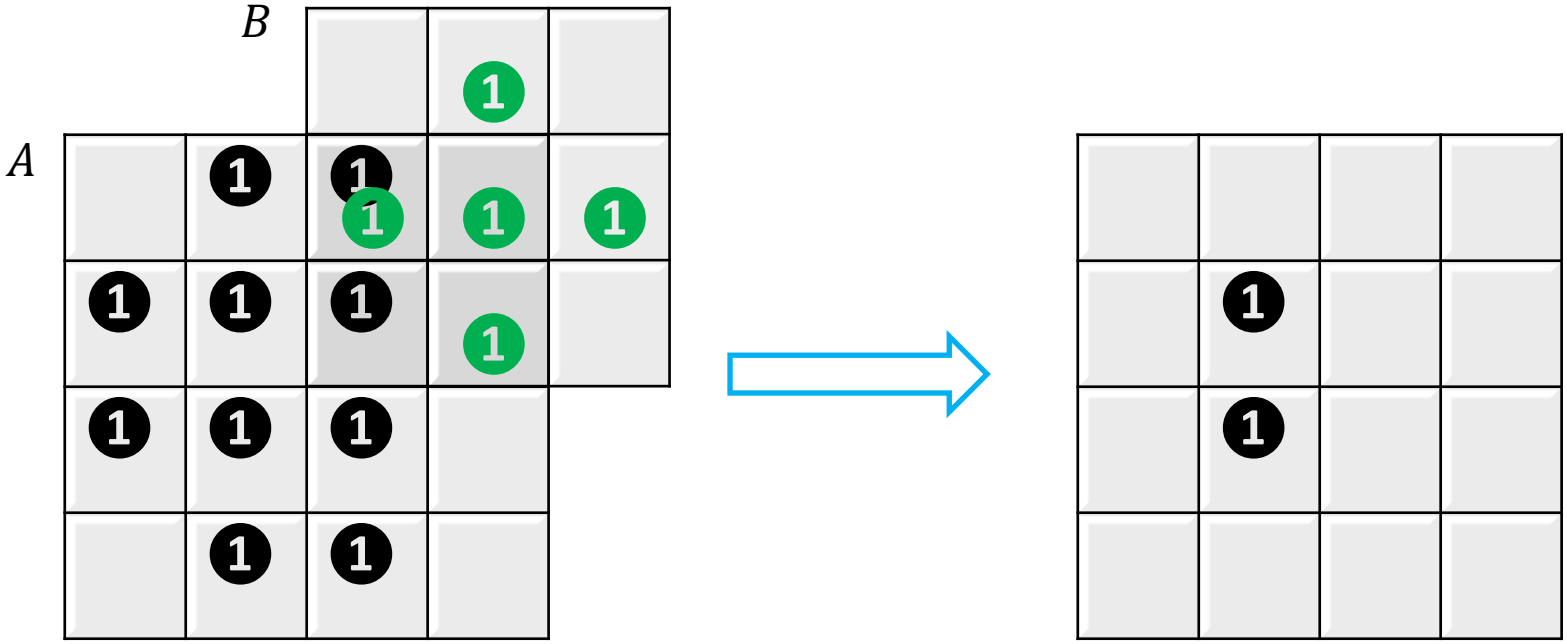
- Menge aller Pixel $z$, bei denen alle „1“ im Strukturelement mit einer „1“ in $A$ überlappen

- Frage: Welche Pixel sind nach der Erosion „1“?

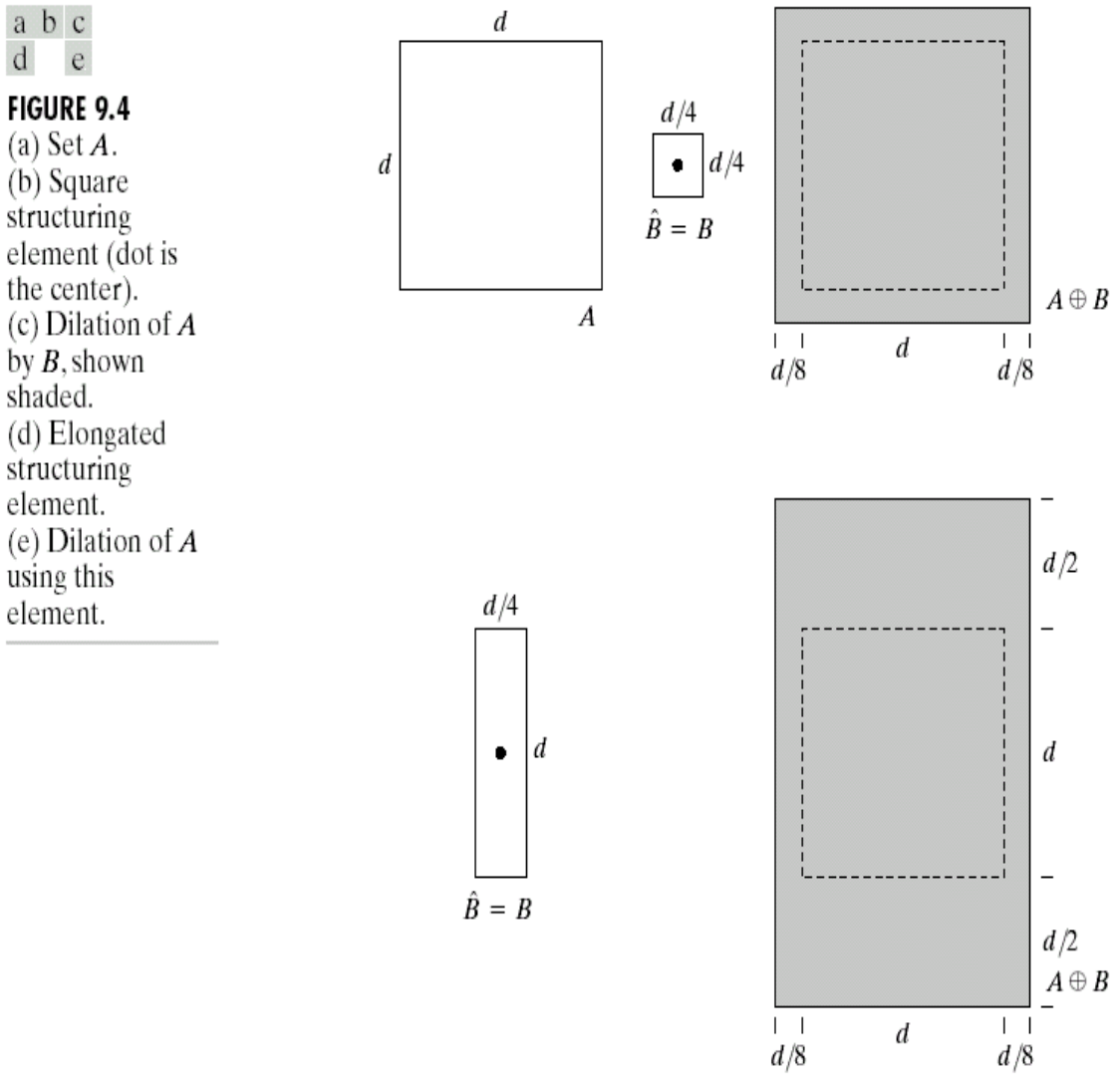


# Erosion

- Ergebnis:



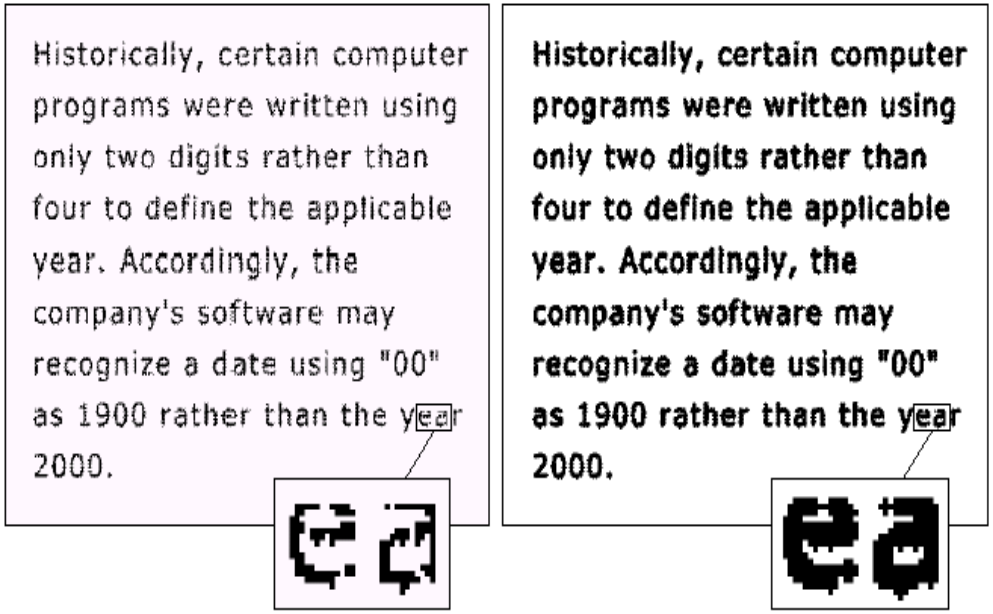
# Dilatation (Erweiterung)




a b c
d e

**FIGURE 9.4**
(a) Set $A$.
(b) Square structuring element (dot is the center).
(c) Dilation of $A$ by $B$, shown shaded.
(d) Elongated structuring element.
(e) Dilation of $A$ using this element.

# Dilatation (Erweiterung)

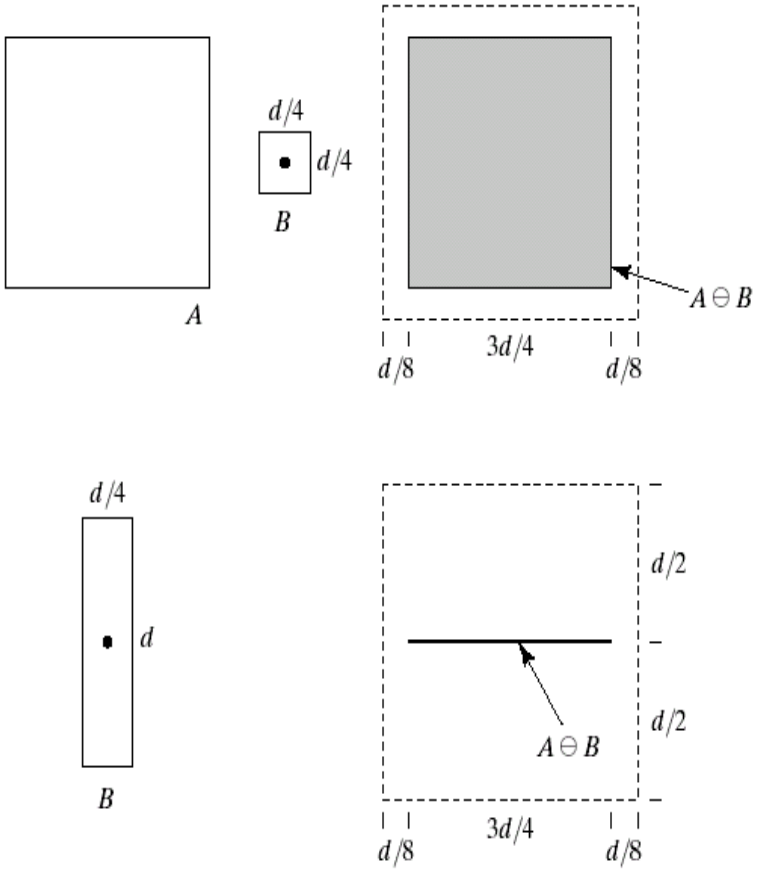



| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

a c
b

**FIGURE 9.5**
(a) Sample text of poor resolution with broken characters (magnified view). (b) Structuring element. (c) Dilation of (a) by (b). Broken segments were joined.

# Erosion




a b c
d e

**FIGURE 9.6** (a) Set A. (b) Square structuring element. (c) Erosion of $A$ by $B$, shown shaded. (d) Elongated structuring element. (e) Erosion of $A$ using this element.

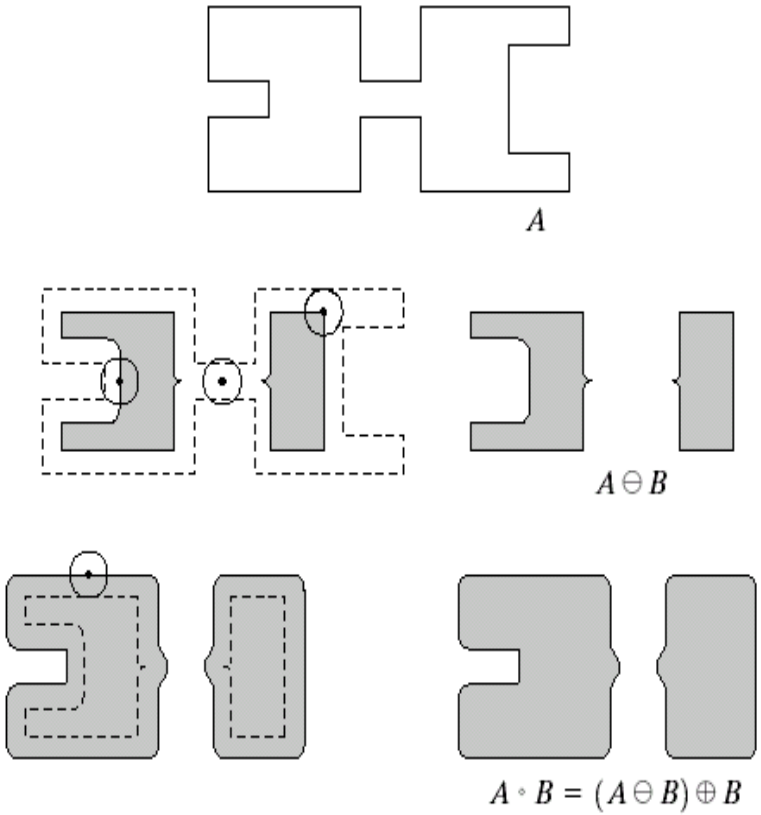
# Erosion (Abtragung)

# Öffnen und Schließen

- **Öffnen**: glättet die Kontur, entfernt Überstände und Brücken

$$A \odot B = (A \ominus B) \oplus B$$

- Erosion, gefolgt von Dilatation:

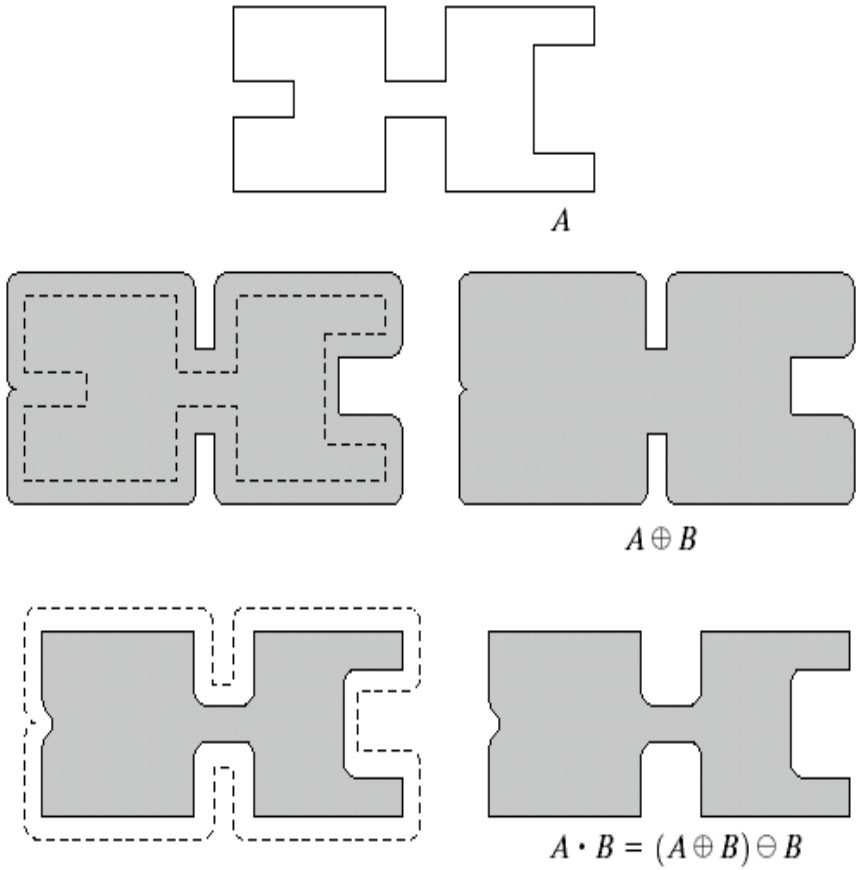


# Öffnen und Schließen

- **Schließen**: glättet Teile der Kontur, schließt kleinere Lücken

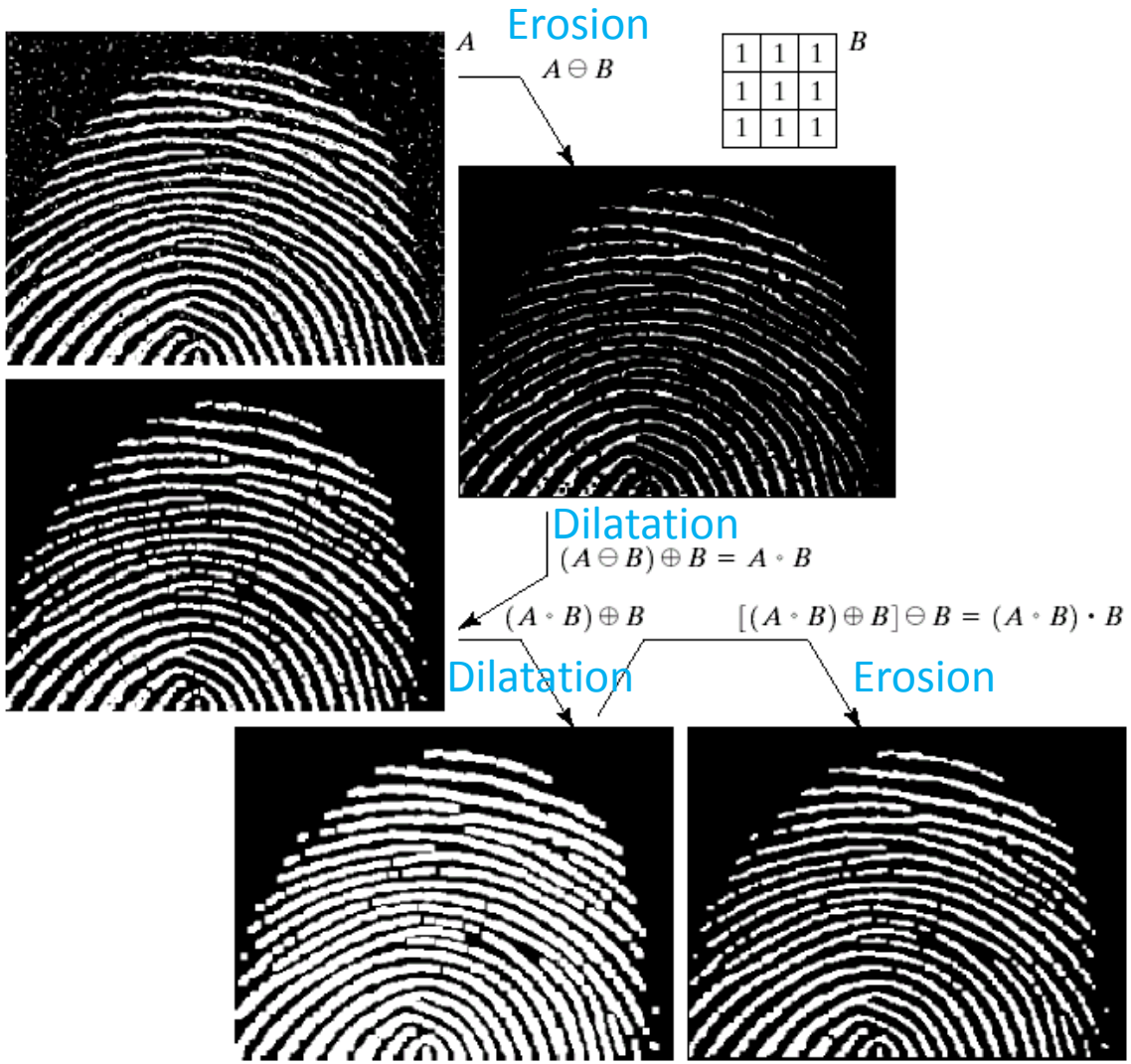
$$A \odot B = (A \oplus B) \ominus B$$

- Dilatation, gefolgt von Erosion



# Öffnen und Schließen

- erst Öffnen (Erosion und Dilatation), dann Schließen (Dilatation und Erosion)




a b
d c
e f

**FIGURE 9.11**
(a) Noisy image. (c) Eroded image. (d) Opening of $A$. (d) Dilation of the opening. (e) Closing of the opening. (Original image for this example courtesy of the National Institute of Standards and Technology.)

# Kontur-Extraktion

- Erosion, gefolgt von einer Subtraktion

$$\beta(A) = A - (A \ominus B)$$

a b
c d

**FIGURE 9.13** (a) Set $A$. (b) Structuring element $B$. (c) $A$ eroded by $B$. (d) Boundary, given by the set difference between $A$ and its erosion.