Faltung, Kanten und Template Matching

Christian Wilms

Computer Vision Group
Universität Hamburg

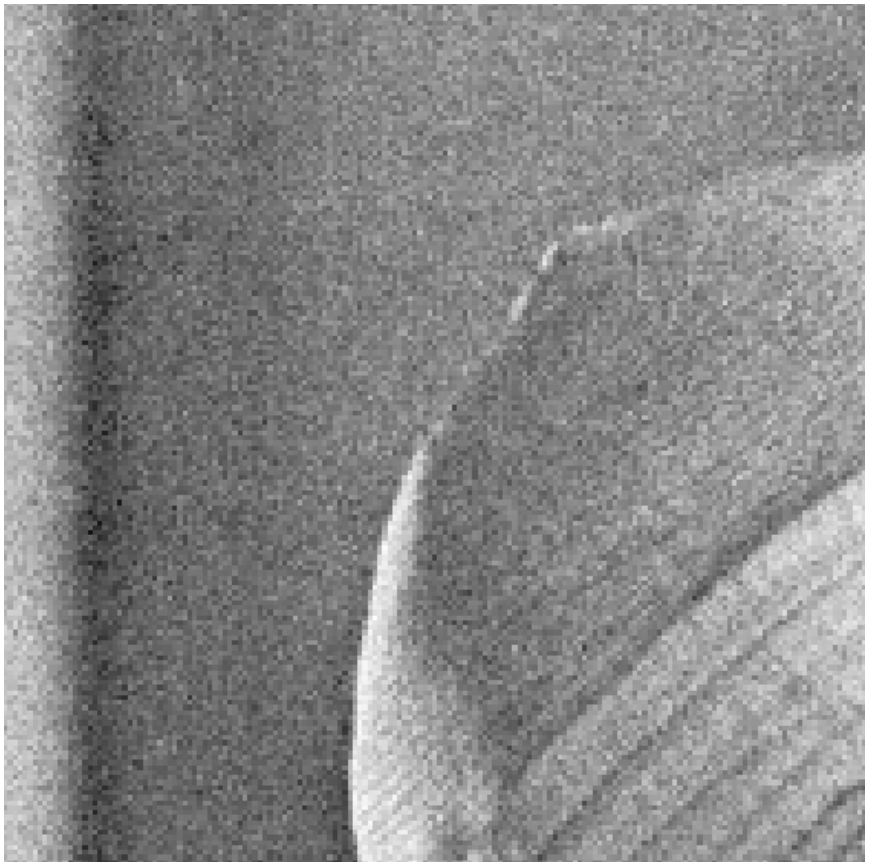
Sommersemester 2018

03. Mai 2018

Übersicht

Habt ihr euch das Bild schon mal näher angesehen?

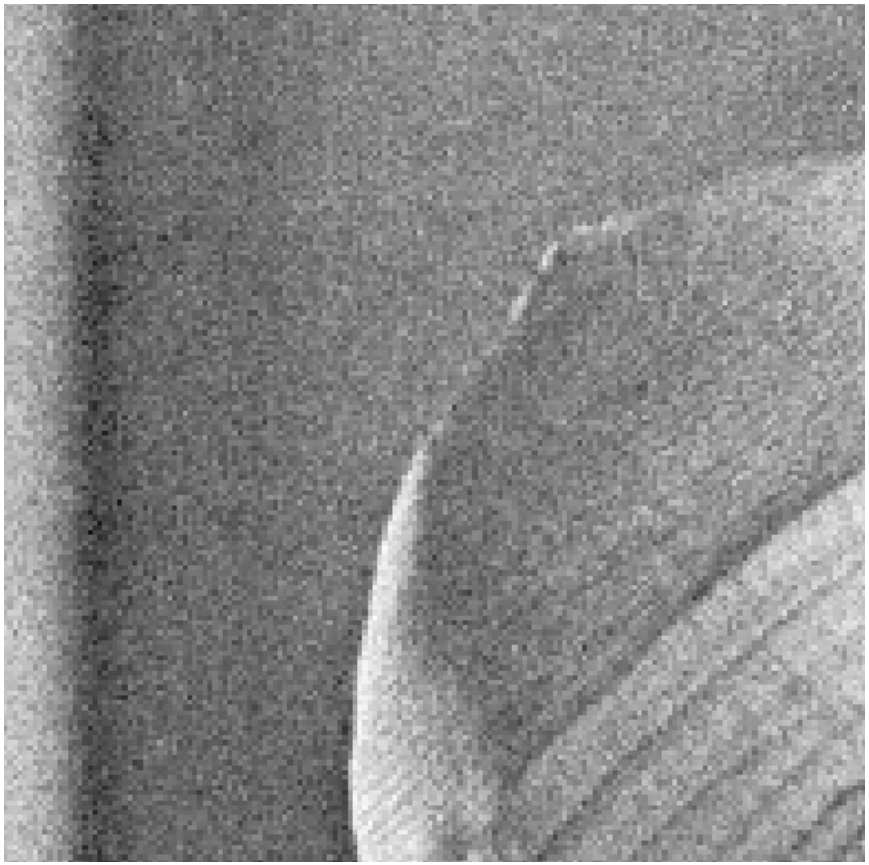
Detailansicht



Irgendwie sieht das 'krisselig' aus... [1]

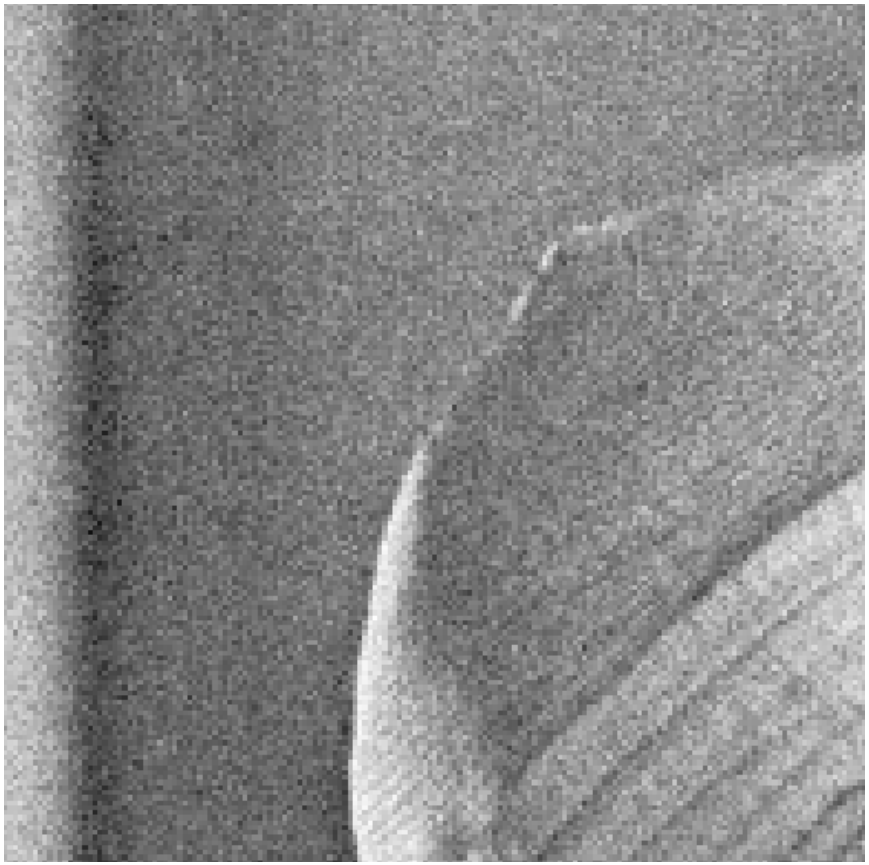
[1] `plt.imshow(img, cmap='Greys_r', interpolation='nearest')`

Detailansicht



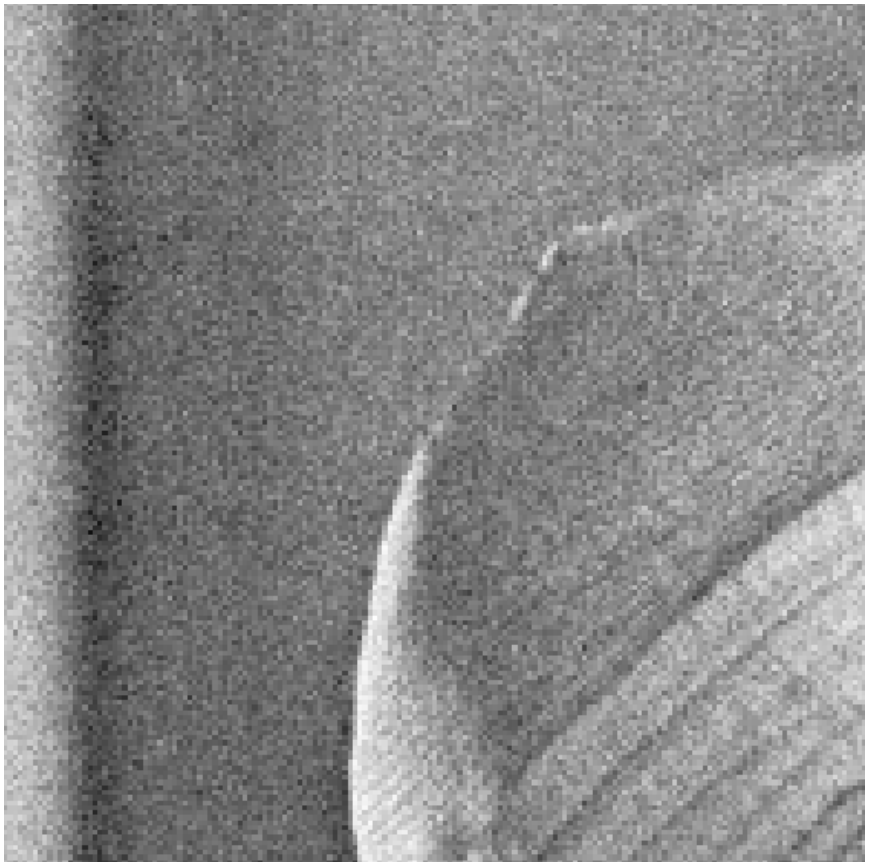
Wie nennt man diese krisseligen Strukturen?

Detailansicht



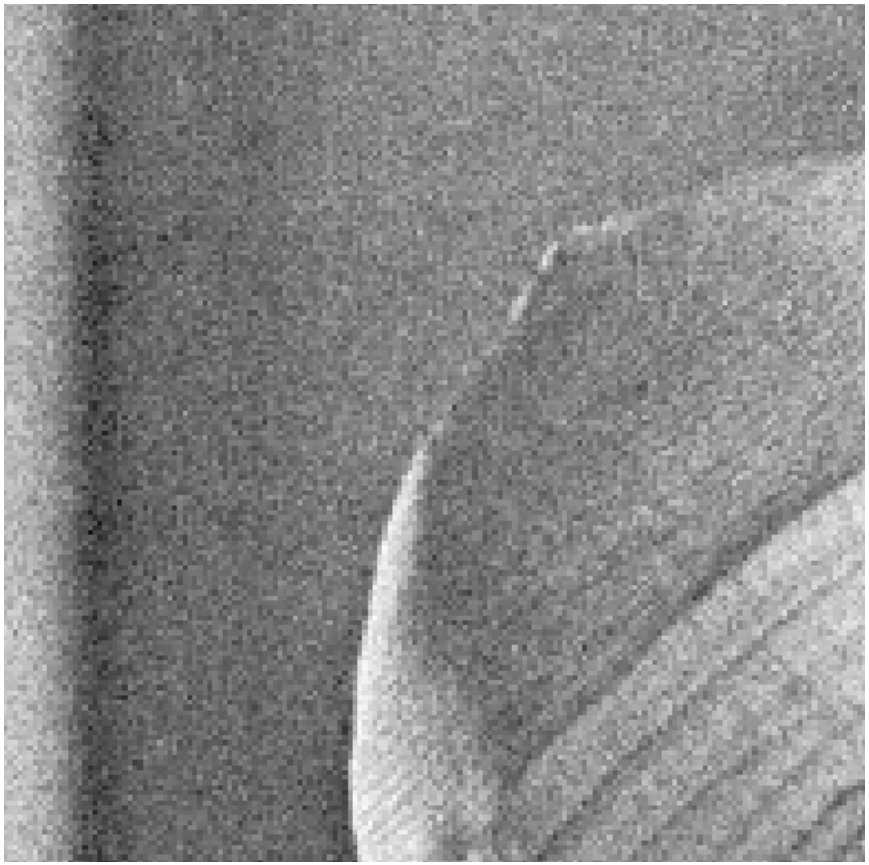
Woher kommen die krisseligen Strukturen?

Detailansicht



Was kann man dagegen tun?

Detailansicht



Mitteln über die Nachbarschaft!

Optischer Vergleich

verrauschte Lenna

entrauschte Lenna (3×3)

Optischer Vergleich

verrauschte Lenna

entrauschte Lenna (5×5)

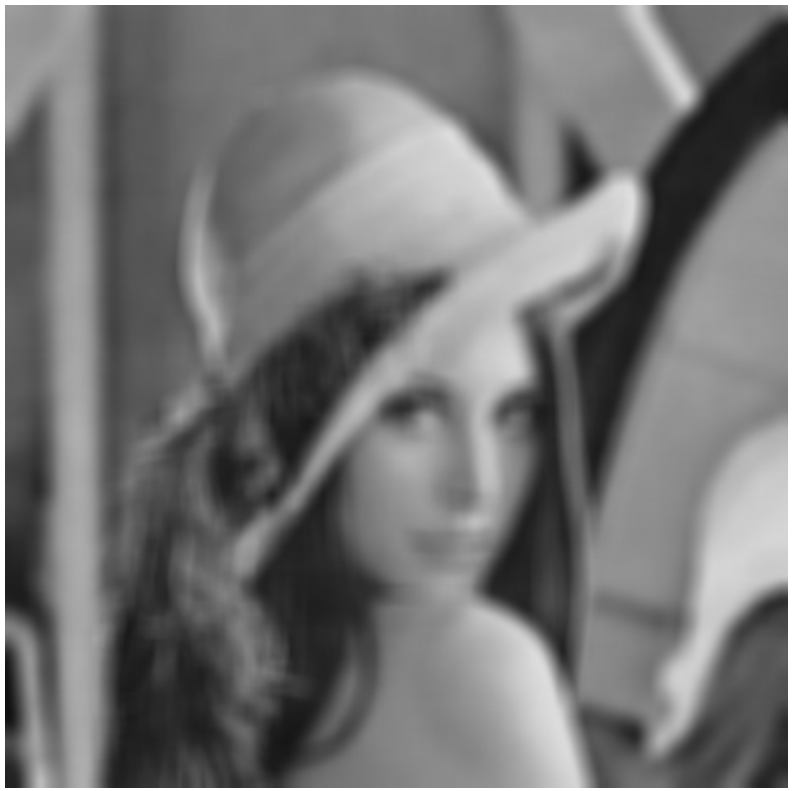
Optischer Vergleich

verrauschte Lenna

entrauschte Lenna (7×7)

Optischer Vergleich

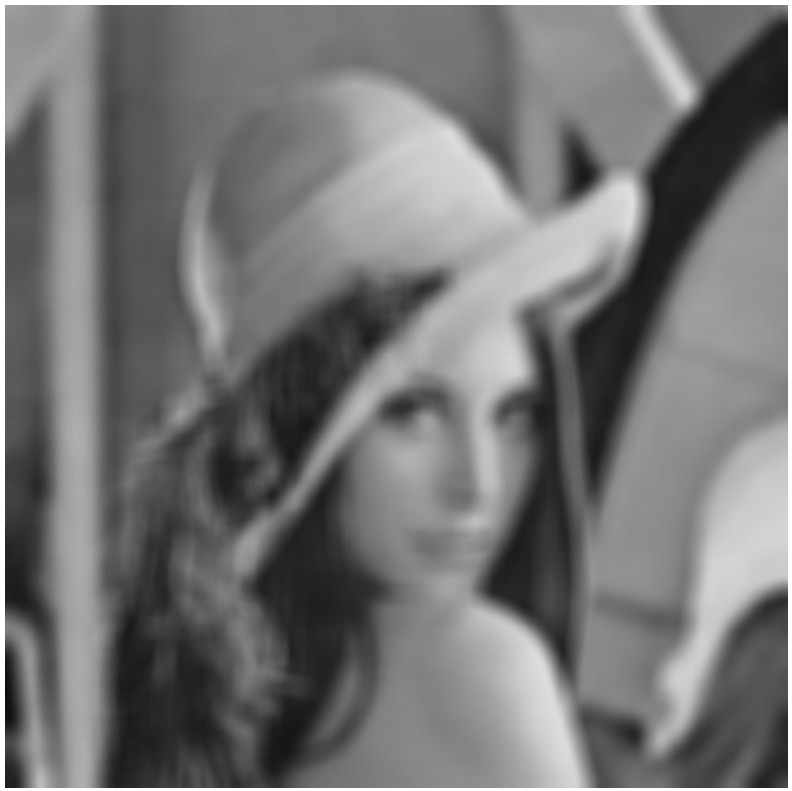
verrauschte Lenna



entrauschte Lenna (11×11)

Optischer Vergleich

verrauschte Lenna

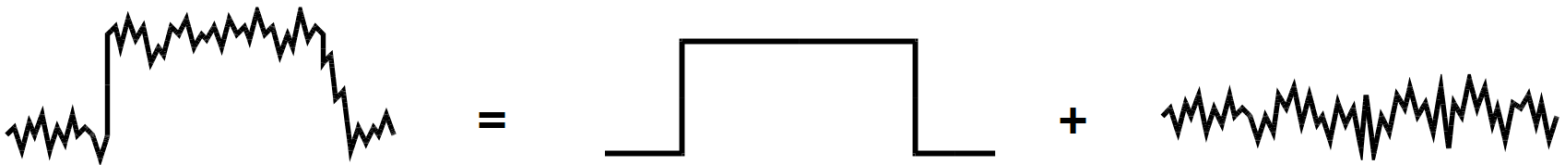


entrauschte Lenna (15×15)

Rauschen

Rauschen...

- ... sind Pixel, die nicht das eigentlich Bildsignal zeigen
- ... entsteht bei der Bildaufnahme bspw. durch schlechte Sensoren, Wärme/Kälte oder atmosphärische Effekte



Kann man den Prozess des Mittels über die Nachbarschaft auch formalisieren?

Kann man den Prozess des Mittels über die Nachbarschaft auch formalisieren?

Na klar! Faltung $*$

Faltung (Convolution) - I

Faltung

Faltung ist ein mathematischer Operator ($*$), der aus zwei Funktionen eine dritte Funktion erzeugt.

Faltung (Convolution) - I

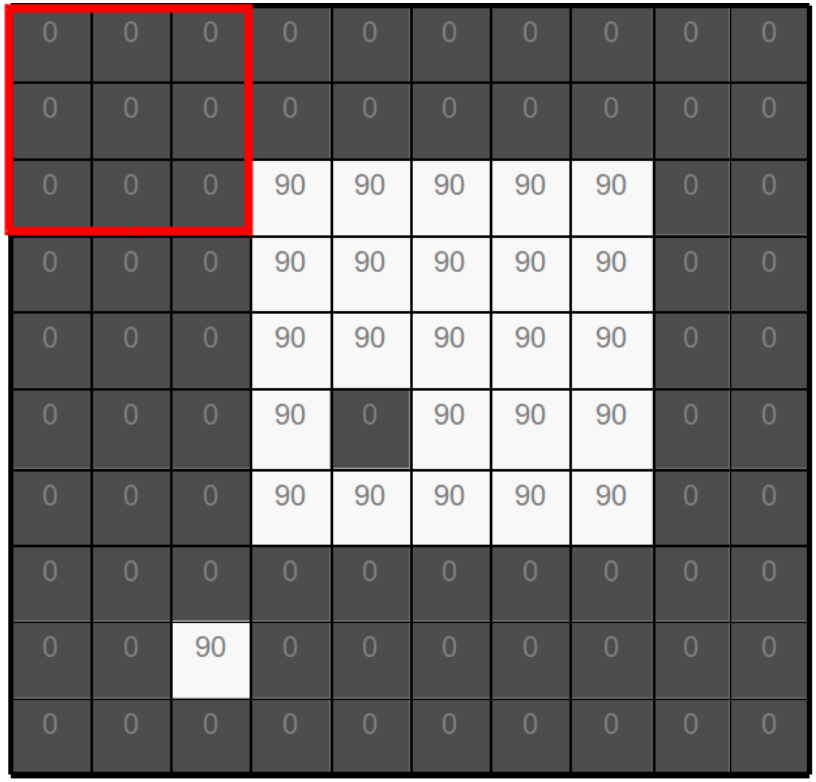
Faltung

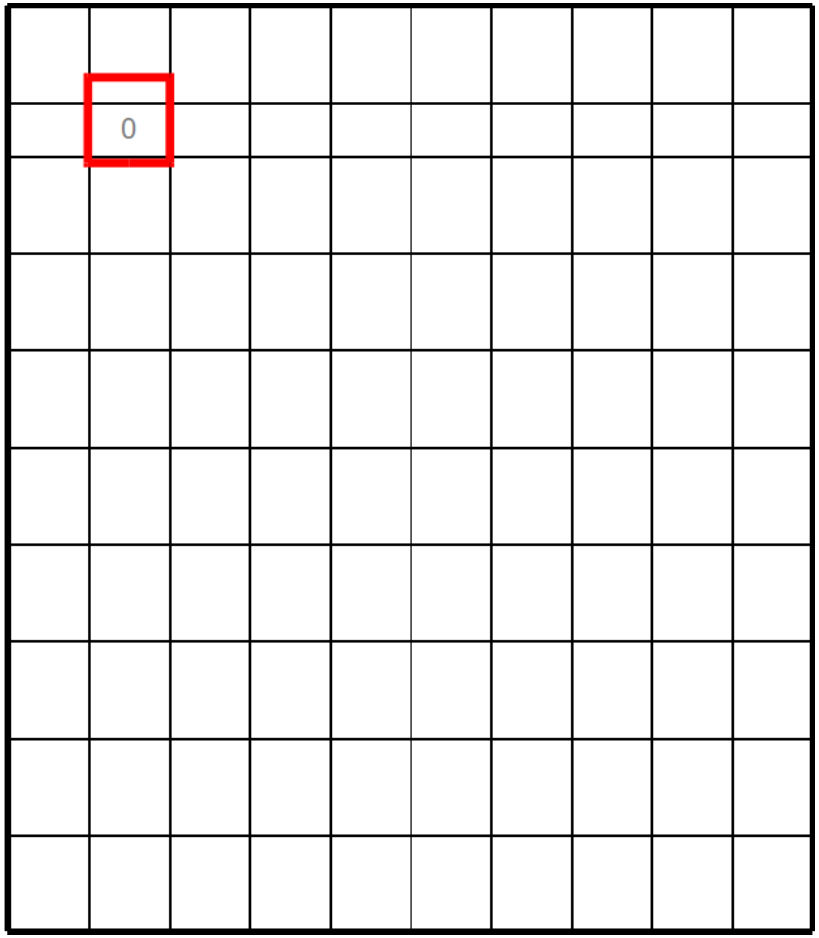
Praktisch: Eine Matrix (Faltungskern) wird auf jedes Pixel im Bild zentriert und die sich überdeckenden Matrixzellen multipliziert und aufsummiert. Das Ergebnis wir in ein neues Bild geschrieben.

Beispiel eines Faltungskerns: $\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} = \begin{bmatrix}\frac{1}{9} & \frac{1}{9} & \frac{1}{9}\\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9}\\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9}\end{bmatrix}$

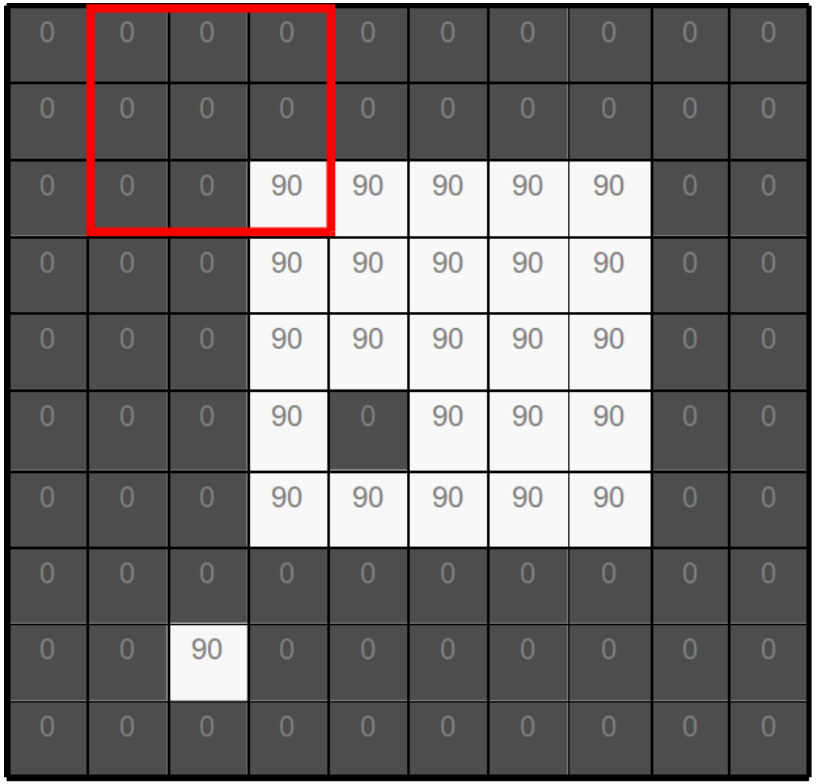
Beachte: Faltungskerne sind üblicherweise quadratische Matrizen mit ungerader Anzahl von Reihen/Spalten und werden vor der Anwendung an beiden Achsen gespiegelt!

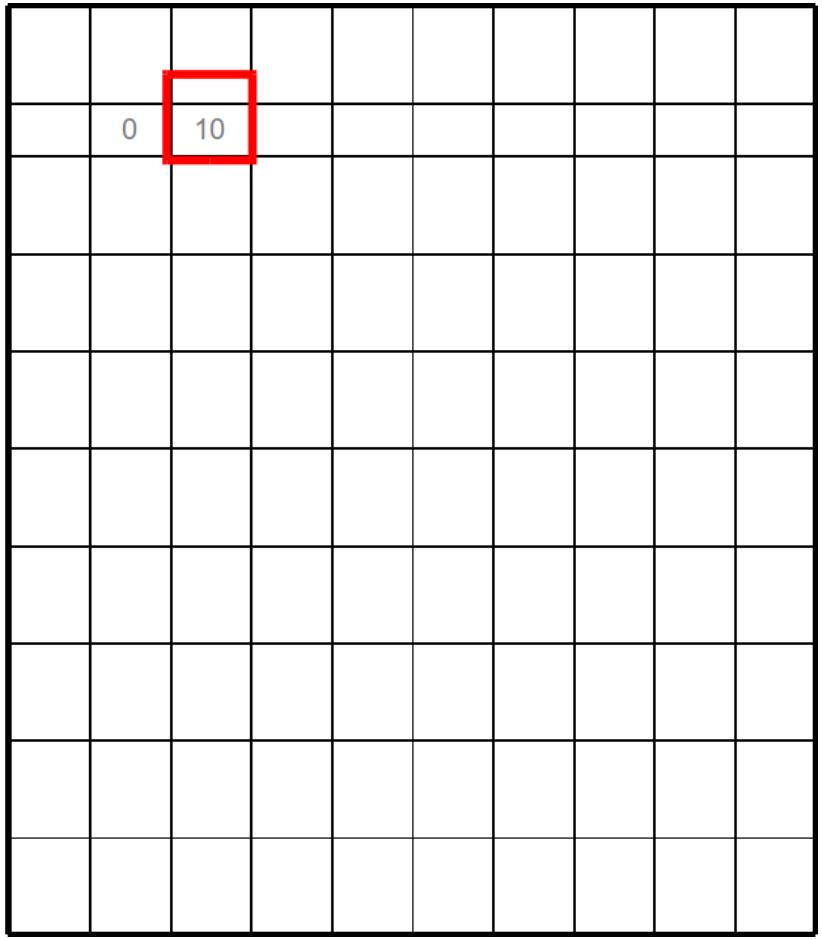
Faltung (Convolution) - II



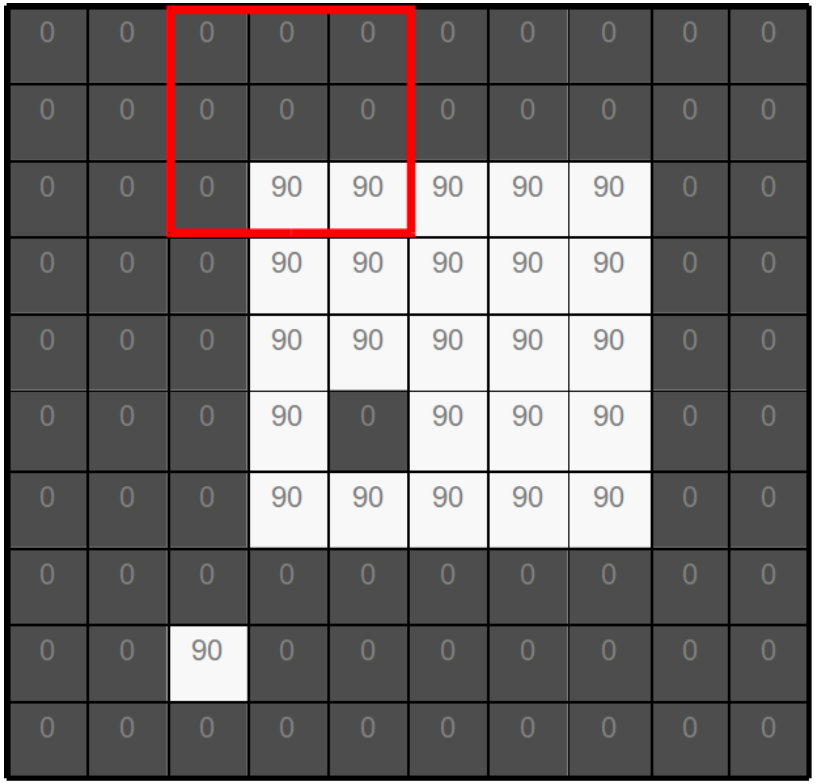


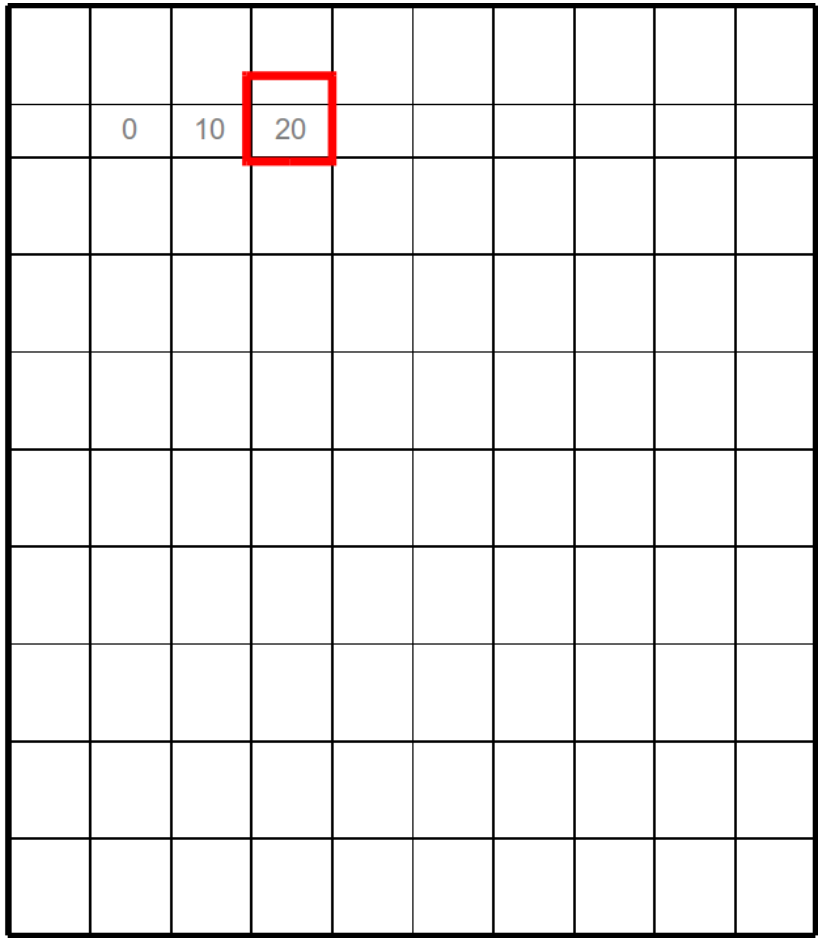
Faltung (Convolution) - II



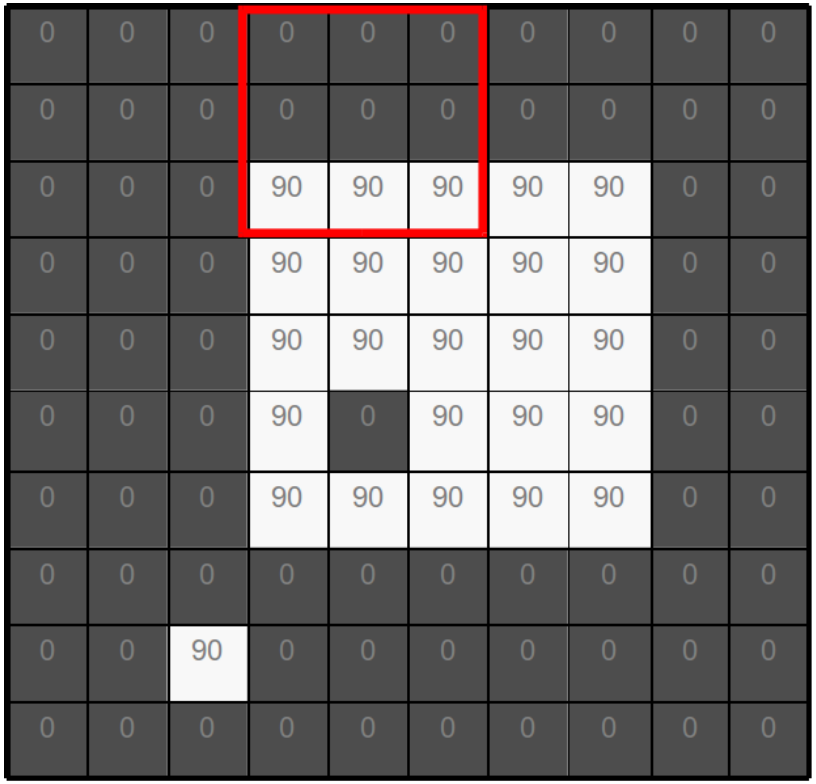


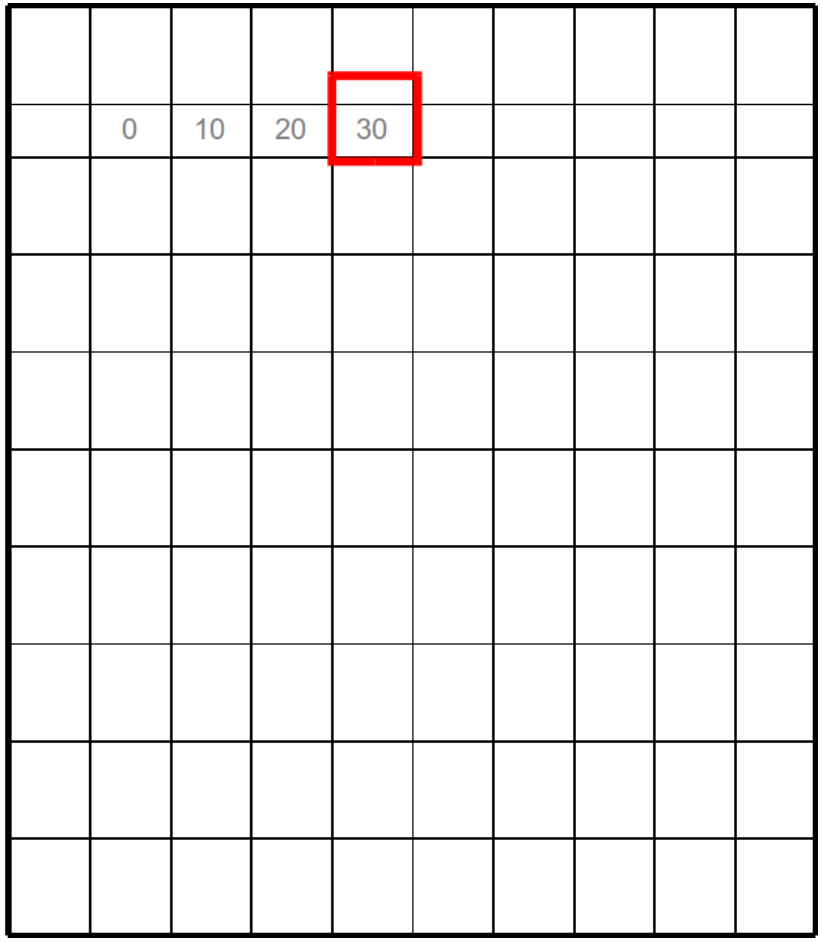
Faltung (Convolution) - II



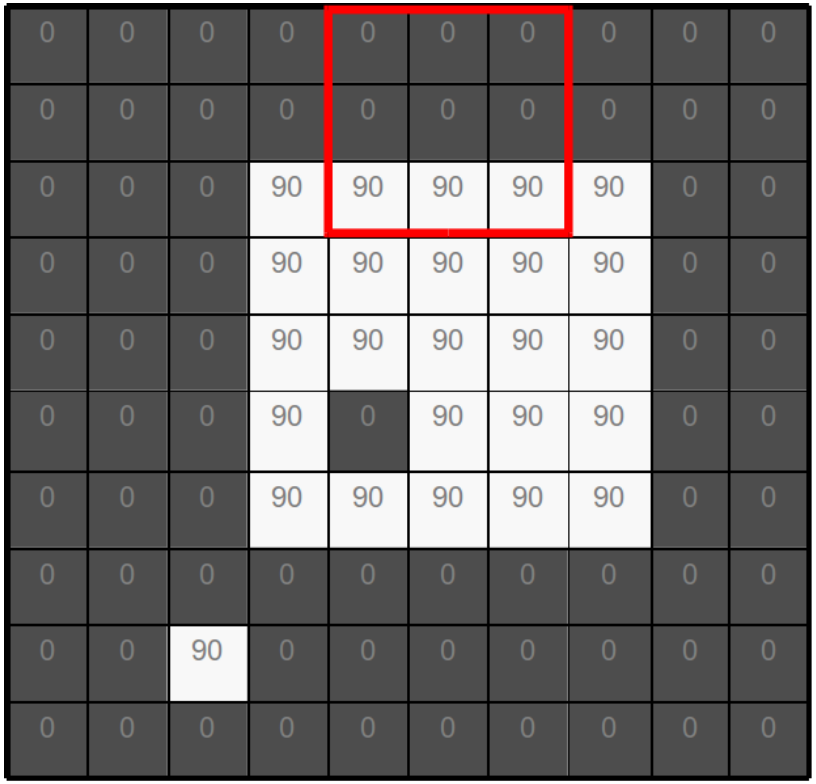


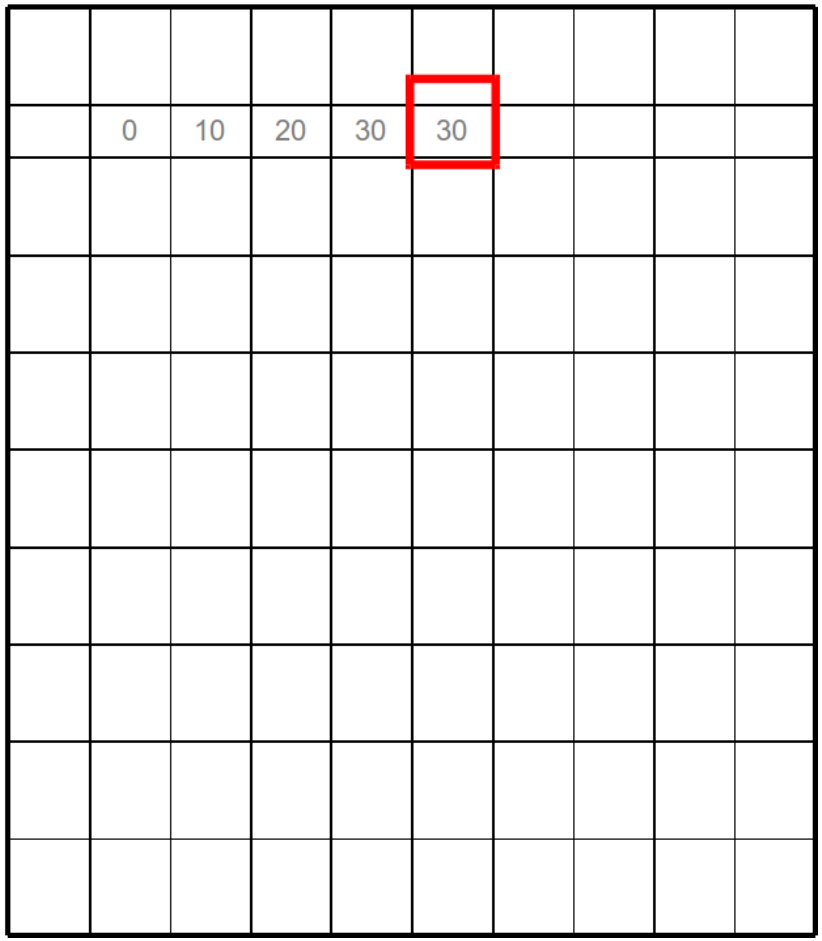
Faltung (Convolution) - II



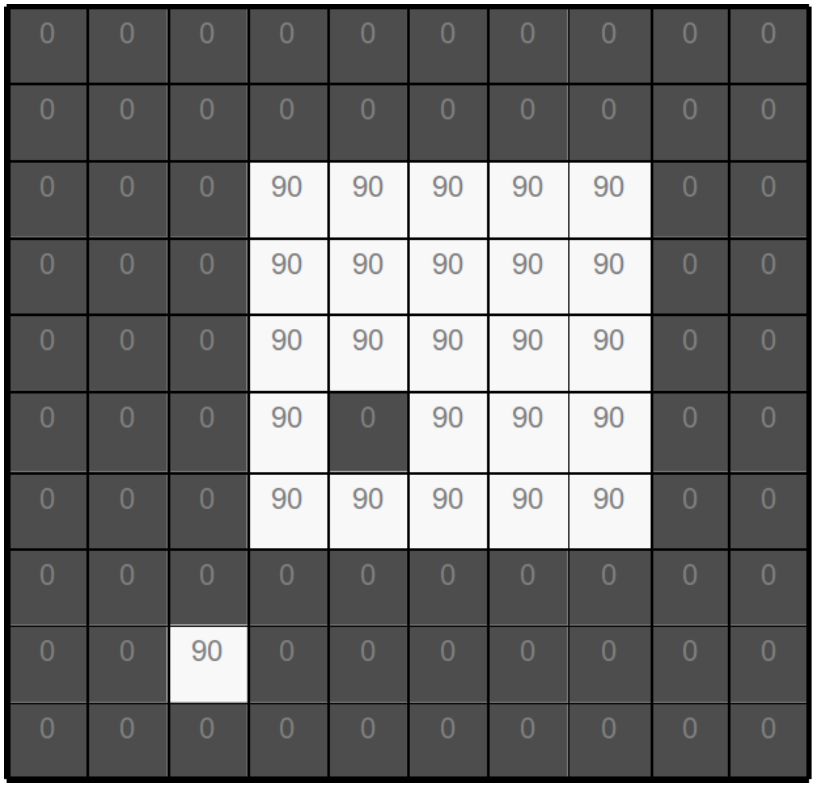


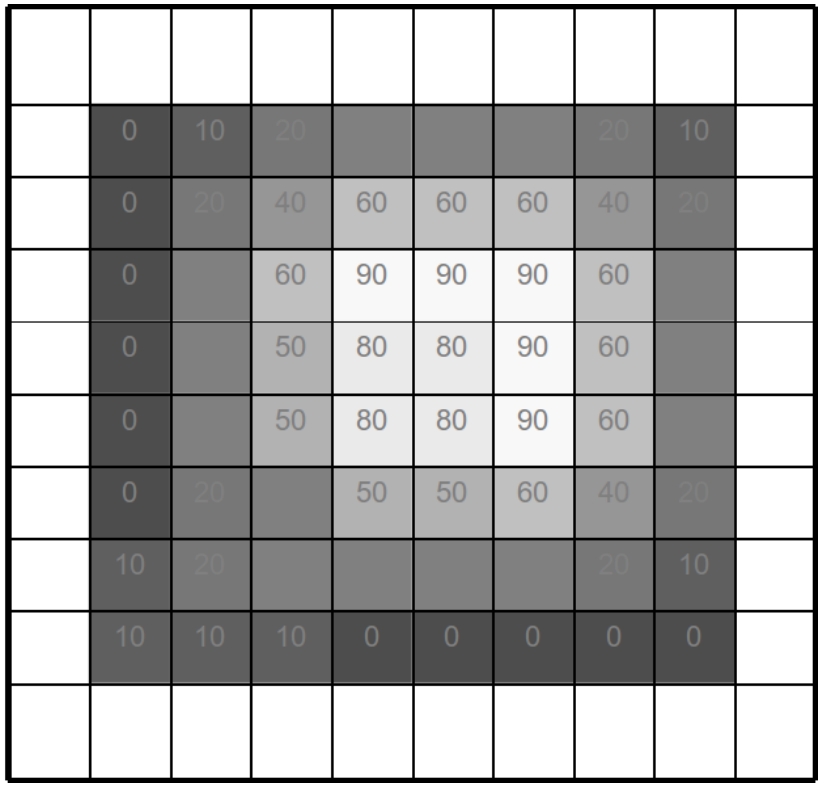
Faltung (Convolution) - II





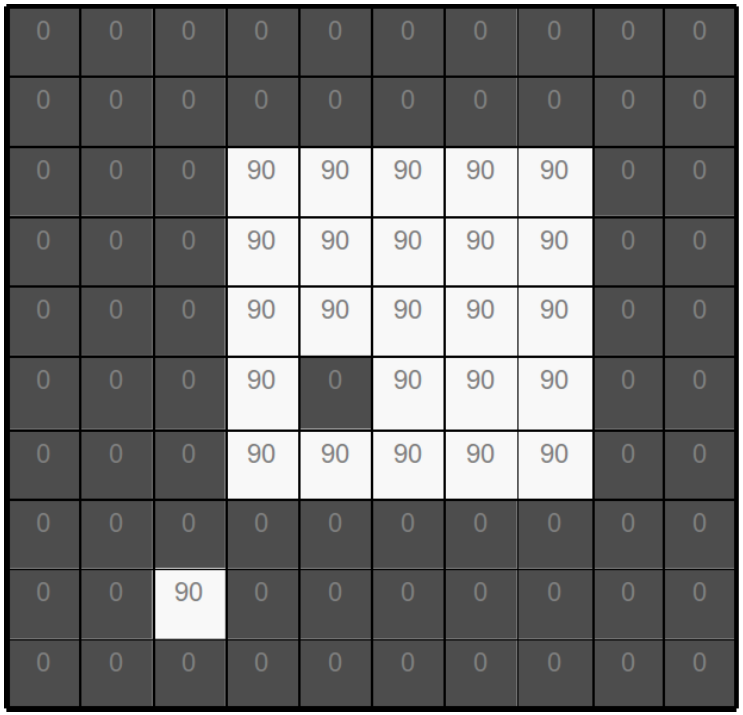
Faltung (Convolution) - II

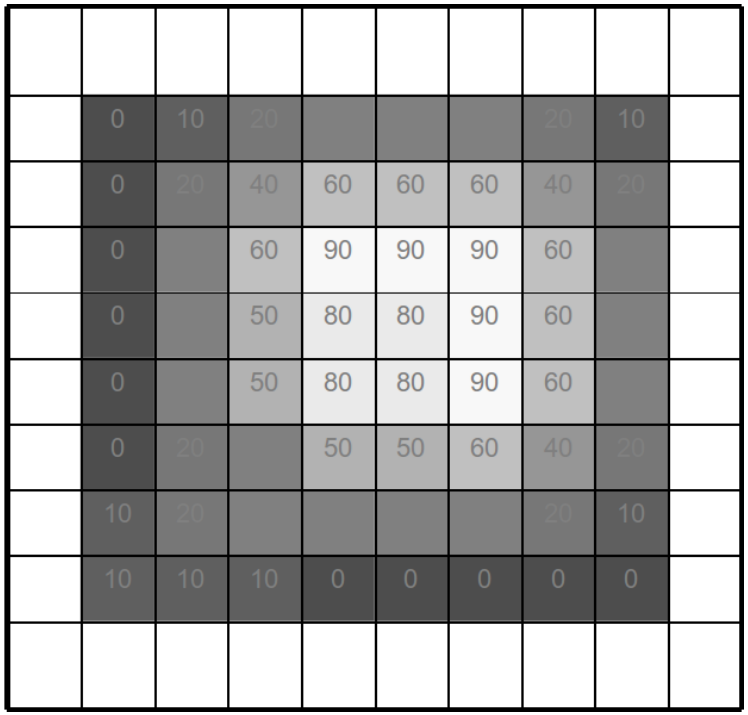




Niemals die Zwischenergebnisse ins Eingangsbild schreiben!

Problemfall Bildrand





- ignorieren → ggf. Maske oder Resultat verkleinern
- Umgebung mit einer Konstanten füllen
- Bild spiegeln oder wiederholen

Übersicht

Was kann man mit der Faltung machen?

- entrauschen/weichzeichnen
- schärfen
- Kanten finden
- ...

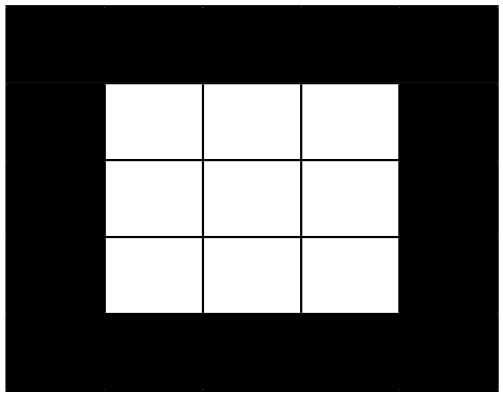
Was bringen uns die Faltung für die Klassifikation von Bildern?

- bspw. um Objekte oder Teilobjekte zu finden, brauchen wir Kanten (Bsp. Auto)
- Kanten lassen sich mit Faltungen aufspüren
- Aber was dann? $\rightarrow$ nächsten Wochen

Was sind Kanten?

- Kanten sind die Bereiche im Bild, an denen sich benachbarte Pixel stark unterscheiden
- wir bleiben für Kanten auf Grauwertbildern

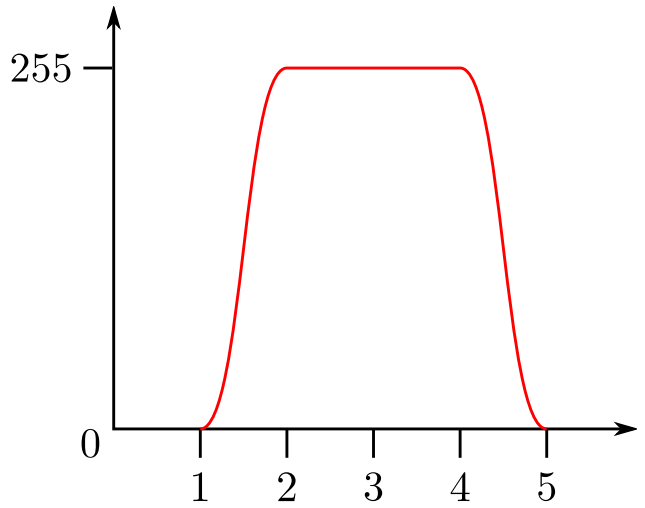
0	0	0	0	0
0	255	255	255	0
0	255	255	255	0
0	255	255	255	0
0	0	0	0	0



Wie kann man Kanten erkennen?

- Kanten sind die Bereiche im Bild, an denen sich benachbarte Pixel stark unterscheiden
- die erste Ableitung kann solche Steigungen aufdecken → Extrempunkte

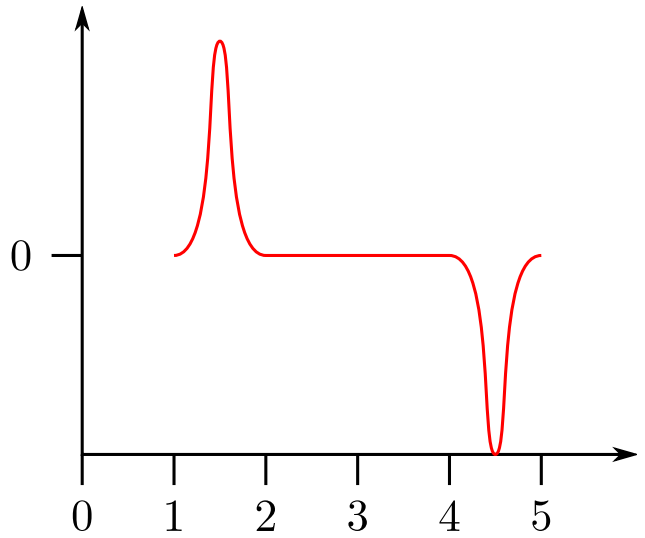
0	0	0	0	0
0	255	255	255	0
0	255	255	255	0
0	255	255	255	0
0	0	0	0	0



Wie kann man Kanten erkennen?

- Kanten sind die Bereiche im Bild, an denen sich benachbarte Pixel stark unterscheiden
- die erste Ableitung kann solche Steigungen aufdecken → Extrempunkte

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



Ableitungen in Bildern?

- die erste Ableitung berechnet sich aus der Differenz von benachbarten Punkten
- Punkte sind in einem Bild die diskreten Pixel
- Ableitung entspricht Differenz zu Nachbarn

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

-1	0	1

Zentraldiff.

Ableitungen in Bildern?

- die erste Ableitung berechnet sich aus der Differenz von benachbarten Punkten
- Punkte sind in einem Bild die diskreten Pixel
- Ableitung entspricht Differenz zu Nachbarn

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

-1	0	1
-2	0	2
-1	0	1

Sobel-Filter besser gegen Rauschen.

Die Sobel-Filter sind bei scikit-image bereits vorhanden.

Von 1D auf 2D

- Erweiterung auf 2D durch unabhängige Betrachtung der Zeilen und Spalten $\rightarrow$ partielle Ableitungen nach x und y

$$\frac{\partial f}{\partial x} = f(x+1, y) - f(x, y)$$

$$\frac{\partial f}{\partial y} = f(x, y+1) - f(x, y)$$

- Differenzen werden entsprechend gedreht berechnet

-1	0	1
-2	0	2
-1	0	1

horizontaler Sobel-Filter

-1	-2	-1
0	0	0
1	2	1

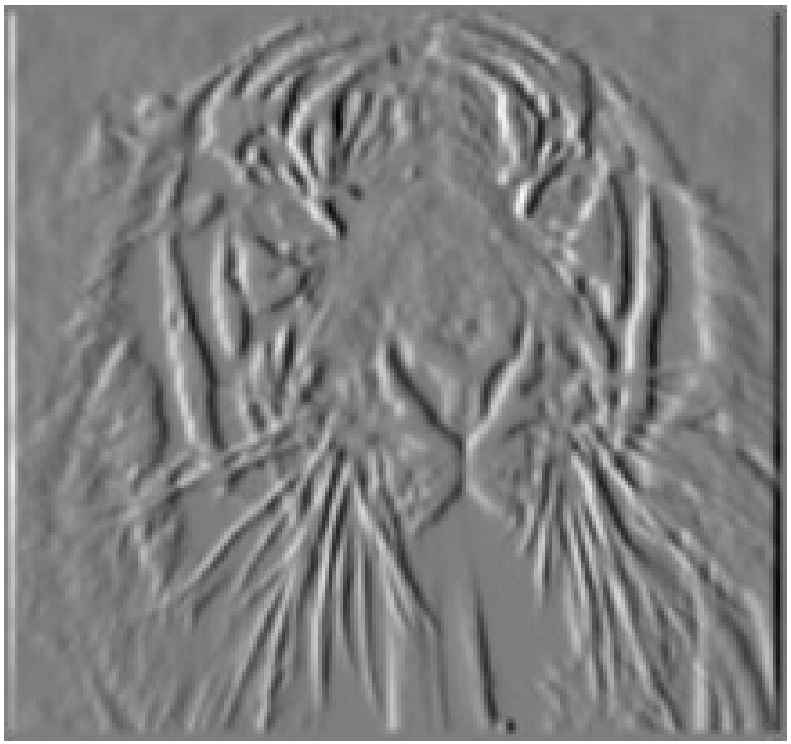
vertikaler Sobel-Filter

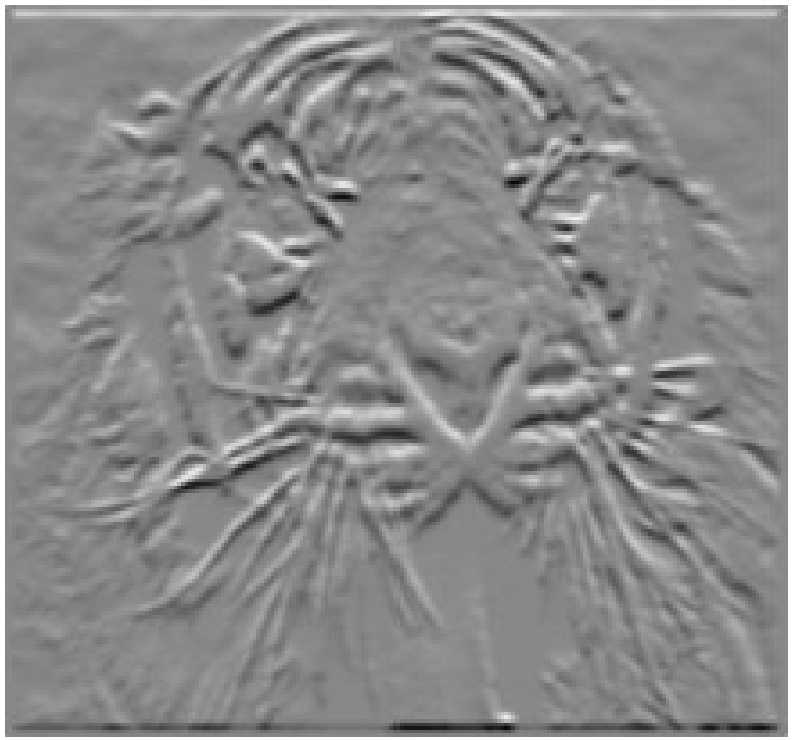
Von 1D auf 2D

- Erweiterung auf 2D durch unabhängige Betrachtung der Zeilen und Spalten $\rightarrow$ partielle Ableitungen nach x und y
- Differenzen werden entsprechend gedreht berechnet

Von 1D auf 2D

- Erweiterung auf 2D durch unabhängige Betrachtung der Zeilen und Spalten $\rightarrow$ partielle Ableitungen nach x und y
- Differenzen werden entsprechend gedreht berechnet





Kombination der Ergebnisse

Gradient

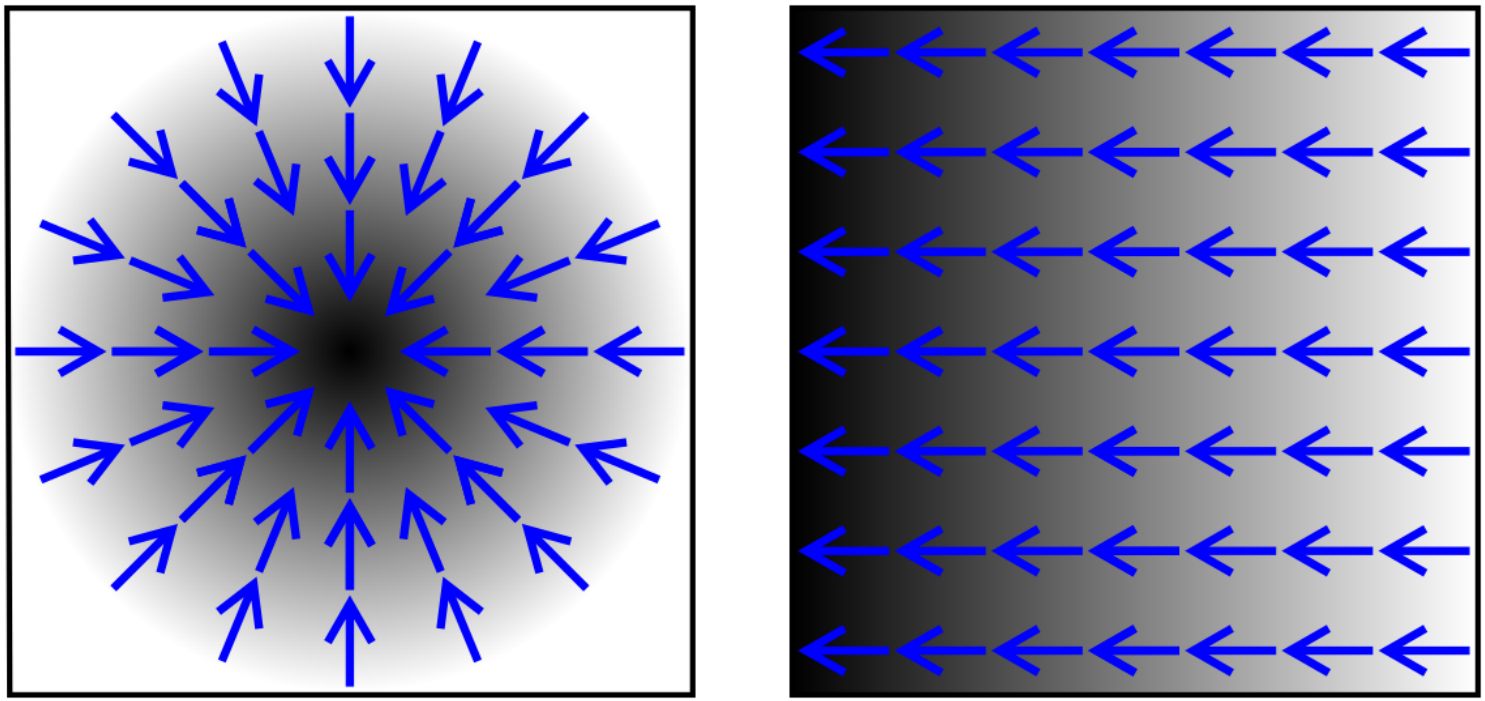
Die Kombination der beiden partiellen Ableitungen bzw. Ergebnisse der Sobel-Filter bilden zusammen einen Vektor:

$$\nabla f = grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]^T$$

- der Gradient lässt sich für jedes Pixel aus den Einzelergebnissen der Sobel-Filter berechnen
- die **Stärke** der Kante entspricht der Länge des Gradienten
- die **Orientierung** des Gradienten und damit der Kante lässt sich bestimmen als:

$$\Theta = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

Was bedeutet das?



Visualisierung der Gradienten

Was kann man damit machen in Bezug auf Klassifikation? $\rightarrow$
Histogramme über Gradientenorientierung im Bild als Deskriptor

Histogram of Oriented Gradients (HOG)

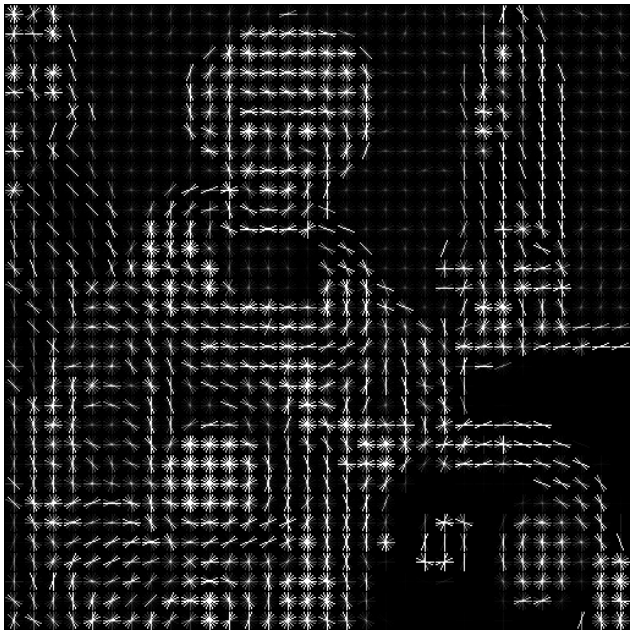
Basisprinzip

- für jeden Pixel Gradienten berechnen
- Gradientenorientierung kalkulieren
- Histogramm über die Orientierungen
- entweder ein Histogramm über das ganze Bild oder in einzelnen Kacheln

Im Verfahren gibt es zusätzlich noch Normalisierungsschritte, um unabhängiger gegen Beleuchtung zu sein.

Histogram of Oriented Gradients (HOG)

Basisprinzip



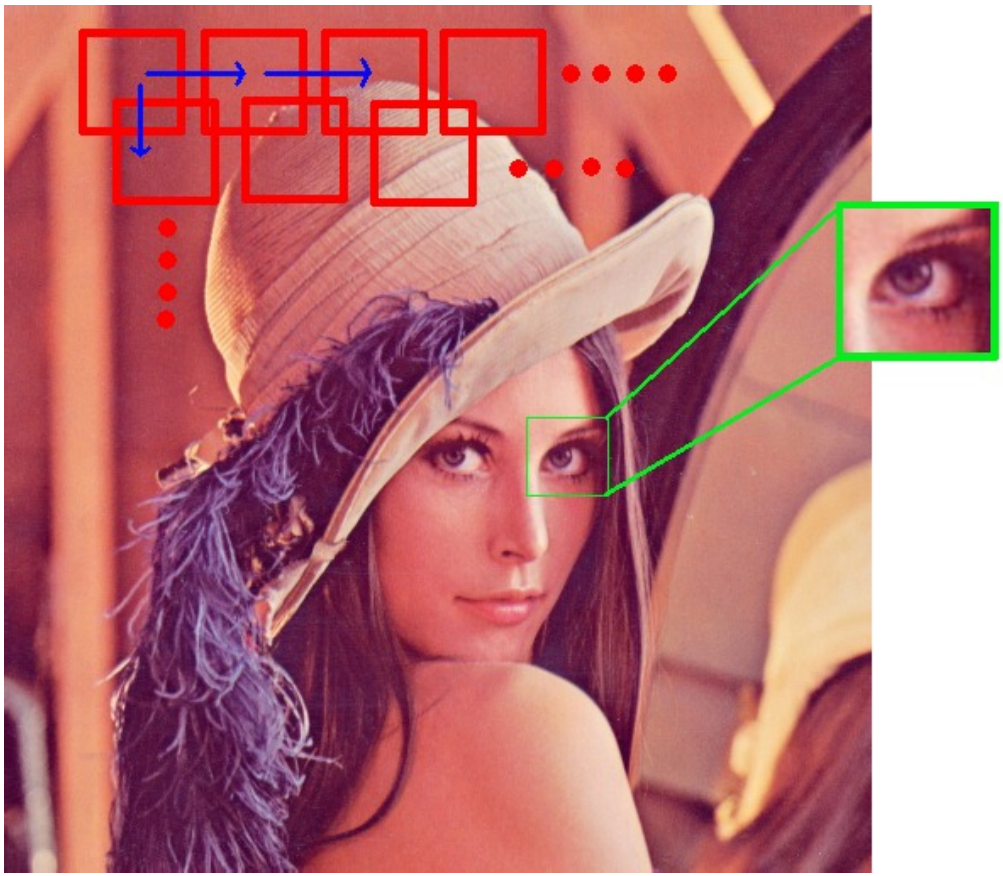
Übersicht

Was kann man mit der Faltung noch so machen?

Korrelation:

- eine mathematische Operation ähnlich zur Faltung
- einziger Unterschied, der Faltungskern wird vor der Anwendung nicht gespiegelt
- bietet die Möglichkeit mit Hilfe einer Maske Formen im Bild zu finden (Wo passt die Maske am besten?)

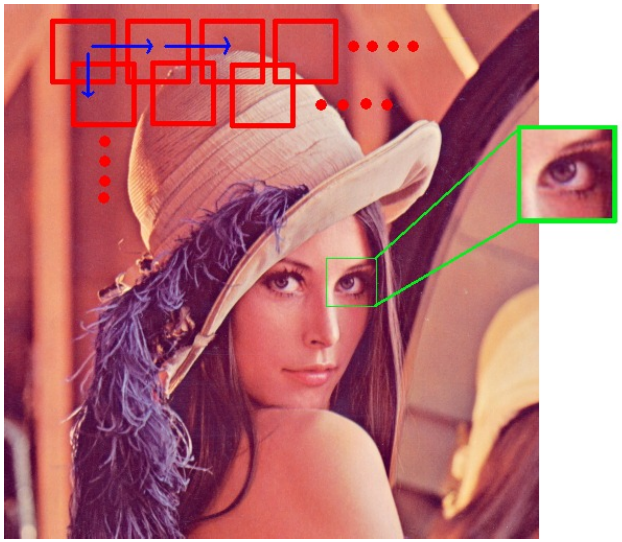
Template Matching - Beispiel



Wo ist das Auge? [1]

[1] https://rvlab.icg.tugraz.at/project_page/project_efficient_ncc/images/lena.jpg

Template Matching

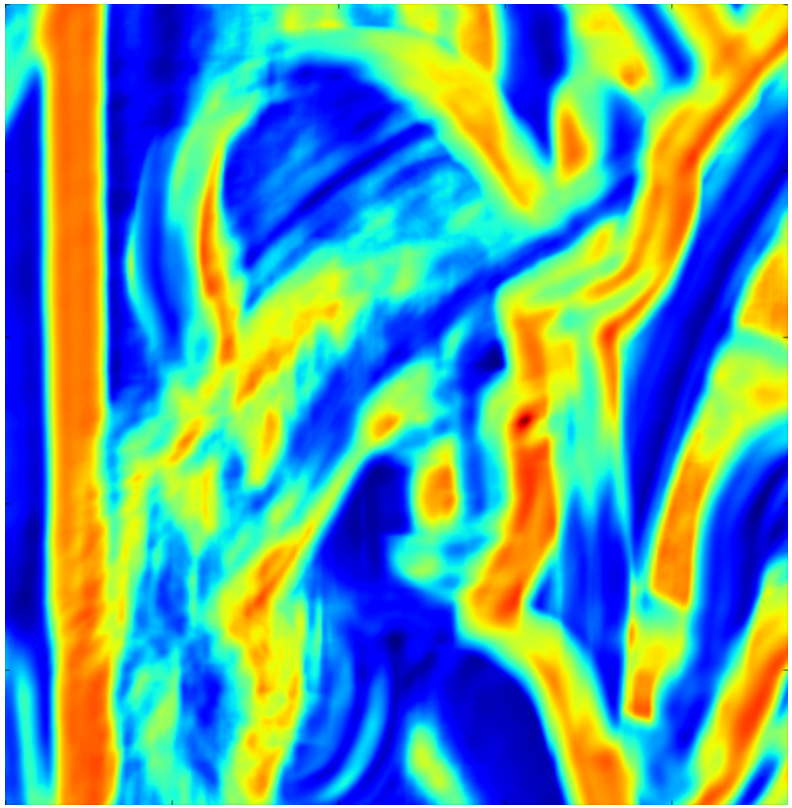


Wo ist das Auge? [1]

Template Matching

- es wird eine Maske auf jedes Pixel im Bild zentriert
- die K. gibt an, wie gut die Maske zu einem Bildteil passt
- um die Ergebnisse zu verbessern, wird die Korrelation noch normiert (normalized cross-correlation)

Template Matching mit scikit-image - Ergebnis anzeigen



```
>>> result = match_template(img2, eye)
>>> plt.imshow(result)
```

Template Matching mit scikit-image - Koordinaten ermitteln

```
>>> result = match_template(img2, eye)
>>> x,y = np.unravel_index(np.argmax(result), result.
    shape)
```

- `argmax` rollt das Array ab $\rightarrow$ 2D-Indices werden zu 1D-Indices
- `unravel_index` macht diesen Prozess rückgängig, sofern die shape des Bildes bekannt ist
- das Bild `result` besteht aus Werten im Bereich $-1, \ldots, 1$
  - -1 steht für invertierte Übereinstimmung
  - 1 steht für Übereinstimmung