

Robotik I: Einführung in die Robotik

Übung 7: Bildverarbeitung

Markus Grotz, Tamim Asfour

Institut für Anthropomatik und Robotik

KIT-Fakultät für Informatik, Institut für Anthropomatik und Robotik (IAR)
Hochperformante Humanoide Technologien (H²T)



■ Aufgabe 1

Farbrepräsentation

■ Aufgabe 3

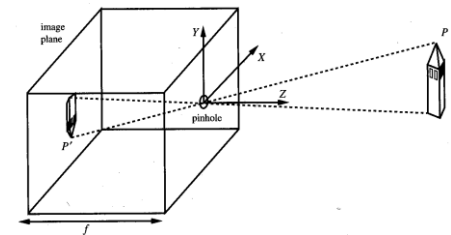
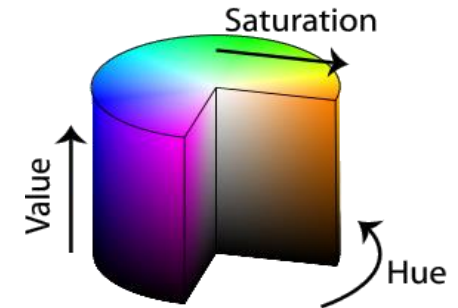
Kameramodell

■ Aufgabe 2

Filter in der Bildverarbeitung

■ Aufgabe 4

Morphologische Operatoren



■ Aufgabe 1

Farbrepräsentation

■ Aufgabe 3

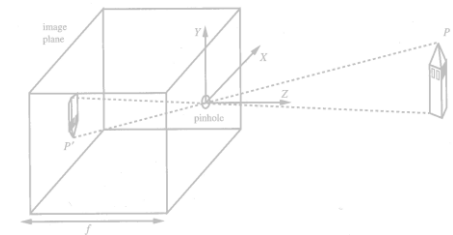
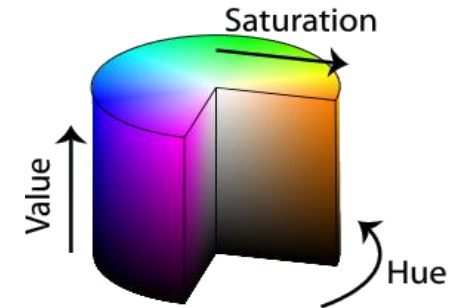
Kameramodell

■ Aufgabe 2

Filter in der Bildverarbeitung

■ Aufgabe 4

Morphologische Operatoren



HSI Farbraum

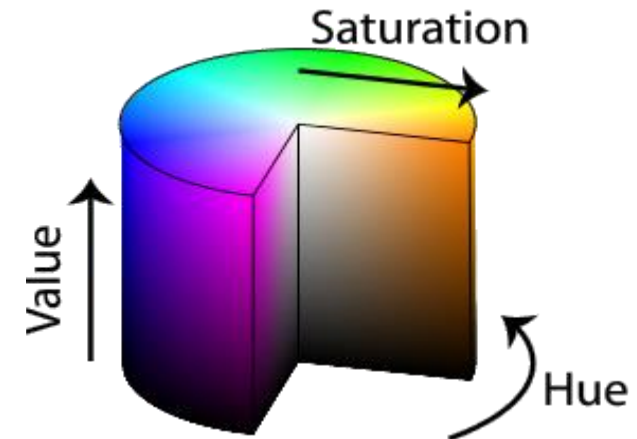
- Transformieren Sie die im RGB-Modell angegebenen Farben (120, 80, 210) und (0, 150, 130) in das HSI-Format.

$$H = \begin{cases} \theta, & \text{falls } B < G \\ 360 - \theta, & \text{sonst} \end{cases}$$

$$\theta = \arccos \frac{2R - G - B}{2\sqrt{(R - G)^2 + (R - B)(G - B)}}$$

$$S = 1 - \frac{3}{R + G + B} \min(R, G, B)$$

$$I = \frac{1}{3} (R + G + B)$$



Vorlesung Kapitel 11 Folie 12

Aufgabe 1

- Transformieren Sie die im RGB-Modell angegebenen Farben (120, 80, 210) und (0, 150, 130) in das HSI-Format.
- Ein Roboter betrachtet eine Müslipackung in einem fensterlosen Labor. Jemand dreht das Licht etwas heller. Ändern sich die R-, G- oder B-Werte der Müslipackung? Ändern sich die H-, S- oder I-Werte der Müslipackung?
- Ihr Haushaltsroboter betrachtet eine Müslipackung in Ihrer Küche. Draußen schiebt sich eine Wolke vor die Sonne. Ändern sich die R-, G- oder B-Werte der Müslipackung? Ändern sich die H-, S- oder I-Werte der Müslipackung?

Hue (120, 80, 210)

Hue (120, 80, 210)

$$\blacksquare c = \cos^{-1}\left(\frac{2R - G - B}{2\sqrt{(R-G)^2 + (R-B)(G-B)}}\right)$$

$$\blacksquare c = \cos^{-1}\left(\frac{240 - 80 - 210}{2\sqrt{(120 - 80)^2 + (120 - 210)(80 - 210)}}\right)$$

$$\blacksquare c = \cos^{-1}\left(\frac{-50}{2\sqrt{1600 + 11700}}\right)$$

$$\blacksquare c \approx \cos^{-1}(0.2168)$$

$$\blacksquare c \approx 102.5^\circ$$

$$\blacksquare \rightarrow H = 360^\circ - c \approx 257.5^\circ, \text{ da } B \geq G$$

Saturation und Intensity (120, 80, 210)

Saturation und Intensity (120, 80, 210)

$$\blacksquare I = \frac{R+G+B}{3}$$

$$\blacksquare I = \frac{410}{3}$$

$$\blacksquare I \approx 136.7$$

$$\blacksquare S = 1 - \frac{3}{R+G+B} \min(R, G, B)$$

$$\blacksquare S = 1 - \frac{3}{120+80+210} 80$$

$$\blacksquare S = 1 - \frac{240}{410}$$

$$\blacksquare S \approx 0.415$$

Hue (0, 150, 130)

Hue (0, 150, 130)

$$\blacksquare c = \cos^{-1}\left(\frac{2R - G - B}{2\sqrt{(R-G)^2 + (R-B)(G-B)}}\right)$$

$$\blacksquare c = \cos^{-1}\left(\frac{-150 - 130}{2\sqrt{(-150)^2 + (0 - 130)(150 - 130)}}\right)$$

$$\blacksquare c = \cos^{-1}\left(\frac{-280}{2\sqrt{22500 + 2600}}\right)$$

$$\blacksquare c \approx \cos^{-1}(-0.9924)$$

$$\blacksquare c \approx 172.9^\circ$$

$$\blacksquare \rightarrow H = c \approx 172.9^\circ, da B < G$$

Saturation und Intensity (0, 150, 130)

Saturation und Intensity (0, 150, 130)

- $I = \frac{R+G+B}{3}$

- $I = \frac{280}{3}$

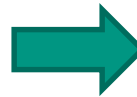
- $I \approx 93.3$

- $S = 1 - \frac{3}{R+G+B} \min(R, G, B)$

- $S = 1$

Farbwerte

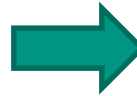
Ein Roboter betrachtet eine Müslipackung in einem fensterlosen Labor. Jemand dreht das Licht etwas heller. Ändern sich die R-, G- oder B-Werte der Müslipackung? Ändern sich die H-, S- oder I-Werte der Müslipackung?



- R, G und B ändern sich gleichmässig, d.h. vergrössern oder verringern sich um den gleichen Faktor. H und S bleiben unverändert, I verändert sich entsprechend der Helligkeitsänderung.

Farbwerte

Ihr Haushaltsroboter betrachtet eine Müslipackung in Ihrer Küche. Draussen schiebt sich eine Wolke vor die Sonne. Ändern sich die R-, G- oder B-Werte der Müslipackung? Ändern sich die H-, S- oder I-Werte der Müslipackung?



- Da sich hier nicht nur die Lichtintensität, sondern auch das Farbspektrum des Lichtes ändert, verändern sich alle Werte, auch H und S - allerdings in den meisten Fällen deutlich weniger als die RGB-Werte.

Anwendung

- Beispiele mit OpenCV
 - <http://opencv.org>
 - Facedetection, Optical Flow, GPU Computing
- Installation mit Python-Bindings:
`apt-get install python-opencv`
- Extraktion des KIT-Logo
- Verwendung des HSV-Farbraums (analog zu HSI)



Nicht klausurrelevant

OpenCV HSV Segmentation

```
import numpy as np  
from matplotlib import pyplot as plt
```

```
import cv2
```

```
bgr_image = cv2.imread('armar-4.png')  
rgb_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2RGB)
```

```
plt.imshow(rgb_image)  
plt.show()
```

```
hsv_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2HSV)
```

Nicht klausurrelevant



OpenCV HSV Segmentation

OpenCV color range for HSV: H: 0-180, S: 0-255, V: 0-255
KIT green is 86.0, 255, 148

```
lower_green = np.array([80, 150, 150])  
upper_green = np.array([90, 255, 255])
```

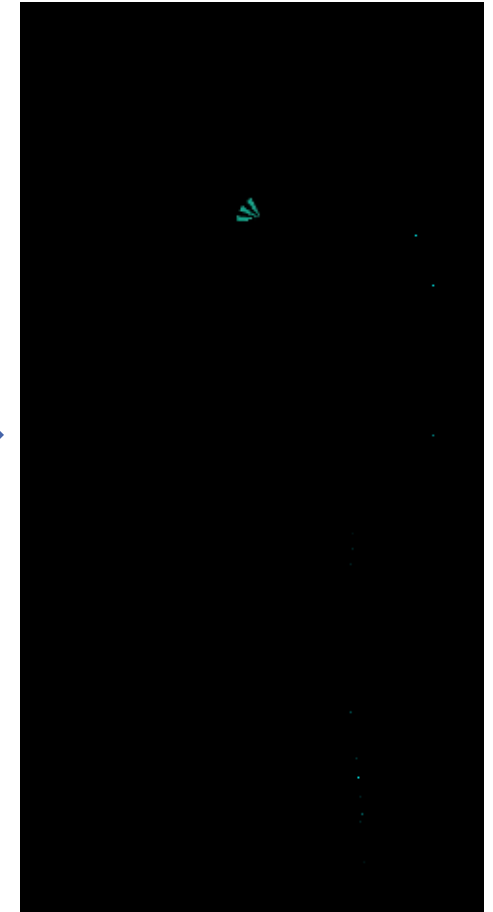
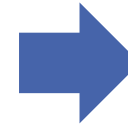
```
mask = cv2.inRange(hsv_image, lower_green, upper_green)  
plt.imshow(mask, cmap='gray')  
plt.show()
```

```
segmented = cv2.bitwise_and(hsv_image, hsv_image, mask=mask)  
result = cv2.cvtColor(segmented, cv2.COLOR_HSV2RGB)
```

```
plt.imshow(result)  
plt.show()
```

Nicht klausurrelevant

OpenCV HSV Segmentation



Nicht klausurrelevant

■ Aufgabe 1

Farbrepräsentation

■ Aufgabe 3

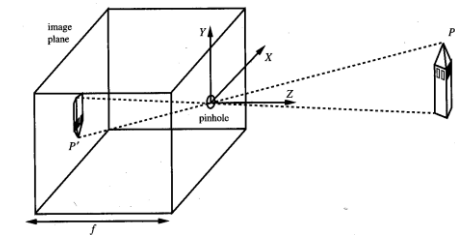
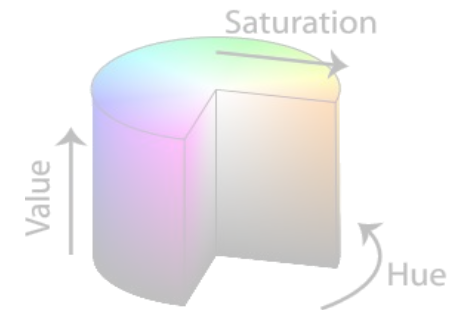
Kameramodell

■ Aufgabe 2

Filter in der Bildverarbeitung

■ Aufgabe 4

Morphologische Operatoren



Aufgabe 3

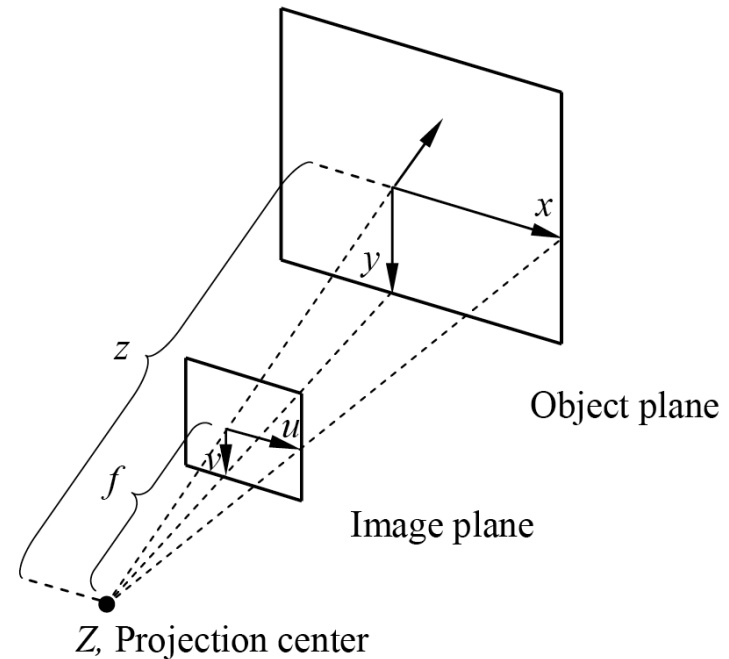
Gegeben sei eine Lochkamera in Positivlage mit Brennweite $f = 20mm$

- Sie fotografieren damit ein Gebäude aus 350 Metern Entfernung. Im Bild ist das Gebäude $0,8\text{ mm}$ hoch. Wie hoch ist das Gebäude in der Welt
- Sie fotografieren den Kölner Dom, der bekanntlich $100\frac{\pi}{2}\text{ m}$ hoch ist, vom gegenüberliegenden Rheinufer aus $800m$ Entfernung. Auf Ihrem Foto ist der Dom 314 Pixel hoch. Wie viele Pixel pro Millimeter hat demnach die Kamera?

Kameramodell

- Gegeben sei eine Lochkamera in Positivlage mit Brennweite $f = 20\text{mm}$

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{f}{z} \begin{pmatrix} x \\ y \end{pmatrix}$$



Vorlesung Kapitel 11 Folie 20

Kameramodell

Sie fotografieren damit ein Gebäude aus 350 *Metern* Entfernung. Im Bild ist das Gebäude 0,8 mm hoch. Wie hoch ist das Gebäude in der Welt?

Kameramodell

Sie fotografieren damit ein Gebäude aus 350 *Metern* Entfernung. Im Bild ist das Gebäude 0,8mm hoch. Wie hoch ist das Gebäude in der Welt?

- $v = \frac{f}{z} y$

- $y = v \frac{z}{f}$

- $y = 0.8mm \cdot \frac{350m}{20mm}$

- $y = 0.8 \cdot \frac{350}{20}$

- $y = 0.4 \cdot 35m = 14m$

Kameramodell

Sie fotografieren den Kölner Dom, der bekanntlich $100\frac{\pi}{2}m$ hoch ist, vom gegenüberliegenden Rheinufer aus $800m$ Entfernung. Auf Ihrem Foto ist der Dom 314 Pixel hoch. Wieviele Pixel pro Millimeter hat demnach die Kamera?

Kameramodell

Sie fotografieren den Kölner Dom, der bekanntlich $100 \frac{\pi}{2} m$ hoch ist, vom gegenüberliegenden Rheinufer aus $800m$ Entfernung. Auf Ihrem Foto ist der Dom 314 Pixel hoch. Wieviele Pixel pro Millimeter hat demnach die Kamera?

- $100 \frac{\pi}{2} m \approx 157m$

- $314px = \frac{20mm}{800m} 157m$

- $314px = \frac{20mm}{800} 157$

- $2px = \frac{20mm}{800}$

- $2px = \frac{1mm}{40}$

- $80px = 1mm$

■ Aufgabe 1

Farbrepräsentation

■ Aufgabe 3

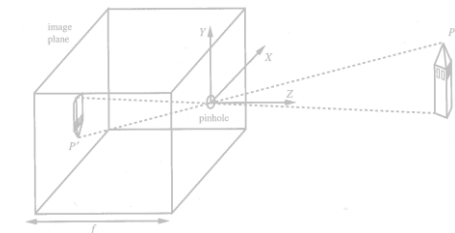
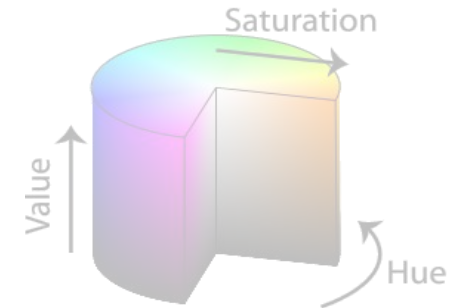
Kameramodell

■ Aufgabe 2

Filter in der Bildverarbeitung

■ Aufgabe 4

Morphologische Operatoren



■ Prewitt-X Filter

$$P_x = \frac{\partial g(x, y)}{\partial x}$$

- Approximiert durch

$$p_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

■ Prewitt-Y Filter

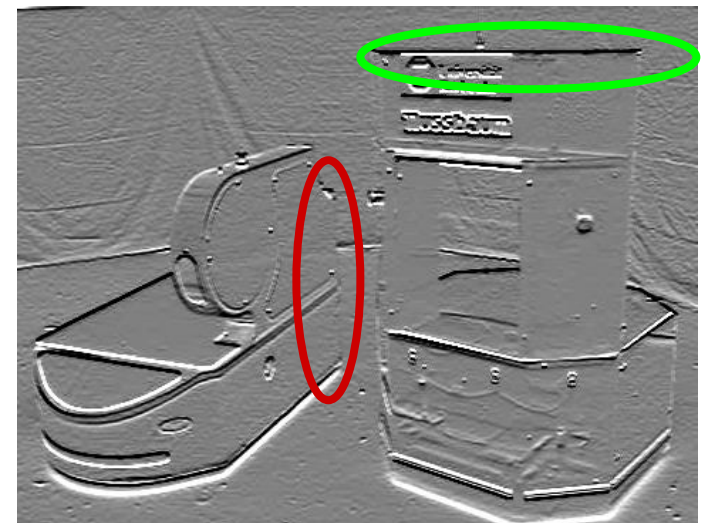
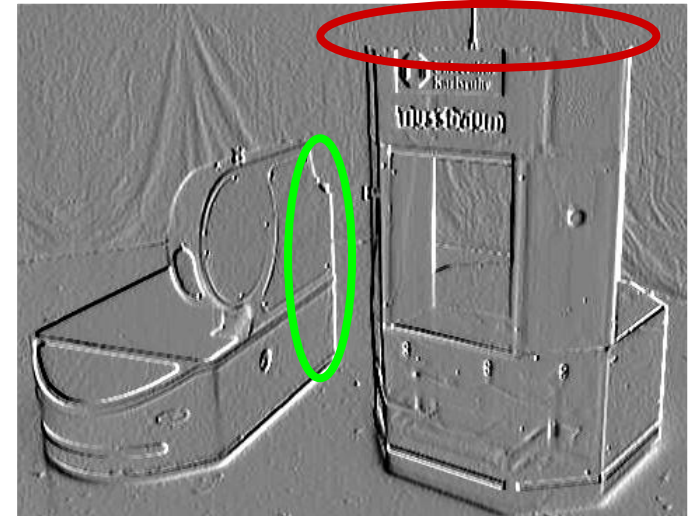
$$P_y = \frac{\partial g(x, y)}{\partial y}$$

- Approximiert durch

$$p_y = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

■ Eigenschaften:

- Gute Ergebnisse bei Detektion von vertikalen bzw. horizontalen Kanten



Prewitt Filter

Gegeben sei das folgende Graustufenbild B

$$B = \begin{pmatrix} 20 & 20 & 20 & 20 & 40 & 40 & 40 & 40 & 30 & 30 & 30 \\ 20 & 20 & 20 & 20 & 40 & 40 & 40 & 40 & 30 & 30 & 30 \\ 20 & 20 & 20 & 20 & 40 & 40 & 40 & 40 & 30 & 30 & 30 \\ 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 & 20 & 20 & 20 \\ 20 & 50 & 50 & 50 & 50 & 50 & 50 & 50 & 20 & 20 & 20 \\ 20 & 20 & 50 & 50 & 50 & 50 & 50 & 50 & 20 & 20 & 20 \\ 20 & 20 & 20 & 50 & 50 & 50 & 50 & 50 & 20 & 20 & 20 \end{pmatrix}$$

- Berechnen Sie das Ergebnis der Filterung von B mit dem Prewitt-Kantenfilter in x-Richtung.

- Berechnen Sie das Ergebnis der Filterung von B mit dem Prewitt-Kantenfilter in x-Richtung.

Prewitt Filter

$$\blacksquare B_x(2,2) = -1 B(1,1) + 0 B(1,2) + 1 B(1,3) - 1(B(2,1) + 0 B(2,2) + 1 B(2,3) - 1 B(3,1) + 0 B(3,2) + 1 B(3,3))$$

$$= -20 + 0 + 20 - 20 + 0 + 20 - 20 + 0 + 20$$

$$= 0$$

$$\blacksquare B_x(2,3) = -20 + 0 + 20 - 20 + 0 + 20 - 20 + 0 + 20 = 0$$

$$\blacksquare B_x(2,4) = -20 + 0 + 40 - 20 + 0 + 40 - 20 + 0 + 40 = 60$$

Prewitt Filter

$$\blacksquare B_x = \begin{pmatrix} 0 & 0 & 60 & 60 & 0 & 0 & -30 & -30 & 0 \\ 0 & 0 & 40 & 40 & 0 & 0 & -50 & -50 & 0 \\ 30 & 0 & 20 & 20 & 0 & 0 & -70 & -70 & 0 \\ 60 & 30 & 0 & 0 & 0 & 0 & -90 & -90 & 0 \\ 60 & 60 & 30 & 0 & 0 & 0 & -90 & -90 & 0 \end{pmatrix}$$

$$\blacksquare B_y = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 90 & 90 & 70 & 50 & 30 & 30 & 10 & -10 & -30 \\ 60 & 90 & 70 & 50 & 30 & 30 & 10 & -10 & -30 \\ -60 & -30 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -60 & -60 & -30 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

OpenCV Canny Edge Detection

```
import cv2
from matplotlib import pyplot as plt

# cv2.CV_LOAD_IMAGE_GRAYSCALE = 0
gray_image = cv2.imread('armar-4.png', 0)

result_canny = cv2.Canny(gray_image, 100, 200)

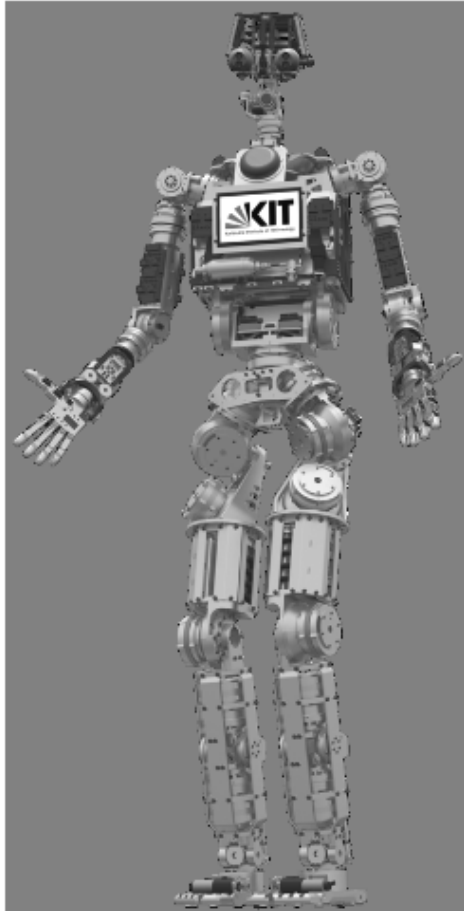
fig, ax = plt.subplots(1, 2)
ax[0].imshow(gray_image, cmap='gray')
ax[0].set_title('input image')

ax[1].imshow(result_canny, cmap='gray')
ax[1].set_title('canny edge detection')
plt.show()
```

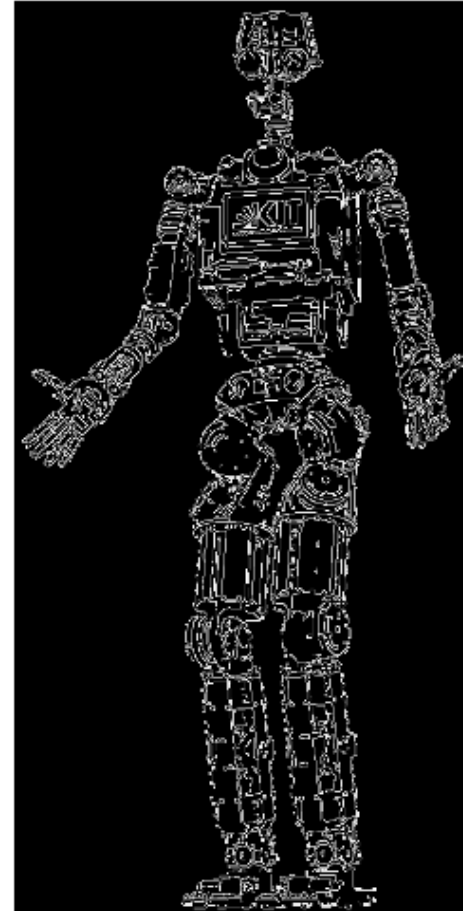
Nicht klausurrelevant

OpenCV Canny Edge Detection

input image



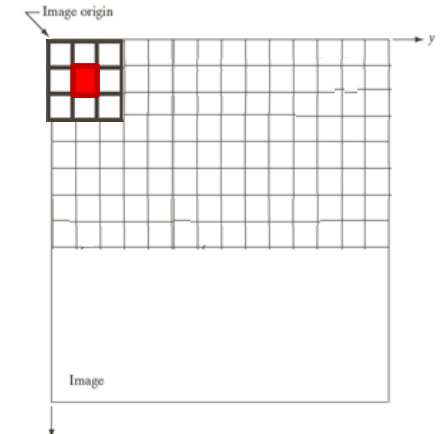
canny edge detection



Nicht klausurrelevant

Filteroperationen - Ränder

- Was passiert an den Ränder?
 - Konstanter Wert (**Constant**),
z.B. 0: Bild wird an den Rändern auf 0 gesetzt.
 - Umschlingen (**Wrap**):
Bild wird „fortgesetzt“



Constant

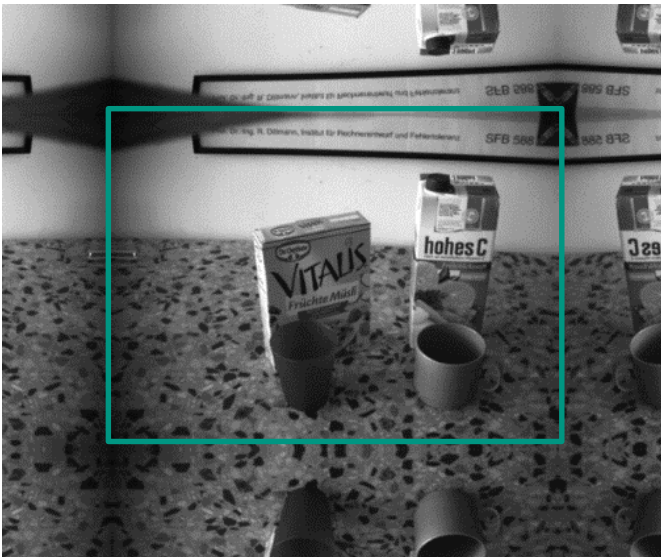
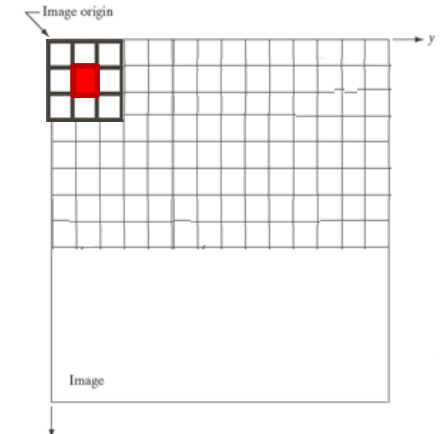


Wrap

Filteroperationen - Ränder

■ Was passiert an den Ränder?

- Spiegeln (**Mirror**, **Reflect**):
Bild wird an den Rändern gespiegelt
- Wiederholen (**Clamp**, **Replicate**):
Nehme letzten Wert



Mirror



Clamp

Filteroperationen - Ränder

- Was passiert an den Ränder?
- Beispiel: Median Filter mit

$$\frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

- Eingabebild

$$\begin{pmatrix} 30 & 60 & 90 \\ 120 & 0 & 150 \\ 180 & 210 & 240 \end{pmatrix}$$

Filteroperationen - Ränder

Konstant:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 30 & 60 & 90 & 0 \\ 0 & 120 & 0 & 150 & 0 \\ 0 & 180 & 210 & 240 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Ergebnis:

$$\frac{1}{9} \begin{pmatrix} 210 & 450 & 300 \\ 600 & 1080 & 750 \\ 510 & 900 & 600 \end{pmatrix}$$

Filteroperationen - Ränder

Wrap:

$$\begin{pmatrix} 240 & 180 & 210 & 240 & 180 \\ 90 & 30 & 60 & 90 & 30 \\ 150 & 120 & 0 & 150 & 120 \\ 240 & 180 & 210 & 240 & 180 \\ 90 & 30 & 60 & 90 & 30 \end{pmatrix}$$

Ergebnis:

$$\frac{1}{9} \begin{pmatrix} 1080 & 1080 & 1080 \\ 1080 & 1080 & 1080 \\ 1080 & 1080 & 1080 \end{pmatrix}$$

Filteroperationen - Ränder

Mirror/Clamp:

$$\begin{pmatrix} 30 & 30 & 60 & 90 & 90 \\ 30 & 30 & 60 & 90 & 90 \\ 120 & 120 & 0 & 150 & 150 \\ 180 & 180 & 210 & 240 & 240 \\ 180 & 180 & 210 & 240 & 240 \end{pmatrix}$$

Ergebnis:

$$\frac{1}{9} \begin{pmatrix} 480 & 630 & 780 \\ 930 & 1080 & 1230 \\ 1380 & 1530 & 1680 \end{pmatrix}$$

Filteroperationen - Ränder

Ignorieren:

$$\begin{pmatrix} 30 & 60 & 90 \\ 120 & 0 & 150 \\ 180 & 210 & 240 \end{pmatrix}$$

Ergebnis:

$$\frac{1}{9} \begin{pmatrix} X & X & X \\ X & 1080 & X \\ X & X & X \end{pmatrix}$$

■ Aufgabe 1

Farbrepräsentation

■ Aufgabe 3

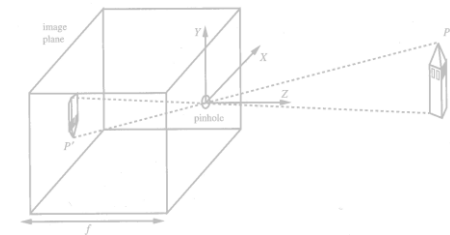
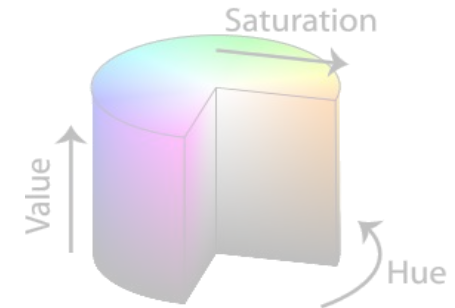
Kameramodell

■ Aufgabe 2

Filter in der Bildverarbeitung

■ Aufgabe 4

Morphologische Operatoren



Morphologische Operatoren: Öffnen & Schließen

Öffnen:

- Anwendung von Erosion danach Dilatation
- Entfernt dünne Stege oder kleine außenliegende Objekte

Schließen:

- Anwendung von Dilatation danach Erosion
- Überbrückung kleiner Distanzen und Schließung von inneren Löchern

Morphologische Operatoren

- Gegeben sei das folgende Graustufenbild:

$$\begin{pmatrix} 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 0 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \end{pmatrix}$$

- sowie das strukturierende Element

$$\begin{pmatrix} 255 & 255 & 255 \\ 255 & 255 & 255 \\ 255 & 255 & 255 \end{pmatrix}$$

- Wenden Sie den morphologischen Operator *Öffnen* auf das Eingabebild an. Ignorieren Sie im Ergebnisbild die Randpixel

Morphologische Operatoren: Erosion

■ Alle Werte auf 0 setzen.

■ Teilergebnis

■ $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

Algorithmus 20 Erosion(I, n) $\rightarrow I'$

$k := \text{div}((n - 1), 2)$

for all pixels (u, v) in I' **do**

$I'(u, v) := 0$

end for

for $v := k$ **to** $h - k - 1$ **do**

for $u := k$ **to** $w - k - 1$ **do**

if $I(u, v) = q$ **then**

for $i := -k$ **to** k **do**

for $j := -k$ **to** k **do**

if $I(u + j, v + i) \neq q$ **then**

goto NEXT

end if

end for

end for

$I'(u, v) = q$

NEXT:

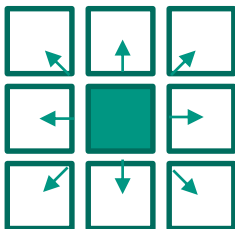
end if

end for

end for

Morphologische Operatoren: Erosion

- Nachbarumgebungen prüfen
- Strukturierendes Element definiert dabei die Umgebung.
- Im Algorithmus vereinfachtes Modell



Algorithmus 20 Erosion(I, n) $\rightarrow I'$

```

 $k := \text{div}((n - 1), 2)$ 
for all pixels  $(u, v)$  in  $I'$  do
     $I'(u, v) := 0$ 
end for
for  $v := k$  to  $h - k - 1$  do
    for  $u := k$  to  $w - k - 1$  do
        if  $I(u, v) = q$  then
            for  $i := -k$  to  $k$  do
                for  $j := -k$  to  $k$  do
                    if  $I(u + j, v + i) \neq q$  then
                        goto NEXT
                    end if
                end for
            end for
             $I'(u, v) = q$ 
        end if
    end for
    NEXT:
end for
end for

```

Morphologische Operatoren: Erosion

■ Eingabebild:

$$\begin{pmatrix} 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 0 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \end{pmatrix}$$

■ Zwischenergebnis nach der ersten Iteration:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Morphologische Operatoren: Erosion

■ Eingabebild:

$$\begin{pmatrix} 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 0 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \end{pmatrix}$$

■ Zwischenergebnis nach der zweiten Iteration:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Morphologische Operatoren: Erosion

■ Eingabebild:

$$\begin{pmatrix} 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 0 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \end{pmatrix}$$

■ Zwischenergebnis nach der dritten Iteration:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Morphologische Operatoren: Erosion

■ Eingabebild:

$$\begin{pmatrix} 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 0 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \end{pmatrix}$$

■ Zwischenergebnis nach der vierten Iteration:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 255 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Morphologische Operatoren: Erosion

■ Eingabebild:

$$\begin{pmatrix} 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 0 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 & 255 & 255 \end{pmatrix}$$

■ Ergebnis:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 255 & 255 & 255 & 0 \\ 0 & 0 & 0 & 0 & 255 & 255 & 255 & 0 \\ 0 & 0 & 0 & 0 & 255 & 255 & 255 & 0 \\ 0 & 0 & 0 & 0 & 255 & 255 & 255 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Morphologische Operatoren: Öffnen

■ Öffnen = Erosion dann Dilatation

■ Eingabebild:

$$\begin{pmatrix} 0 & 0 & 0 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 \end{pmatrix}$$

■ Ergebnis:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 255 & 255 & 255 & 0 \\ 0 & 0 & 255 & 255 & 255 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Morphologische Operatoren: Dilatation

■ Alle Werte auf 0 setzen.

■ Teilergebnis

■
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Algorithmus 19 Dilatation(I, n) $\rightarrow I'$

```

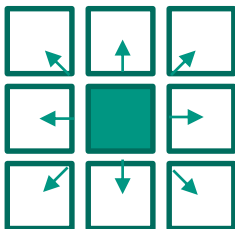
 $k := \text{div}((n - 1), 2)$ 
for all pixels  $(u, v)$  in  $I'$  do
     $I'(u, v) := 0$ 
end for

for  $v := k$  to  $h - k - 1$  do
    for  $u := k$  to  $w - k - 1$  do
        if  $I(u, v) = q$  then
            for  $i := -k$  to  $k$  do
                for  $j := -k$  to  $k$  do
                     $I'(u + j, v + i) = q$ 
                end for
            end for
        end if
    end for
end for

```

Morphologische Operatoren: Dilatation

- Analog zur Erosion
- Nachbarumgebungen prüfen
- Strukturierendes Element definiert dabei die Umgebung.
- Im Algorithmus vereinfachtes Modell



Algorithmus 19 Dilatation(I, n) $\rightarrow I'$

```

 $k := \text{div}((n - 1), 2)$ 
for all pixels  $(u, v)$  in  $I'$  do
     $I'(u, v) := 0$ 
end for
for  $v := k$  to  $h - k - 1$  do
    for  $u := k$  to  $w - k - 1$  do
        if  $I(u, v) = q$  then
            for  $i := -k$  to  $k$  do
                for  $j := -k$  to  $k$  do
                     $I'(u + j, v + i) = q$ 
                end for
            end for
        end if
    end for
end for

```

Morphologische Operatoren: Öffnen

■ Öffnen = Erosion dann Dilatation

■ Eingabebild:
$$\begin{pmatrix} 0 & 0 & 0 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 \end{pmatrix}$$

■ Zwischenergebnis nach der ersten Iteration:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Morphologische Operatoren: Öffnen

■ Öffnen = Erosion dann Dilatation

■ Eingabebild:
$$\begin{pmatrix} 0 & 0 & 0 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 \end{pmatrix}$$

■ Zwischenergebnis nach der zweiten Iteration:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 255 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Morphologische Operatoren: Öffnen

■ Öffnen = Erosion dann Dilatation

■ Eingabebild:
$$\begin{pmatrix} 0 & 0 & 0 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 \\ 0 & 0 & 0 & 255 & 255 & 255 \end{pmatrix}$$

■ Ergebnis:

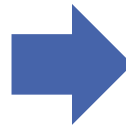
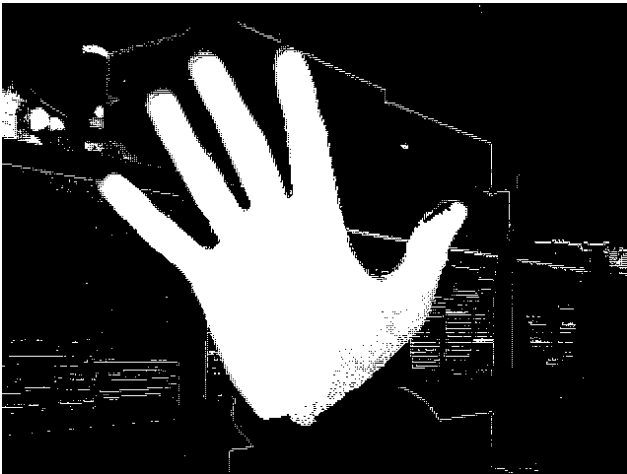
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 255 & 255 & 255 & 0 \\ 0 & 0 & 255 & 255 & 255 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

OpenCV Morphologische Operatoren

```
gray_image = cv2.imread('hand_segmented.png', 0)
```

```
kernel = np.ones((10, 10), dtype=np.uint8) * 255  
result_image = cv2.morphologyEx(src, cv2.MORPH_OPEN, kernel)
```

```
plt.imshow(result_image, cmap='gray')  
plt.show()
```



Nicht klausurrelevant