

Klassifikation und Farbe

Christian Wilms

Computer Vision Group
Universität Hamburg

Sommersemester 2018

19. April 2018

Nachtrag - Einlesen von Bildern

Alt: Mit der Bibliothek scipy Bilder einlesen.

```
>>> from scipy import misc  
>>> img = misc.imread("./icon.png")  
>>> # your magic here  
>>> misc.imsave("./iconModified.png", img)
```

Neu: Mit der Bibliothek scikit-image Bilder einlesen.

```
>>> from skimage.io import imread, imsave  
>>> img = imread("./icon.png")  
>>> # your magic here  
>>> imsave("./iconModified.png", img)
```

Klassifikation - Ein kleines Experiment I



Wir haben Bilder u. drei Klassen. Aber welches Bild gehört wozu?

Klassifikation - Ein kleines Experiment I



Was brauchen wir, um die Bilder den Klassen zuzuordnen?

Klassifikation - Ein kleines Experiment II

Was wir brauchen:

- Bsp. für alle Klassen → Bilder mit Label bzw. Trainingsmenge
 - etwas, das die Klassen beschreibt → Merkmale/Deskriptoren
 - einen Vergleich dieser Deskriptoren → Metrik
 - ein Verfahren, das mit Hilfe der ganzen Infos neue Bilder den Klassen zuordnet → Klassifikator

Klassifikation - Ein kleines Experiment II

Was wir brauchen:

- Bsp. für alle Klassen → Bilder mit Label bzw. Trainingsmenge
 - etwas, das die Klassen beschreibt → Merkmale/Deskriptoren
 - einen Vergleich dieser Deskriptoren → Metrik
 - ein Verfahren, das mit Hilfe der ganzen Infos neue Bilder den Klassen zuordnet → Klassifikator

Klassifikation

Klassifikation ist ein Prozess bei dem ein Objekt anhand von Merkmalen zu einer bestimmten Klasse zugeordnet wird.

Klassifikation - Ein kleines Experiment II

Was wir brauchen:

- Bsp. für alle Klassen → Bilder mit Label bzw. Trainingsmenge
 - etwas, das die Klassen beschreibt → Merkmale/Deskriptoren
 - einen Vergleich dieser Deskriptoren → Metrik
 - ein Verfahren, das mit Hilfe der ganzen Infos neue Bilder den Klassen zuordnet → Klassifikator

Was haben wir:

Bilder

Deskriptor

Metrik

Klassifikator

Klassifikation - Ein kleines Experiment I

Was wir brauchen:

- Bsp. für alle Klassen → Bilder mit Label bzw. Trainingsmenge
 - etwas, das die Klassen beschreibt → Merkmale/Deskriptoren
 - einen Vergleich dieser Deskriptoren → Metrik
 - ein Verfahren, das mit Hilfe der ganzen Infos neue Bilder den Klassen zuordnet → Klassifikator

Was haben wir:

Bilder trainingsDaten.npz und
validierungsDaten.npz im CommSy

Deskriptor

Metrik

Klassifikator

Klassifikation - Ein kleines Experiment I

Was wir brauchen:

- Bsp. für alle Klassen → Bilder mit Label bzw. Trainingsmenge
 - etwas, das die Klassen beschreibt → Merkmale/Deskriptoren
 - einen Vergleich dieser Deskriptoren → Metrik
 - ein Verfahren, das mit Hilfe der ganzen Infos neue Bilder den Klassen zuordnet → Klassifikator

Was haben wir:

Bilder trainingsDaten.npz und
validierungsDaten.npz im CommSy

Deskriptor Bildstatistiken und Histogramm

Metrik

Klassifikator

Klassifikation - Ein kleines Experiment I

Was wir brauchen:

- Bsp. für alle Klassen → Bilder mit Label bzw. Trainingsmenge
 - etwas, das die Klassen beschreibt → Merkmale/Deskriptoren
 - einen Vergleich dieser Deskriptoren → Metrik
 - ein Verfahren, das mit Hilfe der ganzen Infos neue Bilder den Klassen zuordnet → Klassifikator

Was haben wir:

Bilder trainingsDaten.npz und
validierungsDaten.npz im CommSy

Deskriptor Bildstatistiken und Histogramm

Metrik L_2 -Distanz $\sqrt{\sum_{i=0}^n (\vec{u}_i - \vec{v}_i)^2}$

Klassifikator

Klassifikation - Ein kleines Experiment I

Was wir brauchen:

- Bsp. für alle Klassen → Bilder mit Label bzw. Trainingsmenge
 - etwas, das die Klassen beschreibt → Merkmale/Deskriptoren
 - einen Vergleich dieser Deskriptoren → Metrik
 - ein Verfahren, das mit Hilfe der ganzen Infos neue Bilder den Klassen zuordnet → Klassifikator

Was haben wir:

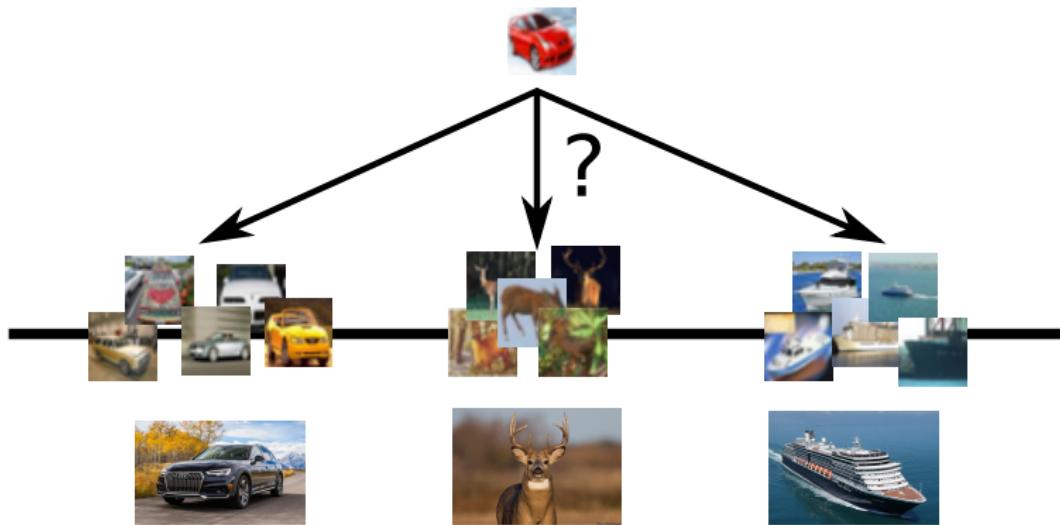
Bilder sind im CommSy

Deskriptor Bildstatistiken und Histogramm

Metrik L_2 -Distanz $\sqrt{\sum_{i=0}^n (\vec{u}_i - \vec{v}_i)^2}$

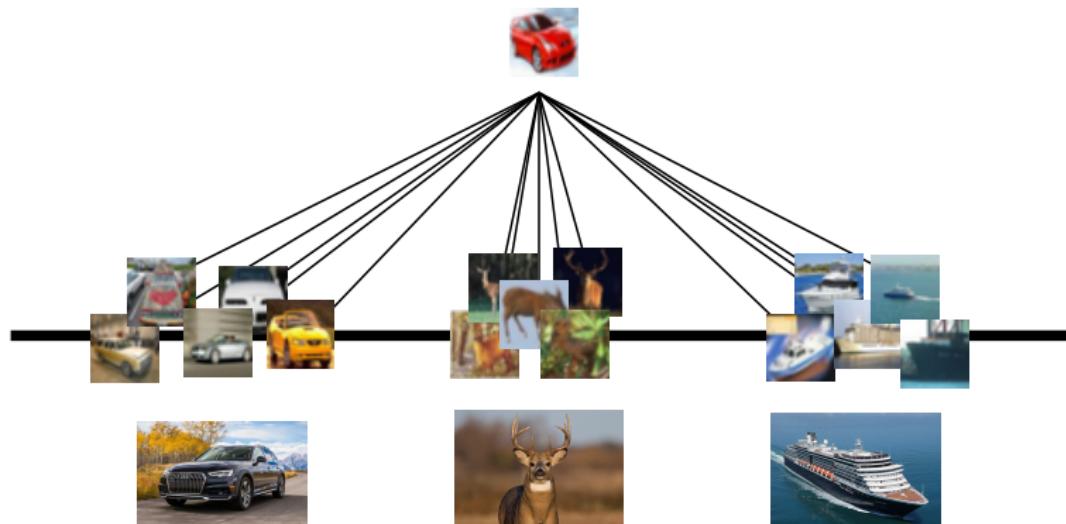
Klassifikator Welcher Deskriptor eines Bildes mit Label (Trainingsbilder) liegt am nächsten?

Klassifikation - Ein kleines Experiment III



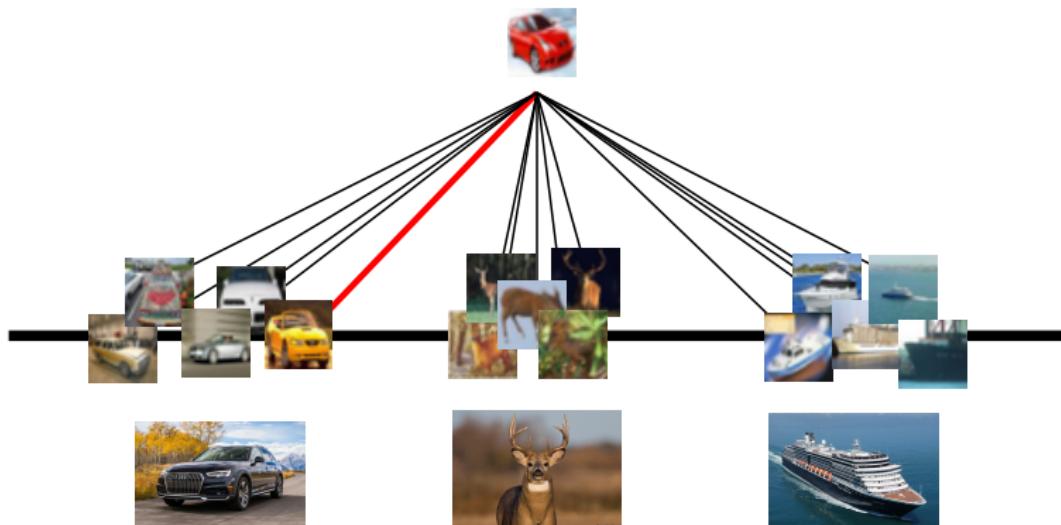
Zu welcher der drei Klassen gehört das Bild?

Klassifikation - Ein kleines Experiment III



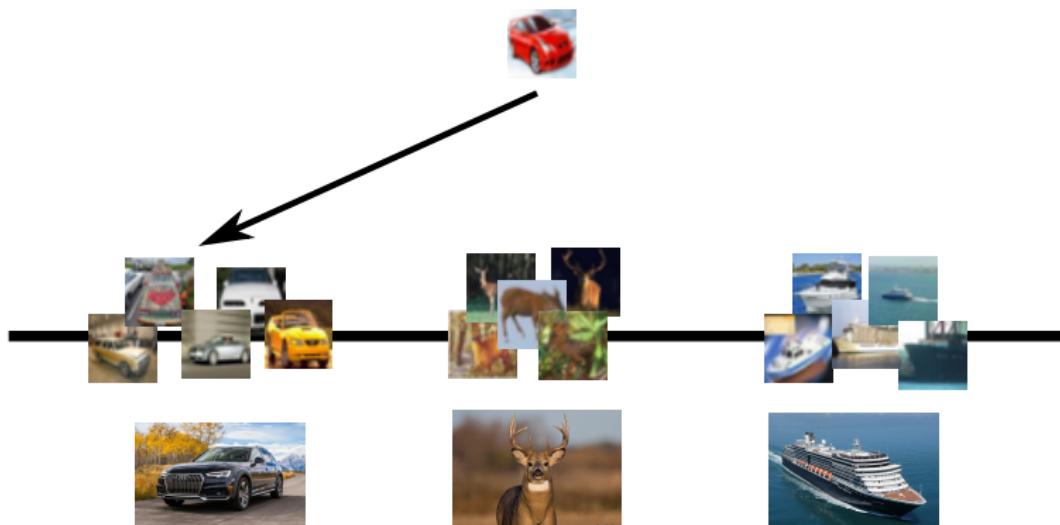
Berechne die Distanz zu allen Trainingsbildern **einzel** (Deskriptor + Metrik).

Klassifikation - Ein kleines Experiment III



Das Label des Bildes mit der geringsten Distanz bestimmt das Label des neuen Bildes.

Klassifikation - Ein kleines Experiment III



Es ist ein Auto!

Deskriptoren mit einem Merkmal

Merkmal = Deskriptor

Beispiele

Mittelwert Deskriptor besteht nur aus einer Zahl → (42)

Standardabweichung Deskriptor besteht nur aus einer Zahl → (23)

Histogramm Deskriptor besteht nur aus dem Histogramm

$$(10 \quad 15 \quad 20 \quad 2)^T$$

Vergleich der Deskriptoren bspw. über euklidische Distanz.

Mehrere Merkmale zu einem Deskriptor kombinieren

Möglichkeit 1: Zusammen betrachten

- Merkmale konkatenieren → (42 23) und (12 26)
- Distanz direkt zwischen den gesamten Deskriptoren berechnen:

$$\Delta = \sqrt{(42 - 12)^2 + (23 - 26)^2}$$

- Nachteile: Gewichtung nicht so einfach möglich + alle Merkmale mit der selben Metrik/Distanz verglichen

Möglichkeit 2: Einzeln betrachten

Mehrere Merkmale zu einem Deskriptor kombinieren

Möglichkeit 1: Zusammen betrachten

Möglichkeit 2: Einzeln betrachten

- Merkmale nicht konkatenieren sondern getrennt speichern (virtueller Deskriptor)
- Distanz zwischen den Teil-Deskriptoren einzeln berechnen und gewichten:

$$\Delta = \sqrt{(42 - 12)^2} + w \cdot \sqrt{(23 - 26)^2}$$

- Vorteile: Gewichtung und unterschiedliche Metriken/Distanzen möglich

Grober Ablauf

- ① Trainingsdaten und Validierungsdaten laden
- ② für alle Trainingsdaten Merkmale berechnen
- ③ ggf. Merkmale zu einem Deskriptor (z.B. Liste oder Array von Zahlen) kombinieren
- ④ ebenso für alle Validierungsdaten
- ⑤ für jedes Bild der Validierungsdaten den Deskriptor mit allen Deskriptoren der Trainingsdaten abgleichen und jeweils Label des am nächsten gelegenen Trainingsbildes merken

Daten laden

- die Daten liegen als .npz-Datei vor, je eine für die Trainings- bzw. die Validierungsdaten
- zwei Dicts für die Bilder ('data') und die Labels ('label')
- Bilder sind zusammen gestapelt in einem Array der Größe $(60 \times 32 \times 32)$ bzw. $(30 \times 32 \times 32)$
- Labels als 1D-Arrays mit Zahlen (Labels der Validierungsdaten nur zur Evaluation nutzen!)

```
>>> d = np.load('./trainingsDaten.npz')
>>> trImg = d['data']
>>> trLabel = d['labels']

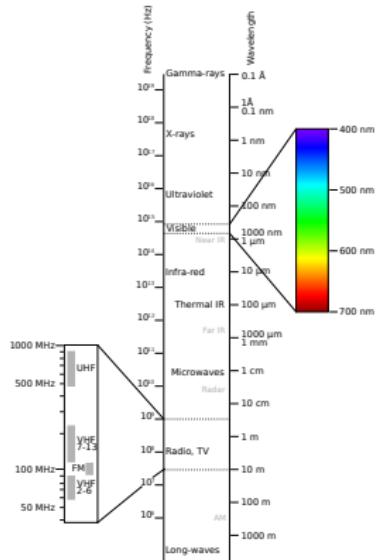
>>> img1 = trImg[0,:,:,:]
>>> label1 = trLabel[0]
```

Aufgaben

Löst das Aufgabenblatt 3.1!

Beachtet auch das Readme zu den Daten!

Wie entstehen Farben und wie unterscheiden sie sich?

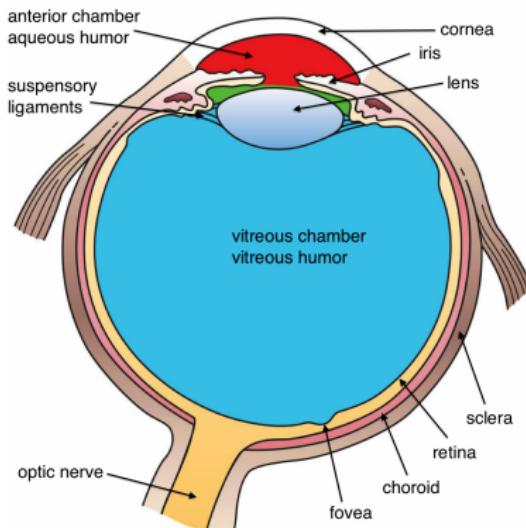


EM Spektrum¹

- Licht besteht aus verschiedenen Frequenzen des elektromagnetischen Spektrums
- der Mensch kann nur den Bereich zwischen ca. 400 nm und ca. 700 nm wahrnehmen (sichtbares Spektrum)
- jede wahrnehmbare Farbe kann durch Mischungen der drei Primärfarben (Rot, Grün und Blau) erzeugt werden (Dreifarbentheorie)

¹<https://commons.wikimedia.org/wiki/File:Electromagnetic-Spectrum.svg>

Wie nimmt der Mensch Farben wahr?

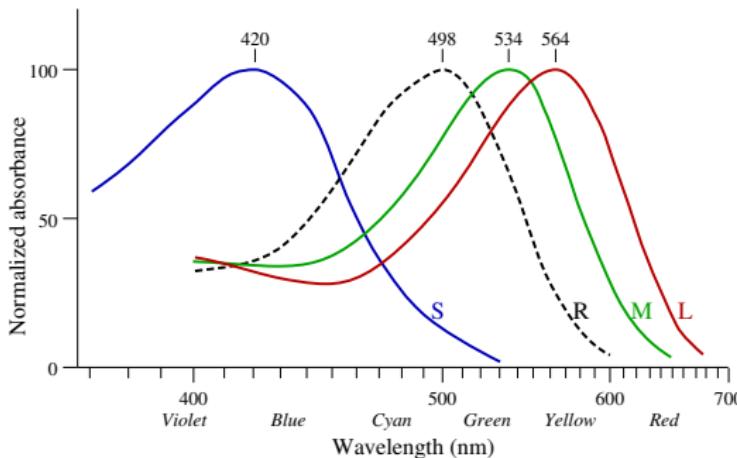


menschliches Auge¹

- Licht wird über die Stäbchen und Zapfen in der Netzhaut wahrgenommen
- 120 Millionen Stäbchen für Helligkeitswahrnehmung
- 7 Millionen Zapfen für Farbwahrnehmung
 - S-Zapfen für blaues Licht
 - M-Zapfen für grünes Licht
 - L-Zapfen für rotes Licht
- Stäbchen sind verteilt, Zapfen in der Fovea konzentriert

¹en.wikipedia.org/wiki/File:Three_Main_Layers_of_the_Eye.png

Empfindlichkeit der Zapfentypen



Verhältnis der Aktivierungen beschreibt eine Farbe.¹

¹<https://commons.wikimedia.org/wiki/File:Cone-response.svg>

Wie können Farben systematisiert werden?

Möglichkeit 1

- Nutzung der Dreifarbtentheorie
- jede Farbe aus Mischung der drei Grundfarben darstellen
- Farbe als Kombination aus rotem, grünem und blauem Licht (oder cyan, gelben und magenta Pigmenten)
- Vektor von drei Zahlen

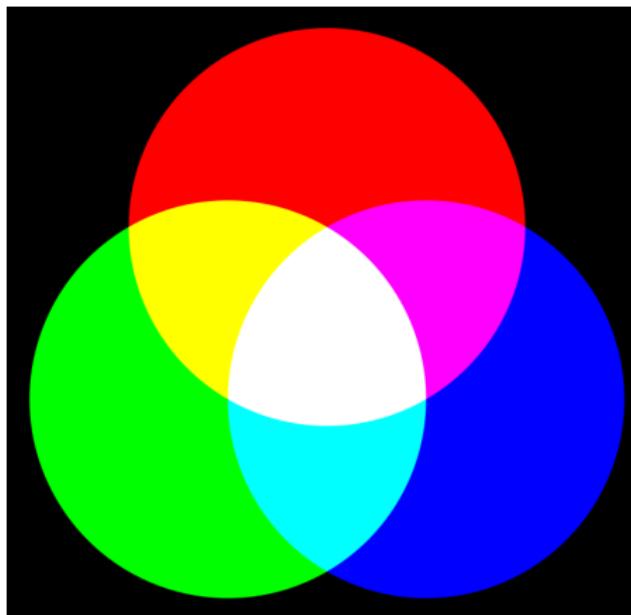
⇒ RGB-Farbraum

Möglichkeit 2

- der menschlichen Beschreibung folgen
- drei Eigenschaften je Farbe
 - Farbton als Winkel im Farbkreis
 - Reinheit/Sättigung als Maß für die Anzahl der versch. Frequenzen in einer Farbe
 - Helligkeit
- Vektor von drei Zahlen

⇒ HSV-Farbraum

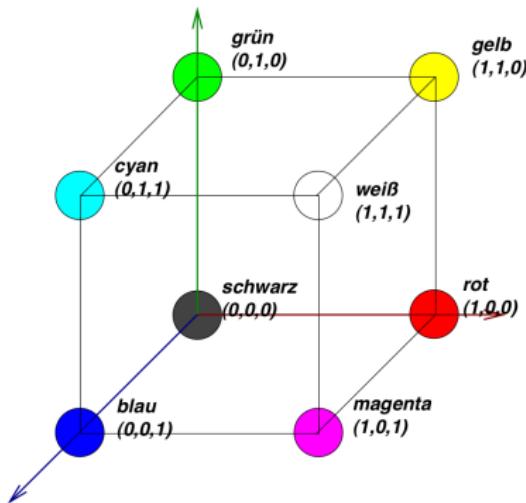
Additives Farbmodell



Mischung von Licht versch. Frequenzen/Farben ergibt neue Farben.¹

¹<https://commons.wikimedia.org/wiki/File:Synthese+.svg>

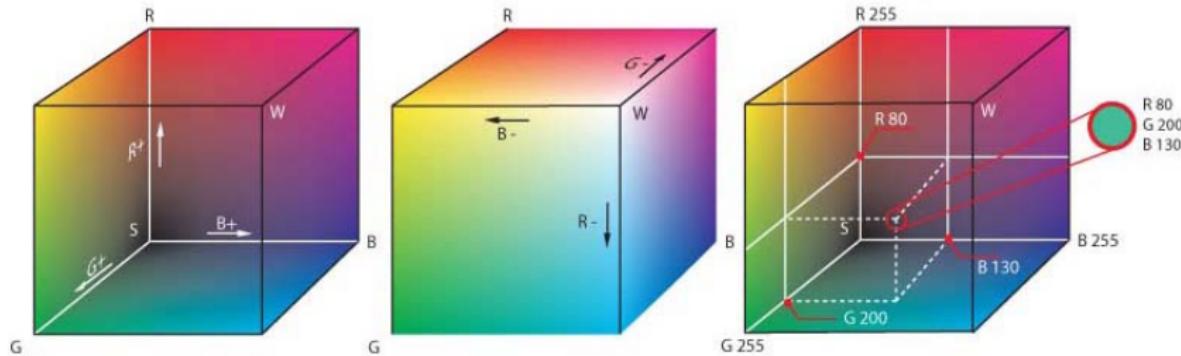
RGB-Farbraum - I



RGB-Einheitswürfel

- Farbe als Punkt im durch Rot, Grün und Blau aufgespannten Raum
- Schwarz ist am Ursprung, Weiß am gegenüberliegen Ende
- auf der Diagonalen von Schwarz zu Weiß liegen die Graustufen (253 verschiedene)
- je Achse gibt es 255 verschiedene Werte ⇒ insgesamt $255^3 = 16.581.375$ verschiedene Farben
- Anwendung etwa bei Bildschirmen

RGB-Farbraum - II



RGB-Farbwürfel (Achsen vertauscht)¹

¹https://de.wikipedia.org/wiki/Datei:RGB_farbwuergel.jpg

RGB-Farbraum - III



R+G+B



R



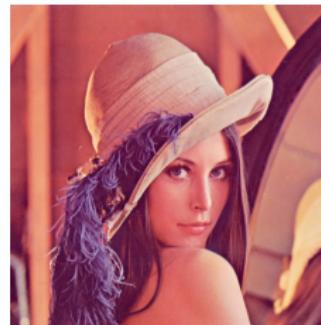
G



B

- Bilder bestehen aus drei Kanälen/Bändern (Rot, Grün, Blau)
- jeder Kanal gibt Auskunft über den Einfluss einer der drei Grundfarben auf jedes Pixel
- jedes Pixel ist ein Vektor von 3 Werten (R, G, B)
- das Bild wird so zu einer 3D-Struktur (Höhe × Breite × Kanäle)

RGB-Bilder in Python - I



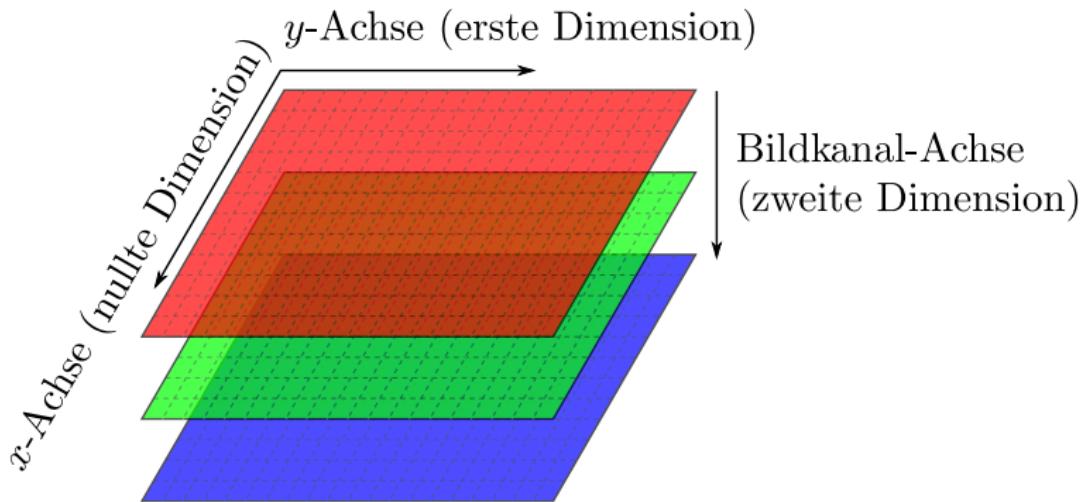
```
>>> img
array([[[226, 137, 125], ..., [200, 99, 90]],
       [[226, 137, 125], ..., [200, 99, 90]],
       ...,
       [[ 84, 18, 60], ..., [177, 62, 79]],
       [[ 82, 22, 57], ..., [185, 74, 81]]],
      dtype=uint8)
```

RGB-Bilder in Python - II

- Bilder sind 3D-Arrays (Höhe×Breite×Kanäle)
- einzelne Kanäle können einfach angesteuert werden
- Vorsicht bei der Anwendung von Funktionen

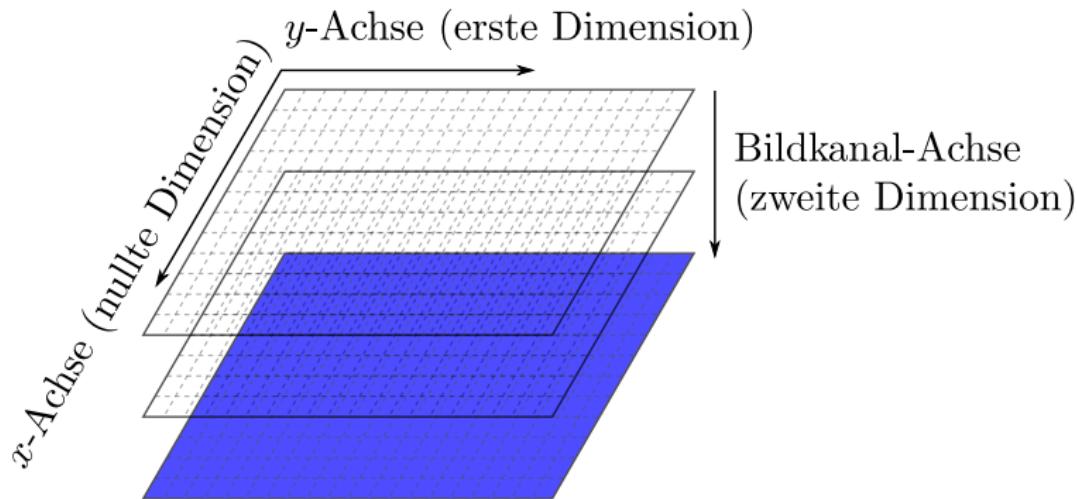
```
>>> img.shape
(512,512,3)
>>> img[:, :, 0] #Rot-Kanal
>>> img[100:200,100:200,0] #Fenster aus Rot-Kanal
>>> img[100:200,100:200, :] #Fenster aus RGB-Bild
>>> plt.imshow(img)
>>> 255-img #Bild invertieren, auch RGB
>>> np.mean(img, axis = (0,1)) #mit axis lassen sich
    die Achsen selektieren
array([ 180.22365952,   99.05121613,  105.41025162])
```

RGB-Bilder in Python - III



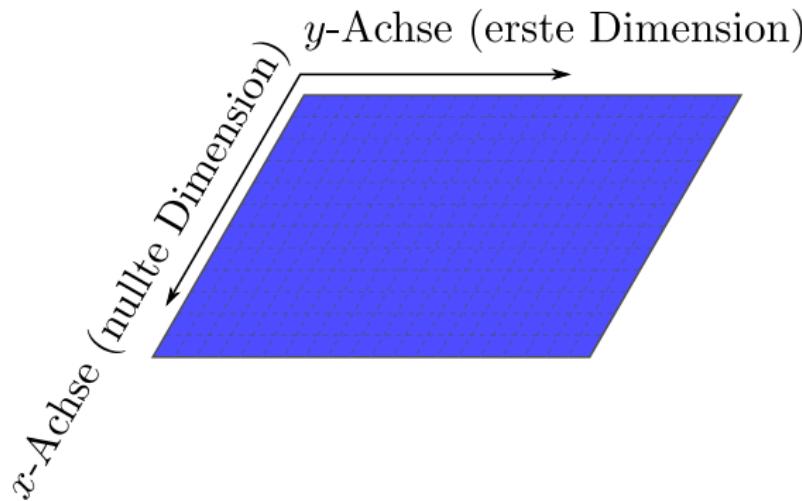
```
>>> ImgRGB
```

Slicing von RGB-Bildern - Kanalauswahl



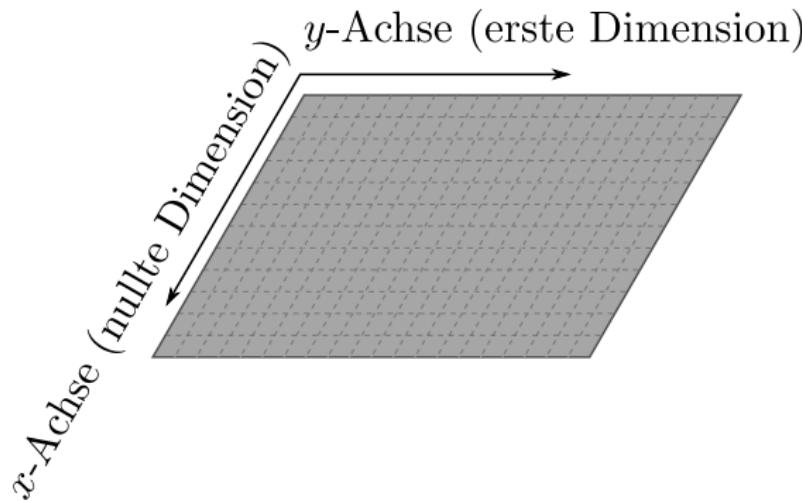
```
>>> #ImgRGB[<von>:<bis>, <von>:<bis>, <von>:<bis>]  
>>> ImgRGB.shape  
(12, 20, 3)  
>>> ImgRGB[:, :, 2] #== ImgRGB[0:12, 0:20, 2]
```

Slicing von RGB-Bildern - Kanalauswahl



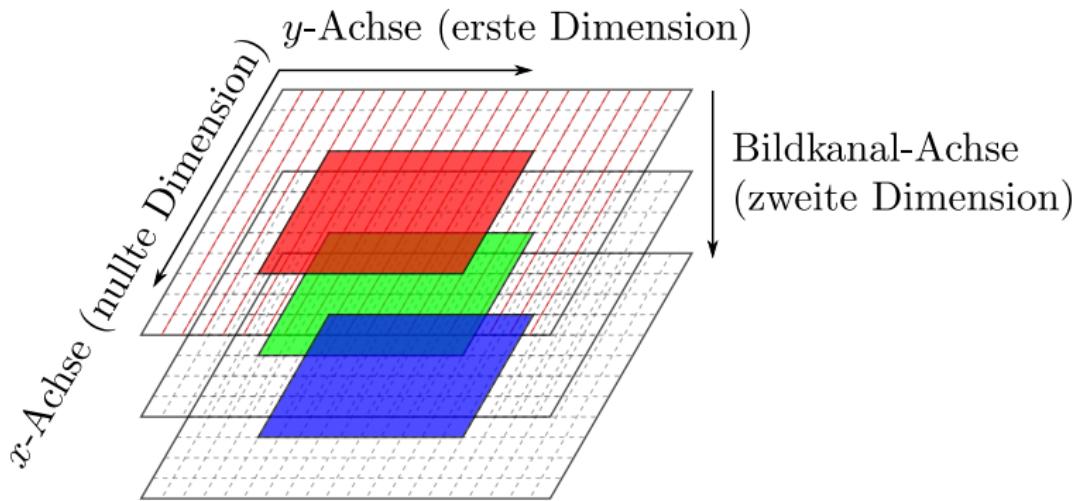
```
>>> #ImgRGB[<von>:<bis>, <von>:<bis>, <von>:<bis>]  
>>> ImgRGB.shape  
(12, 20, 3)  
>>> ImgRGB[:, :, 2] #== ImgRGB[0:12, 0:20, 2]
```

Slicing von RGB-Bildern - Kanalauswahl



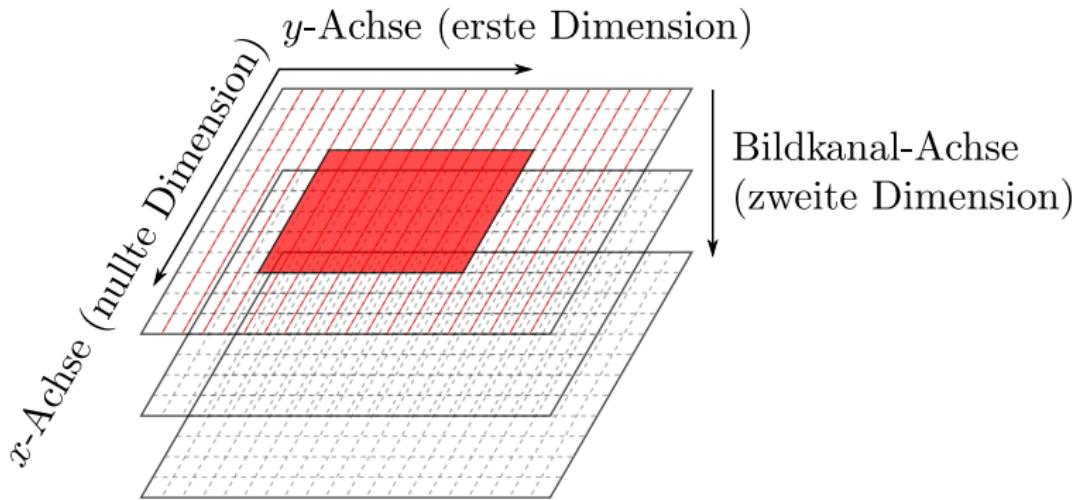
```
>>> #ImgRGB[<von>:<bis>, <von>:<bis>, <von>:<bis>]  
>>> ImgRGB.shape  
(12, 20, 3)  
>>> ImgRGB[:, :, 2] #== ImgRGB[0:12, 0:20, 2]
```

Slicing von RGB-Bildern - Fensterauswahl



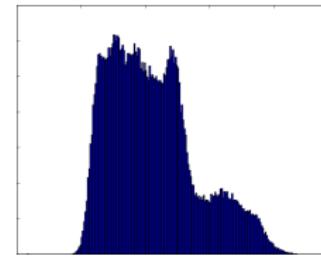
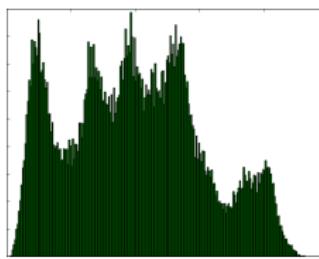
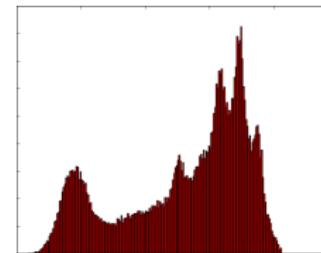
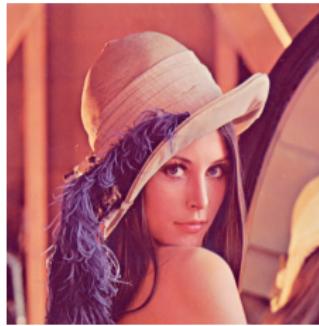
```
>>> #ImgRGB[<von>:<bis>, <von>:<bis>, <von>:<bis>]  
>>> ImgRGB.shape  
(12, 20, 3)  
>>> ImgRGB[3:9, 4:14, :] #== ImgRGB[3:9, 4:14, 0:3]
```

Slicing von RGB-Bildern - Kanal- und Fensterauswahl



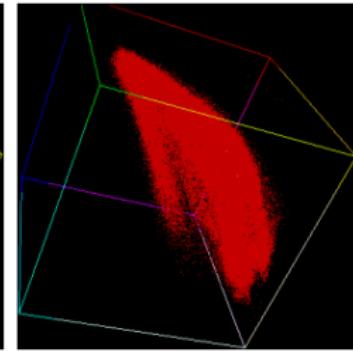
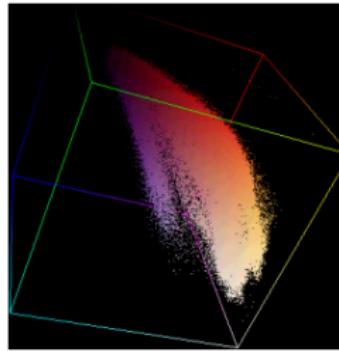
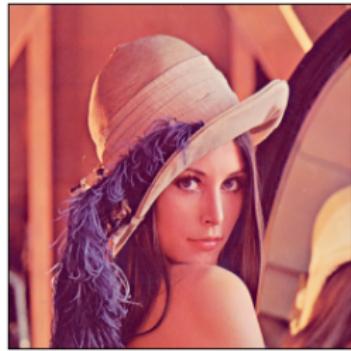
```
>>> #ImgRGB[<von>:<bis>, <von>:<bis>, <von>:<bis>]  
>>> ImgRGB.shape  
(12, 20, 3)  
>>> ImgRGB[3:9, 4:14, 0]
```

Farb-Histogramm - I



Möglichkeit 1: ein Histogramm pro Kanal ($256 \cdot 3$ Behälter).

Farb-Histogramm - II



Möglichkeit 2: ein Histogramm mit drei Dimensionen (256^3 Behälter).

Farb-Histogramm - 1D-Histogramm pro Kanal

- Bild zerlegen in einzelne Kanäle
- auf jedem Kanal die Funktion `histogram` anwenden
- die Parameter sollten in allen drei Fällen die gleichen sein

```
>>> rotKanal = img[:, :, 0]
>>> gruenKanal = img[:, :, 1]
>>> blauKanal = img[:, :, 2]
>>> histR = np.histogram(rotKanal, bins = 256, range
= (0,256))[0]
>>> histG = np.histogram(gruenKanal, bins = 256,
range = (0,256))[0]
>>> histB = np.histogram(blauKanal, bins = 256, range
= (0,256))[0]
```

Farb-Histogramm - 3D-Histogramm

- `histogramdd` kann ein d -dimensionales Histogramm erstellen
- Bild kann nicht direkt übergeben werden
- `histogramdd` erwartet den Input in der Form $p \times k$ mit der Anzahl der Pixel im Bild p und der Anzahl der Kanäle k
- daher muss das Bild reshaped werden
- die Parameter müssen nun jeweils als Listen oder Tupel mit k Elementen übergeben werden

```
>>> img = img.reshape((img.shape[0]*img.shape[1],3))
>>> np.histogramdd(img, bins = [8,8,8], range
    =((0,256),(0,256),(0,256)))[0]
```

Aufgaben

Löst das Aufgabenblatt 3.2!