Computer Vision

Von Pixeln zur Wahrnehmung - Einführung, Anwendungsgebiete & Digitale Bildrepräsentation

Alexandra Posekany

December 2025

Was ist Computer Vision?

Ein interdisziplinäres Feld, das Computern beibringt, visuelle Daten (Bilder/Videos) zu "sehen", zu verarbeiten und zu verstehen.

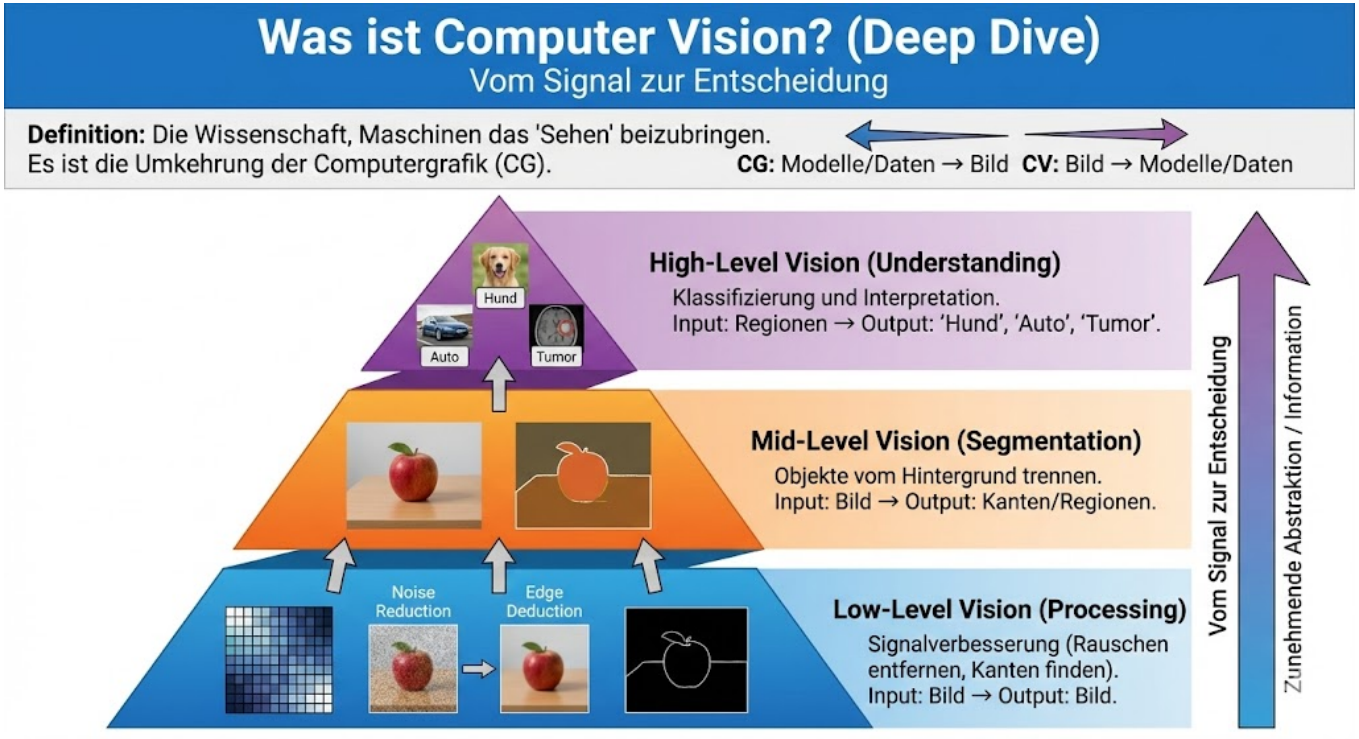
Kernfrage: Wie **extrahieren wir semantische Informationen** (Bedeutung) aus rohen Daten (Pixeln)?

Abgrenzung:

- **Image Processing**: Bild rein $\rightarrow$ Bild raus (z.B. Filter, Helligkeitskorrektur).
- **Computer Vision**: Bild rein $\rightarrow$ Information/Entscheidung raus (z.B. "Da ist ein Fußgänger").

Das Ziel: Nachbildung der menschlichen **visuellen Wahrnehmung**.

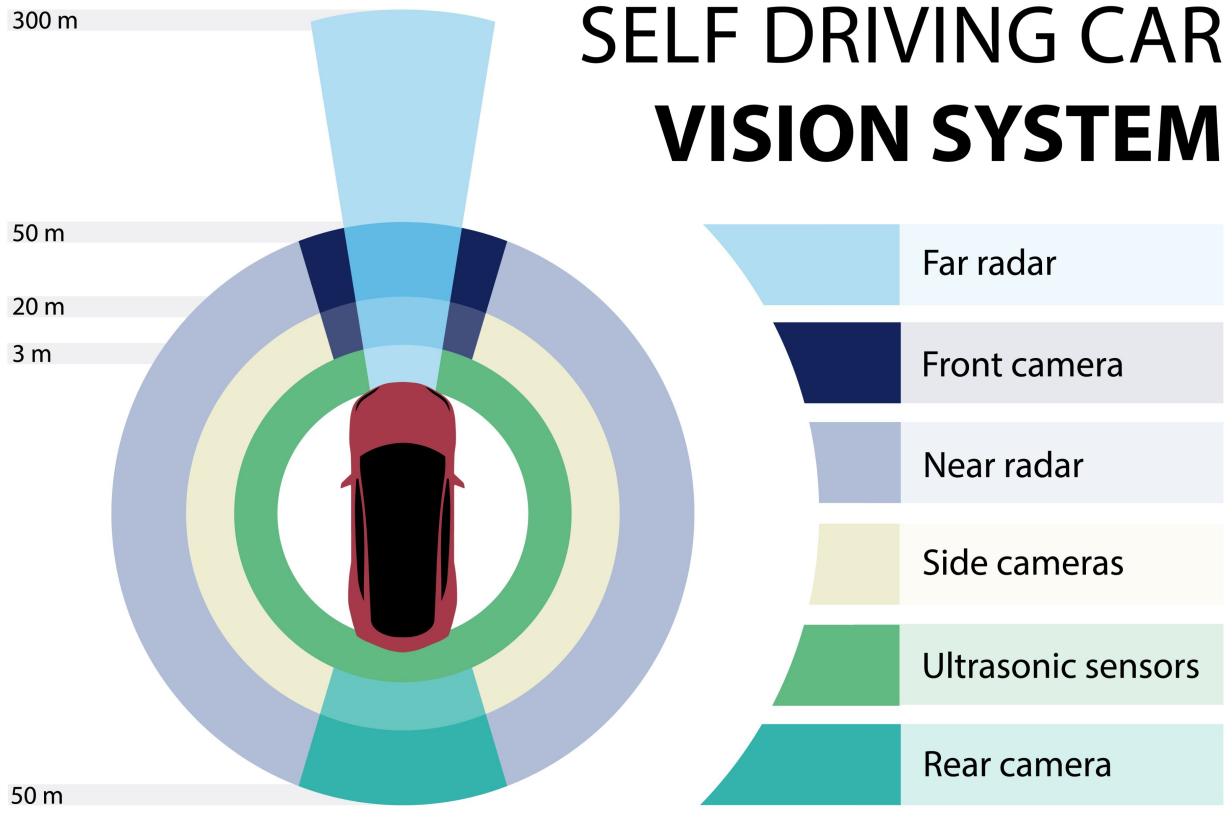
Low vs. High Level Computer Vision



Anwendungsgebiete

- Automotive (Autonomes Fahren):

  Spurhalteassistenten, Verkehrszeichenerkennung, Fußgängererkennung (Tesla Autopilot, Mobileye).



Anwendungsgebiete

- Medizintechnik (Medical Imaging):

  Automatisierte Tumorkennung in MRT/CT-Scans, Zellzählung, KI-gestützte Diagnostik.

Anwendungsgebiete

- Industrie 4.0 & Robotik:

  Qualitätskontrolle (Defekterkennung), Bin Picking (Roboter greift unsortierte Teile), visuelle Navigation.

Anwendungsgebiete

- Augmented Reality (AR) & Entertainment:

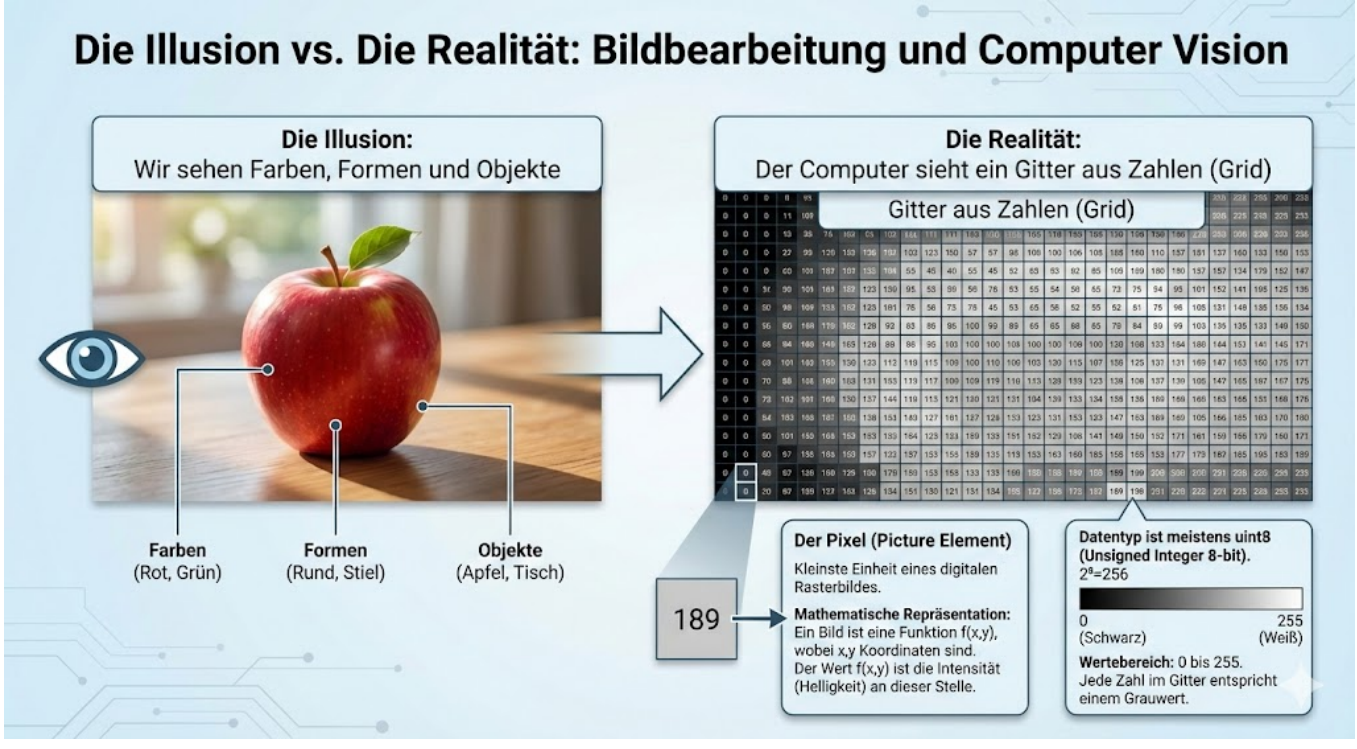
  Gesichtsfilter (Snapchat/TikTok), AR-Navigation (Google Maps), Überlagerung von Montageanleitungen (HoloLens).

Wie “sieht” ein Computer? Das Bild als Datenmatrix

Die Illusion: Wir sehen Farben, Formen und Objekte.

Die Realität: Der Computer sieht ein Gitter aus Zahlen (Grid).

Der Pixel (Picture Element): Kleinste Einheit eines digitalen Rasterbildes.



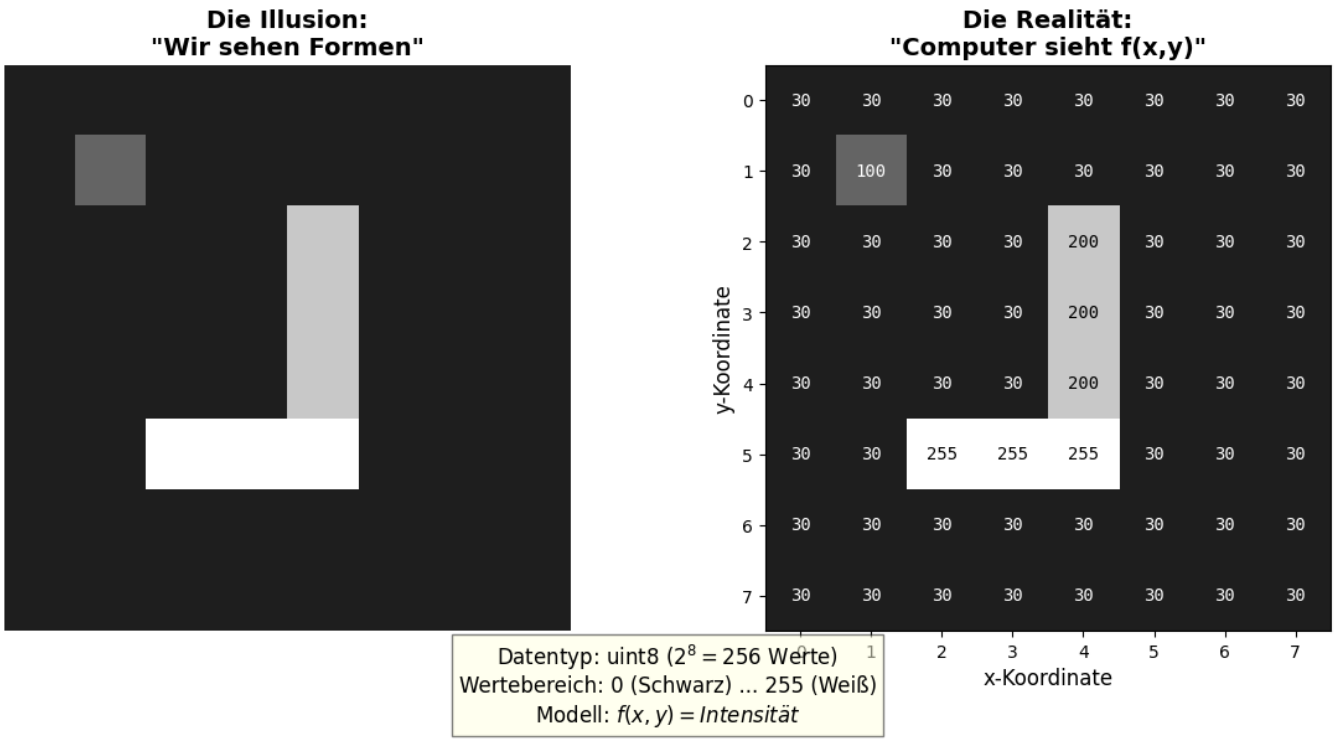
Mathematische Repräsentation

Datentyp ist meistens uint8 (Unsigned Integer 8-bit). $2^8 = 256$ Graustufen

Wertebereich: 0 (Schwarz) bis 255 (Weiß).

Mathematische Repräsentation:

- Ein Bild ist eine Funktion f(x,y), wobei x,y Koordinaten sind.
- Der Wert f(x,y) ist die Intensität (Helligkeit) an dieser Stelle.



Kanäle & Farbräume (RGB)

Graustufenbild: Matrix Dimension: (Höhe x Breite x 1).

Farbbild (RGB): Matrix Dimension: (Höhe x Breite x 3).

Tensor-Dimensionen:

- Ein Bild ist ein 3D-Array (Tensor): (H,W,C).

  H: Height (Zeilen), W: Width (Spalten), C: Channels (Kanäle).

- Drei übereinanderliegende Matrizen (Channels):

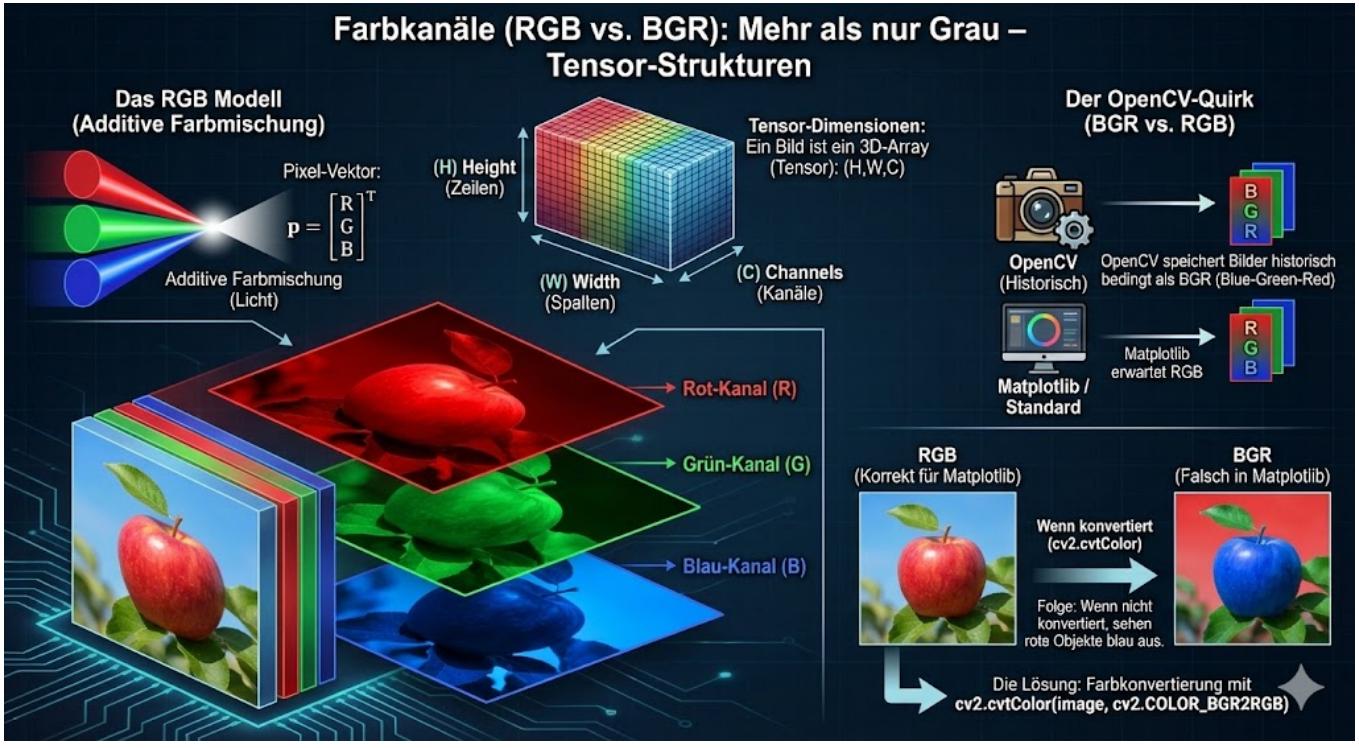
  Red, **G**reen, **B**lue (Additive Farbmischung)

Beispiel Pixel:

- Rot: [255, 0, 0]
- Weiß: [255, 255, 255]
- Schwarz: [0, 0, 0]

Achtung bei OpenCV: OpenCV liest standardmäßig als BGR (Blue-Green-Red), nicht RGB!

RGB


Farbkanäle (RGB vs. BGR): Mehr als nur Grau – Tensor-Strukturen
Das RGB Modell (Additive Farbmischung)
Pixel-Vektor:
$\mathbf{p} = \begin{bmatrix} R \\ G \\ B \end{bmatrix}^T$
Additive Farbmischung (Licht)
(H) Height (Zeilen)
(W) Width (Spalten)
(C) Channels (Kanäle)
Tensor-Dimensionen: Ein Bild ist ein 3D-Array (Tensor): (H,W,C)
Rot-Kanal (R)
Grün-Kanal (G)
Blau-Kanal (B)
Der OpenCV-Quirk (BGR vs. RGB)
OpenCV (Historisch)
B G R
OpenCV speichert Bilder historisch bedingt als BGR (Blue-Green-Red)
Matplotlib / Standard
Matplotlib erwartet RGB
R G B
RGB (Korrekt für Matplotlib)
BGR (Falsch in Matplotlib)
Wenn konvertiert (cv2.cvtColor)
Folge: Wenn nicht konvertiert, sehen rote Objekte blau aus.
Die Lösung: Farbkonvertierung mit cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

Graustufen (Grayscale)

Warum? Reduktion der Datenmenge um Faktor 3.

- Speichereffizienz
- Recheneffizienz

Viele Algorithmen (Kantenerkennung) basieren auf Helligkeitsunterschieden, Farbe ist oft irrelevante Information.

- Näherungsformel:

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

Das menschliche Auge ist empfindlicher für Grün.

Alternative Farbräume (HSV)

Das Problem mit RGB:

- RGB korreliert Farbe und Helligkeit.
- Beispiel: Ein rotes Auto im Schatten hat völlig andere RGB-Werte als in der Sonne (R=100 vs. R=200). Wie soll man programmieren: if pixel == red?

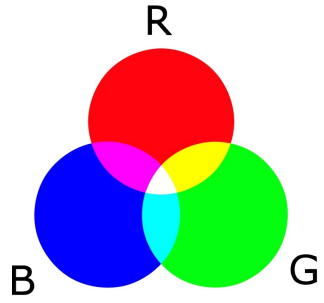
Die Lösung: **HSV** (Hue, Saturation, Value)

- Hue (Farbton): Winkel 0-179 (in OpenCV halbiert, damit es in 8-bit passt). Rot ist bei 0/180.
- Saturation (Sättigung): Wie "rein" ist die Farbe?
- Value (Helligkeit): Wie hell/dunkel ist die Farbe?

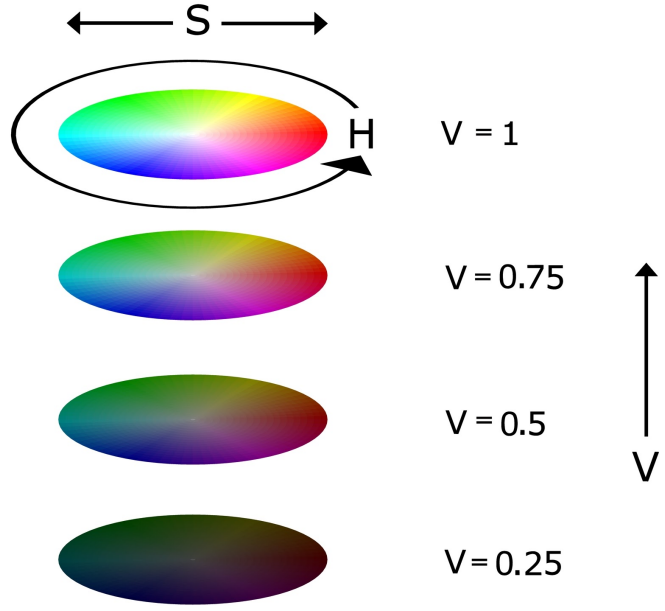
Vorteil: Trennt Farbinformation von der Beleuchtung. Besser für Farberkennung bei wechselndem Licht (z.B. "Erkenne den roten Ball im Schatten").

HSV vs. RGB

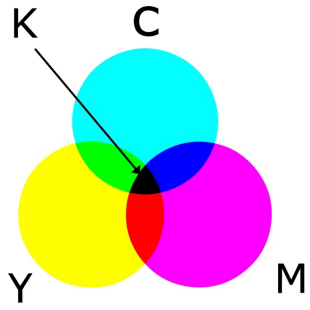
RGB



HSV



CMYK



Histogramm

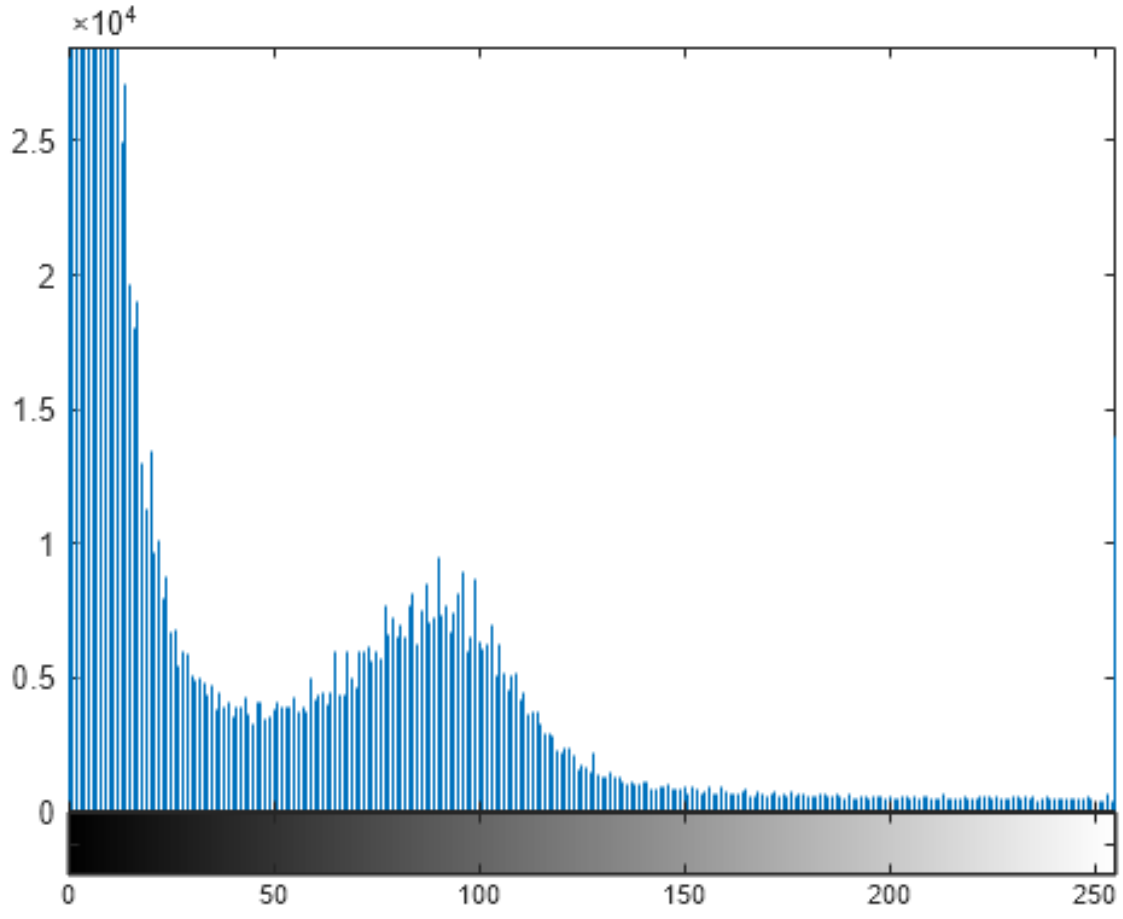
Häufigkeitsverteilung der Pixelwerte im Bild.

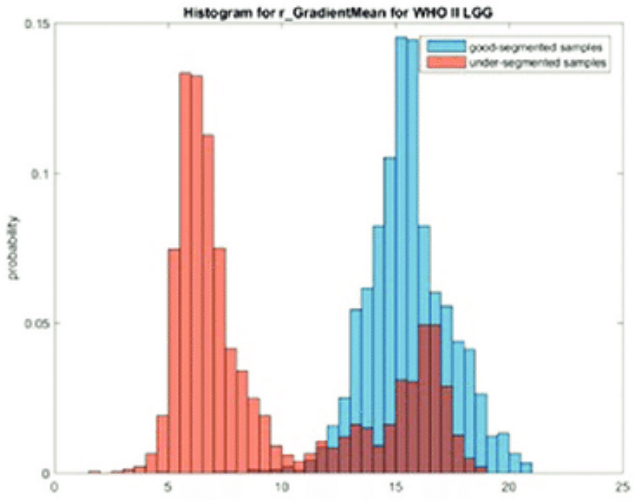
- X-Achse: Pixelwert (0-255).
- Y-Achse: Anzahl der Pixel mit diesem Wert.

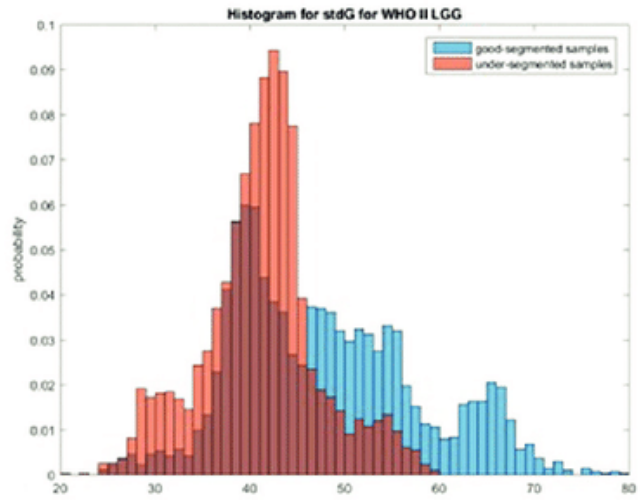
Histogramm lesen lernen als “Fingerabdruck” der Belichtung.

- Schmaler Berg in der Mitte = Kontrastarm (grau in grau).
- Breiter Berg über alles = Guter Kontrast.
- Zwei Berge (bimodal) = Gute Trennbarkeit (z.B. dunkles Objekt auf hellem Grund $\rightarrow$ einfaches Thresholding möglich).

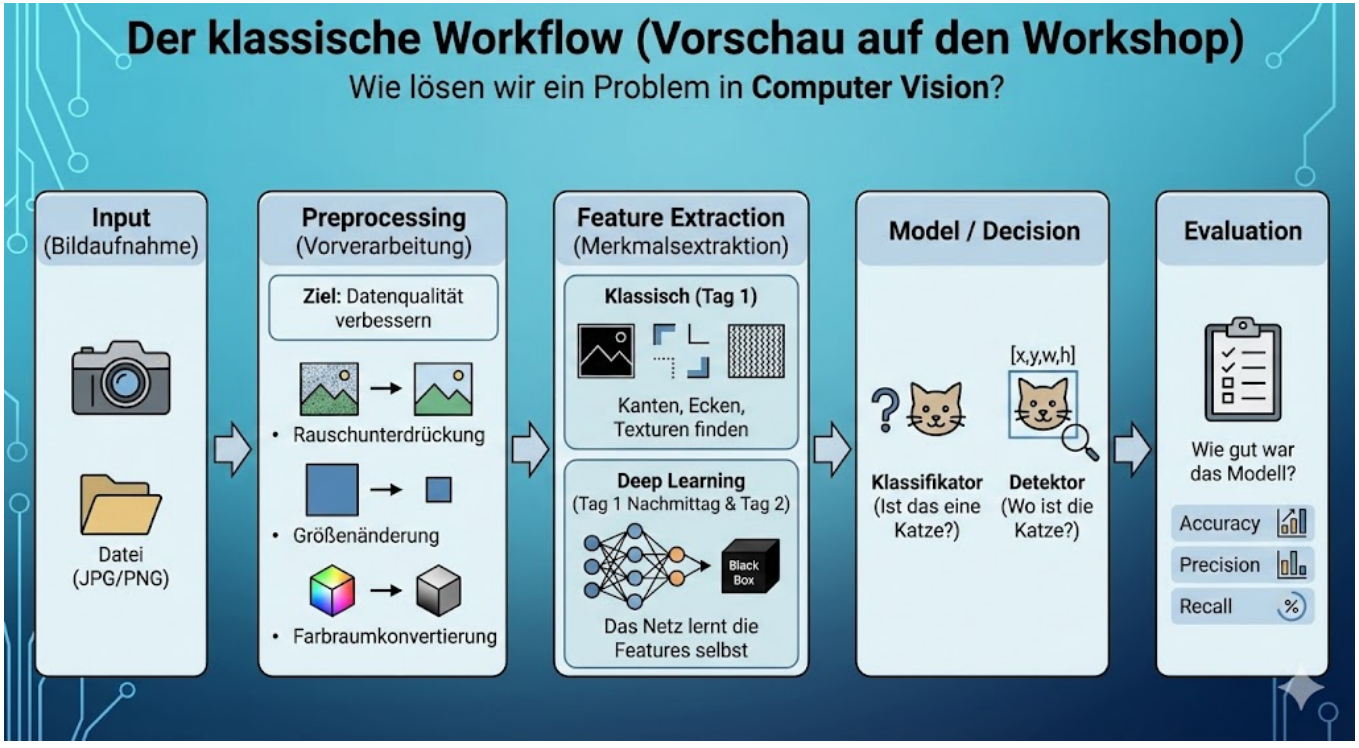
Anwendung: Automatische Belichtungskorrektur, Schwellenwertbestimmung (Thresholding).


×10^4
2.5
2
1.5
1
0.5
0
0
50
100
150
200
250


Histogram for r_GradientMean for WHO II LGG
good-segmented samples
under-segmented samples
probability
0.15
0.1
0.05
0
0
5
10
15
20
25


Histogram for stdG for WHO II LGG
good-segmented samples
under-segmented samples
probability
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
20
30
40
50
60
70
80

Workflow


Der klassische Workflow (Vorschau auf den Workshop)
Wie lösen wir ein Problem in Computer Vision?
Input
(Bildaufnahme)
Datei
(JPG/PNG)
Preprocessing
(Vorverarbeitung)
Ziel: Datenqualität verbessern
• Rauschunterdrückung
• Größenänderung
• Farbraumkonvertierung
Feature Extraction
(Merkmalsextraktion)
Klassisch (Tag 1)
Kanten, Ecken, Texturen finden
Deep Learning
(Tag 1 Nachmittag & Tag 2)
Black Box
Das Netz lernt die Features selbst
Model / Decision
[x,y,w,h]
Klassifikator
(Ist das eine Katze?)
Detektor
(Wo ist die Katze?)
Evaluation
Wie gut war das Modell?
Accuracy
Precision
Recall

Zu den Übungen

Github Repository zu Computer Vision

Beispiel 1 und Strawberry, Gentiana Bilder herunterladen

Ausführen mithilfe von

- Juypter Labs
- Google Colab