Bildverarbeitung OpenCV

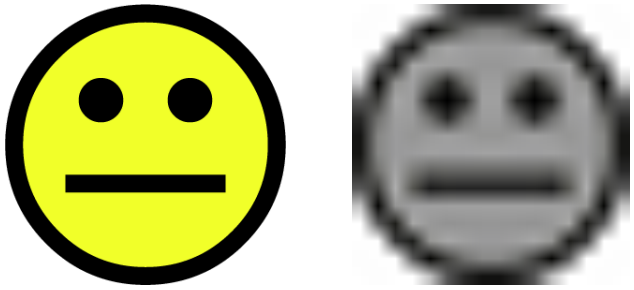
mail@uwe-arzt.de

Inhalte

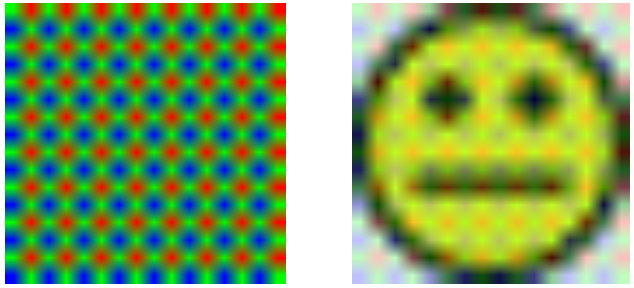
- Wie kommt das Bild in den Computer
- Farbmodelle
- Speichermodelle
- Lokale Operatoren / Randproblem
- Globale Operatoren
- Praktische Anwendungen
- Was gibt es noch in OpenCV / warum OpenCV
- Andere Bibliotheken
- Links, ...

Sensoren

- Graustufen



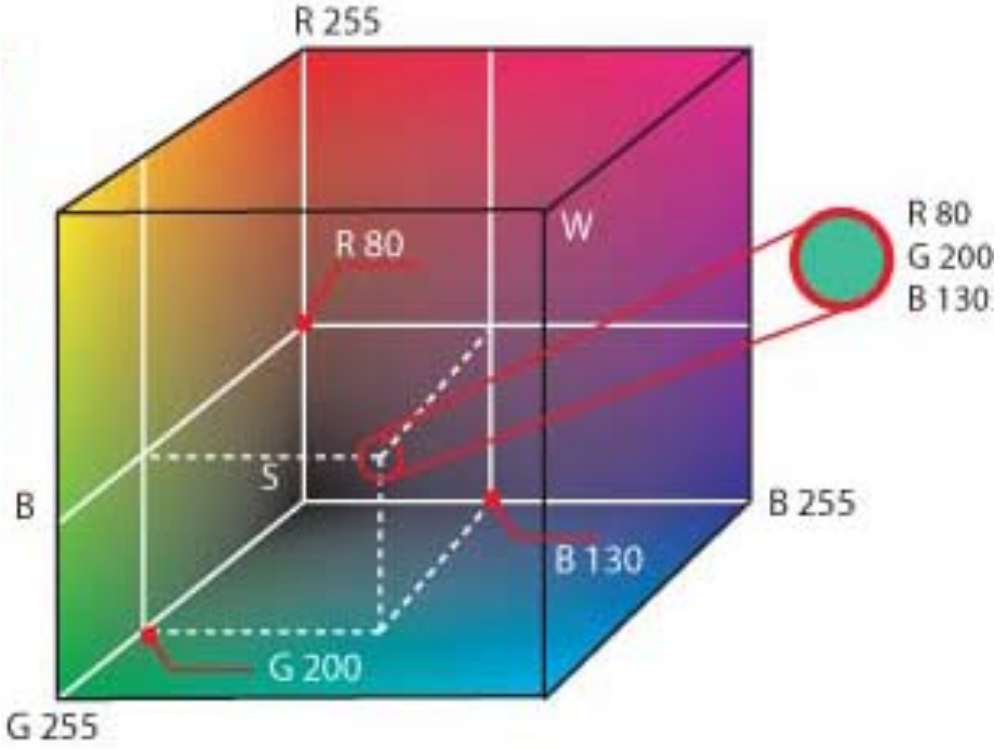
- Bayer Pattern



Rolling / Global Shutter

Farbmodelle

- RGB
  Es werden die Rot/Grün/Blau Anteile der einzelnen Pixel gespeichert. Gängige Farbtiefen sind 8 Bit (wird z.b. auch für Farbangaben im Web verwendet), 10 Bit (Können nicht alle Rechner darstellen), und 565 Bit für die einzelnen Farbanteile.



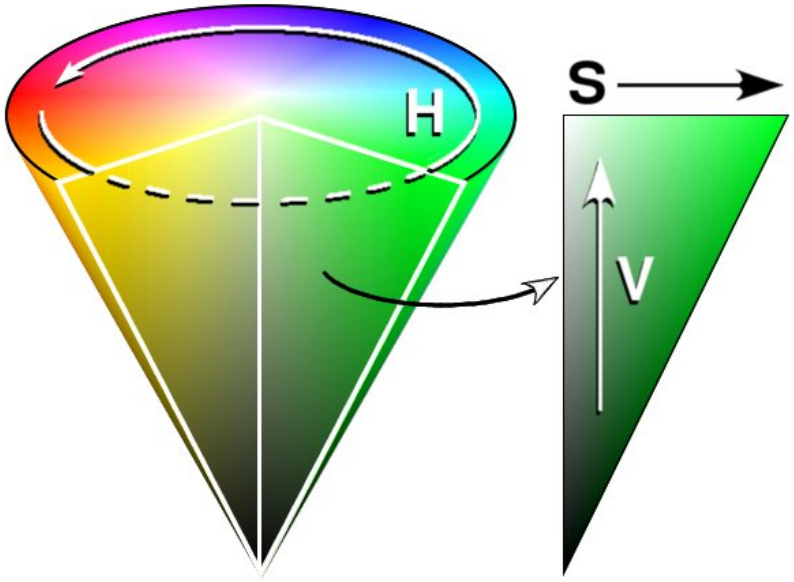
Farbmodelle

- RGBA
  Hier wird zusätzlich zu den 3 Farbanteilen ein Alpha Channel gespeichert. Dieser bestimmt die Transparenz an den entsprechenden Pixeln des Bildes. Oft in 8 Bit Auflösung, kann aber auch in einem Bit (als Maske) gespeichert/verwendet werden.
- Graustufen
  In einem RGB Bild sind alle Pixel grau, bei denen R=B=G ist. Die Speicherung kann dann entsprechend komprimiert erfolgen, d.h nur mit einem Wert pro Pixel.

Farbmodelle

Ein grundsätzliches Problem von RGB ist die „Inkompatibilität“ mit der menschlichen Wahrnehmungspsychologie und Beschreibung.

- HSI / HSV
  - Hue (Farbe)
    -> Winkel im Farbkreis
  - Saturation (Sättigung)
    -> von neutralgrau zu reiner Farbe
  - Intensity (Hellwert) / Value (Dunkelstufe)
- Umrechnungsformeln siehe z.B. Wikipedia

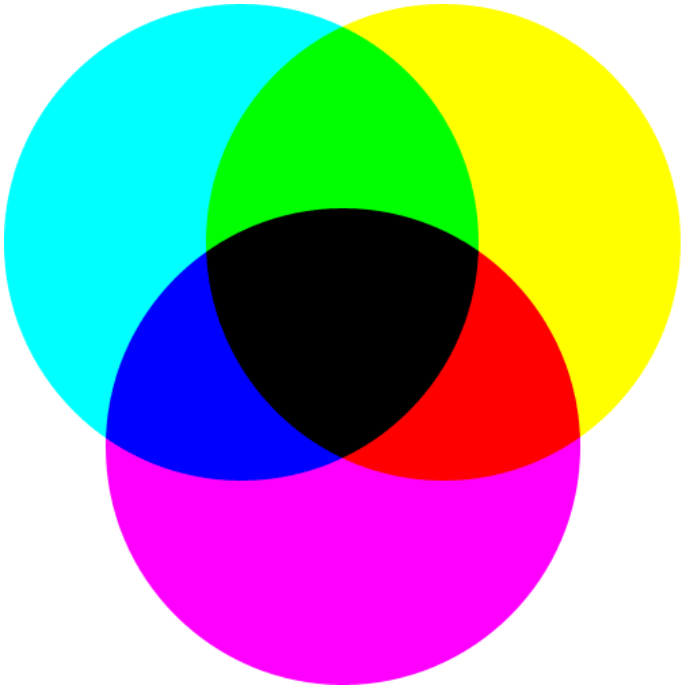


Farbmodelle

- YUV
  - Besteht aus Luminanz (Lichtstärke) und Chrominanz (Farbwert), der wiederum aus zwei Unterkomponenten besteht (U und V)
  - Hat seinen Ursprung in der Umstellung auf Farbfernsehen als zusätzlich zum Schwarzweissanteil ein Farbanteil übertragen werden sollte.
  - Wird oft im Farbanteil mit verminderter Auflösung übertragen, z.B. YUV 4.2.2

Weitere Farbmodelle

- CMYK
  Subtraktive Farbmischung. Wird z.B. in Druckern verwendet

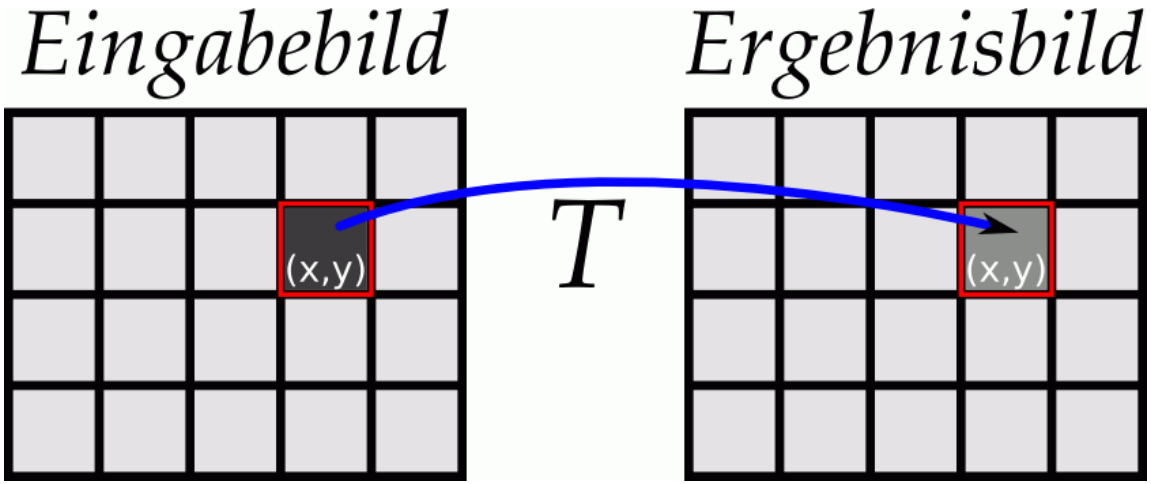


Speichermodelle

- RGB RGB ...
  Die Pixel werden nacheinander im Speicher abgelegt. Das ist die „normale“ Speicherform, da die Pixel (Subpixel) in dieser Reihenfolge aus dem Sensor kommen und in dieser Reihenfolge angezeigt werden.
- RR ... GG ... BB ... / HH ... SS ... VV ...
  Macht Sinn, wenn Operationen nur auf bestimmten Kanäle operieren, da die Daten dann sehr viele lokaler gespeichert sind.
- BGR565 BGR565 ...
  Kompaktere Speicherung, aber unter umständen sehr viele Bitmask Operationen notwendig.

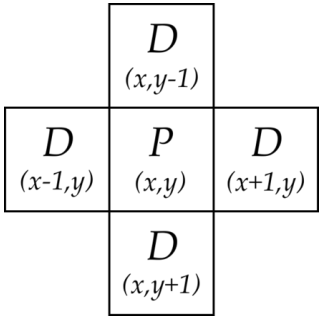
Punktoperator

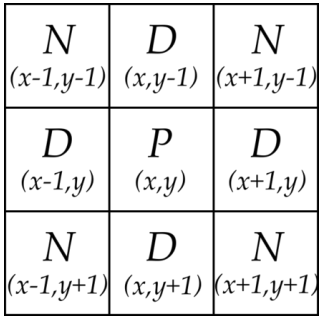
- Es wird genau ein Punkt im Eingangs und Ausgangsbild angepackt
- Optimal parallelisierbar auch bei Inplace Operationen
- Beispiel: Dummes aufhellen (wir addieren 50 zu jedem R, G und B)



Lokale Operatoren

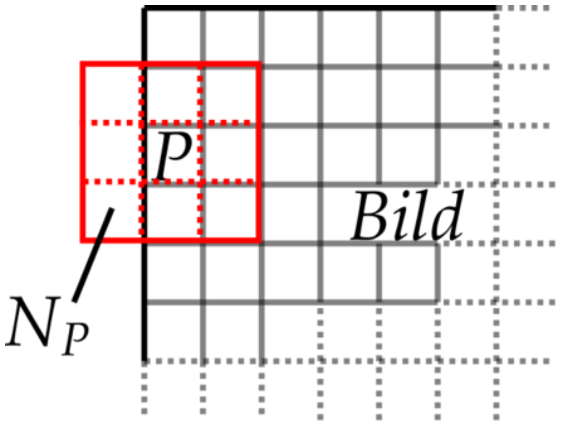
- Es werden mehrere Punkte im Eingabebild mit Hilfe einer Matrix zu einem Punkt im Ergebnisbild
- Als Beispiel hier die Vierer Nachbarschaft, bei die vier angrenzenden Punkte in Die Berechnung eingezogen werden und die Achter Nachbarschaft.
- Gut parallelisierbar (Inplace Operation?)





Randproblem

- Bei all diesen Operatoren tritt das Randproblem auf, sobald man einen Bildrand erreicht, bei dem nicht mehr alle „Nachbarn“ gültig bzw. im Bild sind.
  - Pixel nicht betrachten -> Ergebnisbild wird kleiner
  - Verkleinern der Maske
  - Pixel „erfinden“ -> meist gleicher Wert wir Randpixel, Evtl. periodische Fortsetzung möglich



Beispiel: Sobel (Kantenerkennung)

- Sobel Kantenerkennung gibt es für Kanten in X und in Y Richtung
- Nur ein Beispiel, alleine Wikipedia listet > 10 Operatoren für Kantenerkennung
- Ausführung auf einem Kanal (Graustufenimage)
- Normalerweise immer mit weiteren Operationen kombiniert

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Globale Operatoren

- Jeder Pixel hat Auswirkungen auf das Ergebnis
- Beispiel Histogram
- Ein Histogram macht auf einem Farbbild meisten nur auf den einzelnen Kanälen Sinn
- Basis für viele andere Operationen (z.B. Tonwertspreizung)

Praktische Anwendung

- Blueboxing Demo
- Video + VideoSobel, VideoDiff (Einfachste Form der Bewegungsdetektion)

Historie von OpenCV

- Wurde ursprünglich bei Intel entwickelt
- Heute gepflegt und weiterentwickelt vor allem von WillowGarage

- Historische C, heute
  - C++
  - Python Bindings
  - Java Binding
  - Linux, Mac, Windows
  - Android, iOS

Anwendungsbereiche OpenCV

- Gesichtserkennung
- Objekterkennung
- Roboter
- 3D Sehen
- Tracking
- Photogrametrie
- Maschinelles Lernen
  - Neuronale Netze
  - Bayes
  - …

Weitere Bibliotheken

- Boost Gil -> Nur Container, keine Algorithmen
- Integrating Vision Toolkit
- Halcon -> Kommerziell
- ImageJ -> Java

Und unzählige weitere, siehe Google Wikipedia

Vorteil OpenCV: Es sind schon zahllose Algorithmen implementiert, so das es einfach ist zu vergleichen

Weitere Infos

- Folien und Quelltext
  https://github.com/uwearzt/Bildverarbeitung-OpenCV.git
- OpenCV
  http://opencv.org
- OpenCV Dokumentation
  http://docs.opencv.org
- Wikipedia