

Übungsblatt 3

Julius Auer

Ich bin so frei, die Aufgabenstellung ein wenig zu interpretieren:

- da CAMSHIFT für ein Einzelbild wenig Sinn macht, werde ich eine Folge von Frames aus dem avi untersuchen (die Änderung der Fenstergröße bei CAMSHIFT ist sonst nicht gut zu sehen)
- einige Optimierungen aus dem Paper sind speziell für "Fleisch" vorgenommen worden und sollten für das Auto neu untersucht werden (Fehlerfunktion für Histogramm und Funktion zur Anpassung der Fenstergröße)

Ich löse die Aufgaben somit in Hinblick auf eine gute CAMSHIFT-Implementierung, die für das avi brauchbare Ergebnisse liefert - Fleisch ist dort nunmal eher wenig zu sehen ...

Aufgabe 1 (Farbräume):

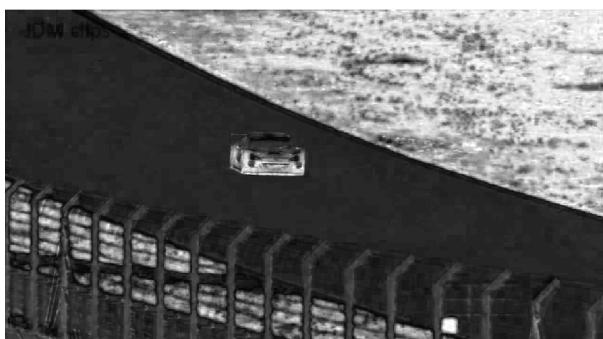
- 1.1) - Lade das angehängte Bild "racecar.png"
 - konvertiere das Bild in den HSV-Farbraum
 - stelle das Histogramm über dem Hue-Kanal für das gesamte Bild dar



(a) Original



(b) Hue



(c) Saturation



(d) Value

Abbildung 1: Komponenten des HSV-Bildes

Abbildung 1 zeigt deutlich, warum es **nicht** sinnvoll ist, das Histogramm naiv aus den *Hue*-Werten zu konstruieren: die Werte des Autos und der Straße sind zu ähnlich. Wo Bradski eine Filterfunktion vorschlägt, die vor Allem Pixel mit niedriger *Value* filtert, erscheint hier das Filtern nach *Saturation* sinnvoll: in dieser Dimension unterscheiden sich Auto und Straße deutlich.

Der vorliegenden Implementierung kann zur Berechnung des *Hue*-Bildes eine Fehlerfunktion übergeben werden. Für den vorliegenden Fall wurde eine Fehlerfunktion gewählt die alle Pixel (x, y) mit $\text{sat}((x, y)) \leq 0.2$ filtert. Hieraus ergeben sich das in Abbildung 2 gezeigte *Hue*-Bild mit zugehörigem Histogramm.

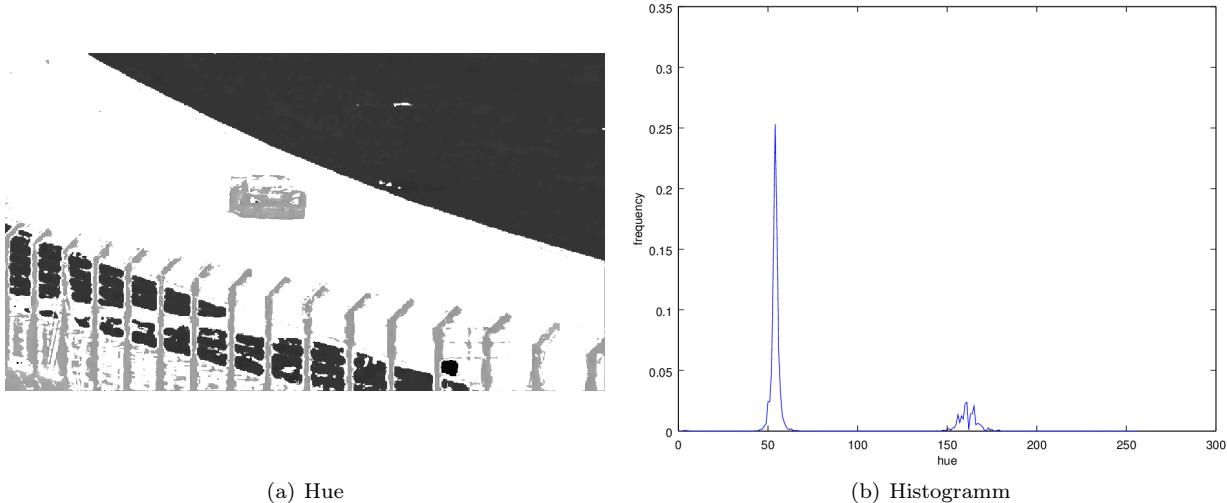


Abbildung 2: Mit Fehlerfunktion gefiltertes Bild

1.2) und für den Ausschnitt $(x, y) = (460, 260)$ bis $(660, 360)$

Siehe Abbildung 3. Der Ausschnitt ist einen Tick kleiner als angegeben.

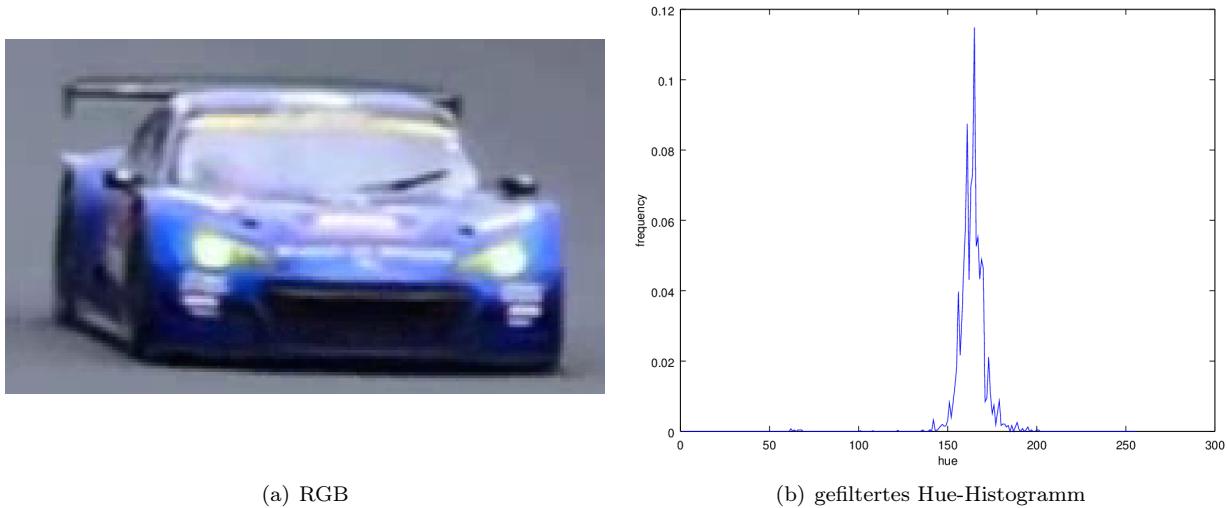


Abbildung 3: Ausschnitt

Aufgabe 2 (CAMSHIFT):

- Lies das paper: http://opencv.jp/opencv-1.0.0_org/docs/papers/camshift.pdf
- implementiere eine Methode, die Dir - gegeben ein *Hue*-Histogramm - die Objekt-Wahrscheinlichkeitsverteilung für ein neues Bild berechnet (1)
- wähle aus dem angehängten Video "racecar.avi" ein Einzelbild (2)
- wende (1) für das Objekthistogramm aus Abgabe 1.2 auf (2) an und stelle das Ergebniss dar

Aus dem Histogramm des gefilterten Hue-Bildes des Ausschnitts wird die Wahrscheinlichkeits-LUT erzeugt. Lookupen der Hue-Werte des Originalbildes (gefiltert) in dieser Tabelle liefert das Wahrscheinlichkeitsverteilungsbild in Abbildung 4.

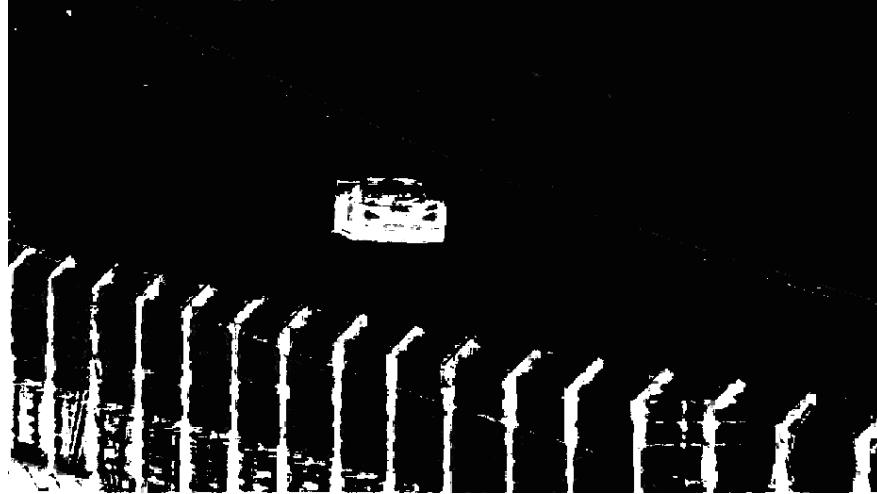


Abbildung 4: Wahrscheinlichkeitsverteilung

Man kann hier bereits erahnen, dass die Differenzierung zwischen den "Streben" vorne im Bild und dem Auto nicht gut ist. Im Rahmen eines Übungszettels erscheint das hinnehmbar - grundsätzlich wäre aber eine ausgefeilte Fehlerfunktion an dieser Stelle wünschenswert.

Mit der zuvor produzierten LUT können nun Einzelbilder aus dem avi analysiert werden: hierfür wird zunächst eine beliebige Fenstergröße angenommen, MEANSIFT durchgeführt (mit Wahrscheinlichkeitsverteilung, LUT, und Fensterparametern), eine neue Fenstergröße berechnet und diese für das nächste Einzelbild gespeichert.

Das Berechnen einer neuen Fenstergröße ist hierbei nicht trivial und wird von Bradski für den konkreten "Fleisch"-Fall beschrieben. Für das Auto gilt dieser nicht da:

- für das Auto kein sinnvoller Roll-Winkel gemessen werden kann
- sich die Aspect-Ratio (die Bradski als konstant annimmt) deutlich ändern kann

In der vorliegenden Implementierung kann an CAMSHIFT eine Funktion zur Berechnung der neuen Fenstergröße übergeben werden, für die als Parameter die Bild-Momente 0.–2. Grades $m_{00}, m_{10}, m_{01}, m_{02}, m_{20}$ zur Verfügung stehen. Experimentell hat sich im vorliegenden Fall eine lineare Funktion von $\sqrt{m_{00}}$ als hinreichend gut erwiesen. Zur Berechnung der neuen Aspect-Ratio ar wird stets

$$ar = m_{20} \cdot \left(\frac{m_{00}}{m_{10}} \right)^2 / m_{02} \cdot \left(\frac{m_{00}}{m_{01}} \right)^2$$

berechnet. Die Abbildung 5-6 zeigen das Resultat bei Anwendung des Algorithmus auf einige aufeinanderfolgende Frames (bei einer Framerate von 1).



(a) Frame 1



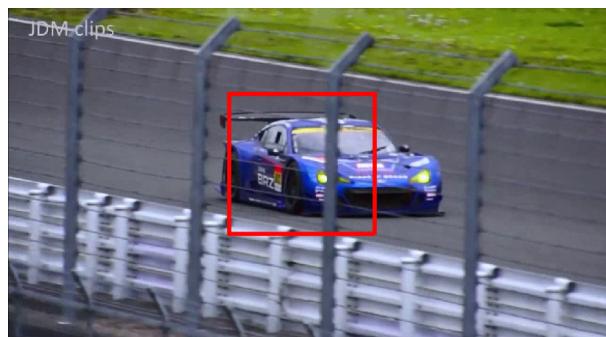
(b) Frame 2



(c) Frame 3



(d) Frame 4

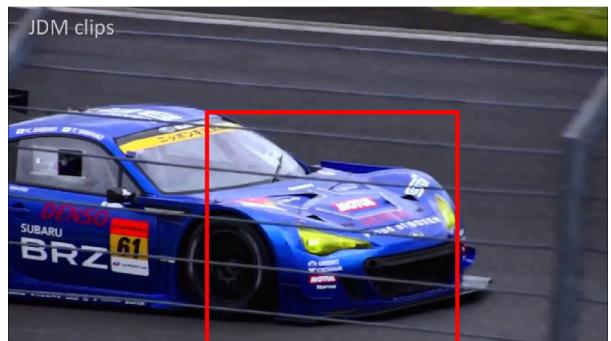


(e) Frame 5



(f) Frame 6

Abbildung 5: Resultat (1)



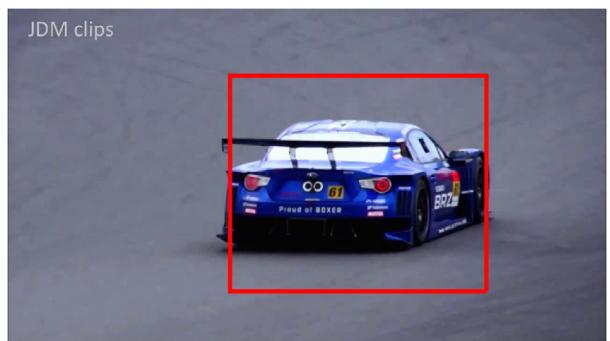
(a) Frame 7



(b) Frame 8



(c) Frame 9



(d) Frame 10



(e) Frame 11

Abbildung 6: Resultat (2)