

Übungsblatt 4

Julius Auer

Aufgabe 1 (CAMSHIFT (Teil 2)):

- *implementiere die Berechnung des Fensterzentrums, -größe und Objektorientierung wie im Paper*
- *tracke das Auto vom ersten zum letzten Frame*

Nun, ich hatte die Aufgabe ja aus Versehen letzte Woche schon gelöst. Um nicht ganz untätig zu sein, habe ich diese Woche meine Lösung nach C++ portiert, um sie Echtzeit-fähig zu machen (der Octave-Prototyp von letzter Woche war zu langsam, um das ganze Video zu verarbeiten).

Das Ergebnis ist sehr gut. Für Basis-Operationen wie Farbraum-Konvertierungen und IO benutze ich OpenCV, CAMshift selbst ist von mir implementiert (obwohl OpenCV das auch angeboten hätte). Ich habe den Code und das Ergebnis als Video hochgeladen. Einzelbilder zeigt Abbildung 1.

Was die eigentliche Aufgabenstellung betrifft, wiederhole ich nur kurz das Wesentliche vom letzten Zettel:

- Da hier kein Fleisch sondern ein Auto getrackt wird, muss an einigen Stellen vom Vorgehen im Paper abgewichen werden:
- Für das Hue-Bild wird ein Fehler anhand der Satuation-Werte berechnet (bei Fleisch: Value)
- Die Aspect-Ratio für das neue Fenster wird aus M_{20}, M_{02} berechnet (bei Fleisch: fest)
- Der Roll des Kopfes wird nicht berechnet (es gibt keinen Kopf)
- Details zu diesen Anpassungen (mit Bildern) finden sich auf meinem letzten Übungszettel



Abbildung 1: Resultat (Auszug)

Aufgabe 2 (Hu-Momente):

- lies das paper:
<http://www.sci.utah.edu/~gerig/CS7960-S2010/handouts/Hu.pdf>
- implementiere die sieben Momente von Hu
- nimm an, wir suchen ein Symbol ('needle.png') in einem Bild voller Symbole ('haystack.png')
- berechne die sieben Momente für das gesuchte Symbol
- suche die bounding boxes der Symbole im Haystack (wenn ihr mögt auch gerne über Drittsoftware)
- bestimme für jedes Symbol im Haystack die Ähnlichkeit (oder den Abstand) seiner sieben Momente zu denen des gesuchten Symbols

Dazu habe ich nur wenig zu sagen: Der Code ist C++ (wieder mit ein bisschen OpenCV) wobei ich - aus purer Lust am Coden - **nicht** von OpenCV die Bildmomente habe berechnen lassen, sondern alle M, μ, I selbst ausrechne. Das ist der schrecklichste, unleserlichste Code den ich seit langem geschrieben habe :D

Ich lasse von OpenCV die Bounding-Boxes berechnen (Abbildung 2), berechne dann für Needle und alle Boxen jeweils alle Hu-Momente und betrachte den Quadratischen Fehler. Die Box mit dem geringsten Fehler beinhaltet erwartungsgemäß die Nadel (Abbildung 3).

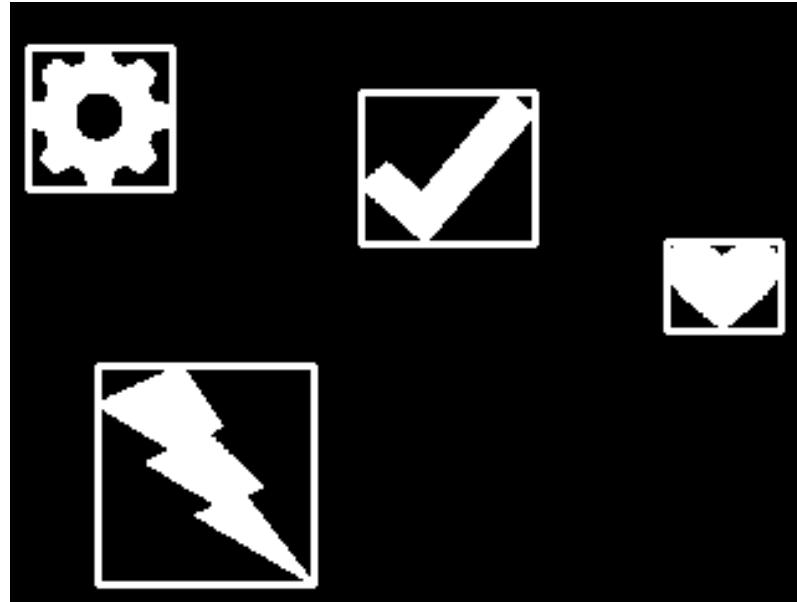


Abbildung 2: Alle Bounding-Boxes (von OpenCV berechnet)

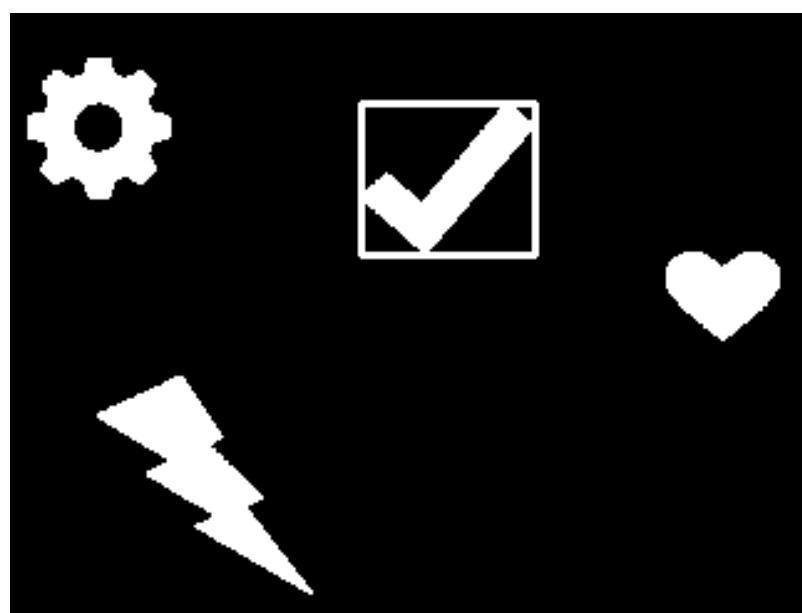


Abbildung 3: Die Box mit dem niedrigsten Fehler