Ludwig-Maximilians-Universität München
LFE Medieninformatik, Mobile Interaction Group
Prof. Dr. Michael Rohs
Dipl.-Inform. Sven Kratz

Sommersemester 2012
Übungsblatt 3
23.5.2012

Übung Computergrafik 2

Abgabetermin:
Die Lösung zu diesem Übungsblatt ist bis zum **30.05.2012** abzugeben.

Form der Abgabe:

- Die Übungen können in Gruppen von 2–3 Studierenden bearbeitet werden.
- Die Abgaben bestehen aus den Python-Quelltexten samt aller Bilder, die für die Bearbeitung der Übung verwendet wurden. Idealerweise wird das gesamte PyDev-Projektverzeichnis für die jeweilige Übung über Eclipse als .zip Archiv exportiert. Wichtig: Zur Vermeidung von Namenskonflikten in Eclipse, euren Nachnamen bitte als Präfix vor die Namen der die PyDev-Projekte setzen, z.B.: „müller-übung-x“.
- Teilaufgaben, deren Python-Dateien wegen Syntaxfehlern nicht ausführbar sind, werden nicht weiter korrigiert.
- Textaufgaben sollten in Form einer .doc, .odt oder (bevorzugt) als PDF-Datei abgegeben werden, und sollten (falls erforderlich) die benötigten Ausgabebilder der Aufgaben enthalten.
- Alle Übungsabgaben erfolgen über UniWorX[1].

Inhalt:

Diese Übung befasst sich mit der Konvertierung in den HSV-Farbraum und Weißabgleich. Es wird Back Projection as Technik der Farbwerterkennung mittels HS-Histogrammen vorgestellt. Als Alternative zur Back Projection wird schließlich die euklidische Farbdistanz zum Farbvergleich behandelt.

[1] https://uniworx.ifi.lmu.de

Aufgabe 1 HSV-Farbraum und Weißabgleich (⋆⋆)

a) **HSV-Konvertierung** Implementieren Sie eine Python-Funktion, die eine Konvertierung eines RGB-Bildes in den HSV-Farbraum vornimmt. Verwenden sie dazu die Lösung von Foley/van Dam, zu finden auf Folie 36 der Vorlesung vom 22.05.2012.

b) **Weißabgleich** Bei Digitalaufnahmen lässt sich über Bildverarbeitungssoftware der Weißabgleich zu einem gewissen Grad per Software korrigieren. Dazu skaliert man die relativen Luminanzen der Kanäle Rot, Grün und Blau. Die Skalierung wird über eine Multiplikation mit der folgenden Diagonalmatrix realisiert:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 255/R'_w & 0 & 0 \\ 0 & 255/G'_w & 0 \\ 0 & 0 & 255/B'_w \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

Das Problem ist nun die Parameter R'_w, G'_w, G'_w für den optimalen Weißpunkt zu wählen. Den Weißpunkt kann man entweder per Manuell in einer Bildverarbeitungssoftware auswählen oder auch automatisch annähern. Im Folgenden wählen Sie einfach selbst einen geeigneten Weißpunkt aus:

(i) Bestimmen Sie ein Pixel p_{white} aus dem Bild, das als Weißpunkt dient.

(ii) R'_w, G'_w, B'_w sind dann genau die r, g, b Werte von p_{white}.

(a)

(b)

(c)

(d)

(a) - (d)
CC-BY-SA
Flickr-User
cerfman17, arturbo, truester, arellis49

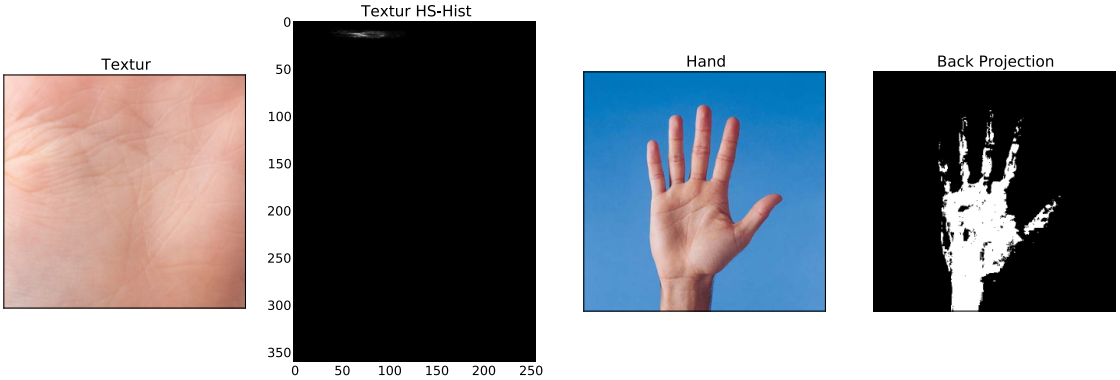
Implementieren Sie den oben beschriebenen Algorithmus zur Anpassung des Weißabgleichs in Python, und wenden Sie diesen auf die Bilder (a)-(d), die auch auf der Webseite zum Download bereitstehen, an. Beurteilen Sie die Ergebnisse.

Aufgabe 2 HS-Histogramme und Back Projection (⋆⋆⋆)

a) **HS-Histogramm** Ein HS-Histogramm ist ein zweidimensionales Histogramm über die hue (h) und saturation (s) Werte eines Bildes. D.h. ein Histogrammeintrag $hs\_hist(h,s)$ gibt genau die Anzahl der Pixel im Bild zurück, die die Werte h, s im HSV-Farbraum haben.

Implementieren Sie eine Python-Funktion, die ein Bild in HSV-Darstellung einliest und ein HS-Histogramm ausgibt.

b) **Back-Projection** Durch Back-Projection lässt sich anhand von Histogrammen festhalten, wie gut die Pixel eines gegebenen Bildes zu der Verteilung der Pixel eines Modellbildes passen. Z.B. lassen sich bestimmte Texturen, wie Hauttöne, in einem Bild finden, wenn man die HS-Verteilung der Hauttöne anhand eines Beispielbildes kennt:



Zur Berechnung der Back Projection verwenden wir die Ratio-Histogram-Methode von Swain und Ballard[2]. Dafür müssen folgende Schritte durchgeführt werden.

(i) Gegeben: Modellbild M und Eingabebild I, jeweils im HSV-Format

(ii) Berechnung des HS-Histogramms für das Modellbild M_i und das zu untersuchende Eingabebild I_i

(iii) Berechnung das Verhältnishistogramms $R_i = min\left(\frac{M_i}{I_i}, 1\right)$

(iv) Ausgabe der Back Projection P für $\forall$ Pixel x, y aus I :

$$\begin{aligned} h, s &= I(x,y) \\ P(x,y) &= R_i(h,s) \end{aligned}$$

Die Idee hinter diesem Algorithmus ist folgende: Der Ausdruck M_i/I_i wird groß, falls M_i groß und I_i klein ist, also wenn die Farbe i häufig im Modellbild, aber selten im Eingabebild vorkommt. Dann ist diese Farbe besonders charakteristisch für das Modellbild und kann gut für die Suche verwendet werden.

Implementieren Sie den Back Projection Algorithmus nach Swain und Ballard in Python. Wählen Sie eine markante Beispiel-Textur und berechnen Sie die Back Projection auf mindestens drei Eingabebildern (z.B. Suche nach Gesichtern in einem Gruppenbild).

Aufgabe 3 Euklidische (Farb-)Distanz (⋆⋆)

Eine Alternative zur Back-Projection um Pixel mit gewissen Farbeigenschaften in einem Eingabebild zu finden ist die euklidische Distanz für die Farbwerte zu verwenden.

Berechnen Sie dazu aus einem Modellbild den Durchschnittsfarbwert sowie die maximale Distanz D_{max} der Pixel im Modellbild zum Durchschnittsfarbwert. Aus der maximalen Distanz erhält man dann, z.B. für Bilder im RGB-Farbraum, im dreidimensionalen Raum die folgende Kugelgleichung:

$$r^2 + g^2 + b^2 = D_{max}^2$$

Anstelle der Backprojection kann man nun zum Farb-Matching eines Modellbildes mit einem Eingabebild alle Pixel ausschließen, deren Farbwert-Distanz $> D_{max}$ ist.

[2] Swain, M.J. and Ballard, D.H.: Color Indexing. International Journal of Computer Vision,vol. 7, Springer, 1991

a) Wenden Sie den oben beschriebenen Algorithmus auf ihre Testbilder aus Aufgabe 2 an. Berechnen Sie dazu D_{max} jeweils im RGB und HSV-Farbraum.

b) Nutzen Sie dieses Mal nur die h, s Werte aus dem HSV Farbraum.

c) Vergleichen Sie ihre Ergebnisse mit denen aus Aufgabe 2.