3. Übung

Histogrammanpassung

In dieser Übung werden Sie das kumulative Histogramm eines Bildes berechnen. Mithilfe dieses Histogramms werden Sie dann ein Bild an das kumulative Histogramm eines anderen Bildes anpassen.

---

1. Lesen Sie die Kapitel 5 (*Punktoperationen - 5.2 Punktoperationen und Histogramme*) und Kapitel 6 (*Filter 6.1 - 6.3*) aus dem Buch "Digitale Bildverarbeitung".
2. Implementieren Sie eine Funktion, die aus einem 8-Bit-Graustufenbild das zugehörige kumulative Histogramm berechnet.
   - **Prototyp:** cumHisto = compute_cumHisto(image, binSize)
   - Das image ist ein 8-Bit-Grauwert Bild.
3. Beantworten Sie folgende Fragen:
   a) Was ist eine homogene und was eine nicht-homogene Punktoperation?
   b) Was ist der Unterschied zwischen Punktoperationen und Filteroperationen?
4. Implementieren Sie eine Funktion, die das Bild01 an das Bild02 mittels Histogrammanpassung angleicht. Hierzu soll das kumulative Histogramm von Bild02 als Referenzverteilung dienen und das Bild01 so verändert werden, dass sein kumulatives Histogramm an die Referenzverteilung angeglichen wird. Schließlich soll das Referenzbild (02) und das verarbeitete Bild (01) und deren kumulativen Histogramme angezeigt werden.
   - **Prototyp:** LUT = match_Histo(img_histo, ref_histo)
   - **img_histo:** Histogramm des anzupassenden Bildes.
   - **ref_histo:** Histogramm des Referenzbildes.
   - **LUT:** Die Lookup Tabelle, welche auf das anzupassende Bild angewendet, die Histogrammanpassung durchführt.
   - Die beiden Eingangshistogramme in Aufgabe 4 müssen normiert sein.

Abgabe
Die Aufgaben werden per Mail an tkocher@htwg-konstanz.de vor der nächsten Übungsstunde abgegeben. Außerdem werden die Lösungen nächstes Mal mündlich präsentiert.

Aufgabe 2

```
7  def compute_cumHisto(img, binSize=1):
8
9      return np.cumsum(bin_Histo(img, binSize))
```

Aufgabe 3 Code

```
24
25 def match_Histo(img_histo, ref_histo):
26     #img_histo . . . original histogram
27     #ref_histo . . . reference histogram
28     #returns the mapping function LUT to be applied to the image
29
30     LUT = np.zeros(shape=256, dtype=np.int)
31
32     for a in range(0, 256):
33         j = 256 - 1
34         while (j >= 0 and img_histo[a] <= ref_histo[j]):
35             LUT[a] = j
36             j -= 1
37
38     return LUT
```

Aufgabe 3 Code

```
24
25 def apply_LUT(img, lut):
26
27     for x in range(0, img.width):
28         for y in range(0, img.height):
29             edit = img.getpixel((x, y))
30             lut_edit = lut[edit].item()
31             img.putpixel((x, y), lut_edit)
32
33     return img
34
35
36 def clamping(value):
37     if value > 255:
38         return 255
39     else:
40         return value
```

Aufgabe 3.a)
Was ist eine homogene und was eine nicht-homogene Punktoperation?

Antwort:
Eine homogene Punktoperation ist eine Operation die unabhängig von der Position oder der Lage eines Pixel durchgeführt werden kann.

Aufgabe 3.b)
Was ist der Unterschied zwischen Punktoperationen und Filteroperationen?

Antwort:
Eine Punktoperation kann *in place* durchgeführt werden. Eine Filteroperation brauch zusätzlichen Speicher um das Original nicht zu verändern um spätere Filteroperationen noch am gleichen Original durchführen zu können.

Aufgabe 4