Aufgabenblatt 6

Einführung in die Bildverarbeitung
Christian Wilms und Simone Frintrop
SoSe 2020

Bei Fragen und Problemen schickt eine Mail an
wilms@informatik.uni-hamburg.de

Ausgabe: 12. Juni 2020 - **Abgabe bis: 19. Juni 2020, 10:00**

Nach der Auswertung der Evaluation ergeben sich **folgende Änderungen für die Übung**:

- Ihr dürft ab sofort auch gelöste Teilaufgaben abgeben und bekommt Teilpunkte. Die Punkte je Teilaufgabe sind im Header der jeweiligen Aufgabe notiert. Beispielsweise bedeutet (5+5+5), dass es drei Teilaufgaben (1-3) gibt, die jeweils 5 Punkte bringen.
- Um zukünftig Missverständnisse zu vermeiden, fügt in den Email-Text eurer Abgabe oder in eine separate txt-Datei eine Liste der von euch bearbeiteten Aufgaben ein. Eine Vorlage zum Ankreuzen (`(x) Aufgabe 1.1` → Aufgabe 1.1 gelöst) findet ihr jeweils am Anfang eines jeden Aufgabenblatts.
- In dieser Woche wird es eine anonymisierte Liste mit Punkten (Matrikelnummer - Punktestand) im Moodle geben. Ab der nächsten Woche wird es jede Woche eine solche Liste mit dem Ergebnis des jeweiligen Aufgabenblatts im Moodle geben.
- Es wird ab dieser Woche jeweils ein kurzes Video mit Folien und Demo zum Aufgabenblatt geben, in dem relevante Python-Funktionen, die wichtige Konzepte aus der Vorlesung abbilden, besprochen und demonstriert werden. In den Aufgabenblättern wird davon ausgegangen, dass ihr diese Videos geschaut habt.

```
 Gelöste Aufgaben:
(_) Aufgabe 1.1
(_) Aufgabe 1.2
(_) Aufgabe 1.3
(_) Aufgabe 1.4
(_) Aufgabe 1.5
(_) Aufgabe 2.1
(_) Aufgabe 2.2
(_) Aufgabe 2.3
(_) Aufgabe 3
(_) Aufgabe 4.1
(_) Aufgabe 4.2
(_) Aufgabe 4.3
(_) Zusatzaufgabe 5.1
(_) Zusatzaufgabe 5.2
(_) Zusatzaufgabe 5.3
```

Aufgabe 1 — Histogramme berechnen - $3+3+3+3+3 = 15$ Punkte - Theorieaufgabe
Gegeben sei das 3-bit Graustufenbild in Abbildung 1. Da es sich um ein 3-bit Graustufenbild handelt, kann das Bild Werte im Bereich $0, 1, \dots 7$ beinhalten. Löst zu dem Bild die folgenden Aufgaben händisch:

1. Ermittelt das nicht normierte Histogramm mit 8 bins $(0, 1, \dots 7)$ und zeichnet es auf.
2. Normiert das erzeugte Histogramm nun und stellt es ebenfalls grafisch dar.

4	6	1	2
3	1	3	7
3	0	3	4
0	7	4	7

Abbildung 1: 3-bit Graustufenbild für Aufgabe 1 mit Pixelwerten als Zahlen.

3. Berechnet auf Basis des normierten Histogramms das kumulierte Histogramm und skizziert es ebenfalls.
4. Berechnet aus einem der Histogramme den Mittelwert des Bildes und notiert dabei euren Rechenweg.
5. Ermittelt aus einem der Histogramme ebenso die Varianz des Bildes und haltet erneut euren Rechenweg fest.

Aufgabe 2 — Varianz im Bild berechnen - $10 + 10 + 5 = 25$ Punkte - Programmieraufgabe
Erlaubte (Sub-)Pakete: `numpy`, `skimage.io`
Wie in der Vorlesung bereits angedeutet lässt sich die Varianz eines Bildes

$$\sigma^2 = \frac{1}{N_{\text{cols}} N_{\text{rows}}} \sum_{x=1}^{N_{\text{cols}}} \sum_{y=1}^{N_{\text{rows}}} \left[I(x,y) - \mu_I \right]^2 \tag{1}$$

auch mit nur einem Durchlauf durch das Bild berechnen:

$$\sigma^2 = \left[\frac{1}{N_{\text{cols}} N_{\text{rows}}} \sum_{x=1}^{N_{\text{cols}}} \sum_{y=1}^{N_{\text{rows}}} I(x,y)^2 \right] - \mu_I^2. \tag{2}$$

Die Definition des Mittelwerts μ_I eines Bildes I könnt ihr den Folien entnehmen. Ein Durchlauf bedeutet dabei, dass jedes Pixel nur einmal betrachtet wird. In Formel 2 ist dies daran zu erkennen, dass der Mittelwert μ_I nicht mehr in jedem Summanden benötigt wird, sondern nur noch einmal am Ende abgezogen wird. Das heißt, dass der Mittelwert im selben Durchlauf parallel zur Doppelsumme in der Varianz berechnet werden kann.

1. Schreibt zunächst eine Funktion, welche die Varianz nach Formel 1 in zwei Durchläufen durch das Bild berechnet. Im ersten Durchlauf soll dabei der Mittelwert μ_I berechnet werden und im zweiten Durchlauf schließlich die Varianz. Nutzt dazu unbedingt zwei verschachtelte `for`-Schleifen pro Durchlauf, um über die Pixel zu iterieren, und nutzt zur Berechnung **keine** Funktionen wie `np.mean()` oder `np.var()`. Testet eure Funktion an dem bekannten Testbild `mandrill.png` aus dem Moodle.
   Tipp: Zur Abschätzung der Richtigkeit euer Ergebnisse, nicht zur Berechnung, könnt ihr das Resultat der Funktion `numpy.var(img)` nutzen.
2. Implementiert nun eine Funktion für die Varianzberechnung in einem Durchlauf nach Formel 2. Nutzt dazu erneut unbedingt zwei verschachtelte `for`-Schleifen pro Durchlauf, um über die Pixel zu iterieren, und nutzt zur Berechnung erneut **keine** Funktionen wie `np.mean()` oder `np.var()`. Testet eure Funktion wieder am Testbild `mandrill.png`.
3. Vergleicht nun eure Implementationen im Hinblick auf das Ergebnis und auf die Ausführungszeit. Findet und notiert eine Erklärung, warum die Ergebnisse leicht voneinander abweichen. Zum Vergleich der Ausführungszeiten soll je Funktion 10 mal die Varianz auf dem Testbild `mandrill.png` berechnet werden. Für die Zeitmessung solltet ihr auf die Funktion `time.time()`, die auf dem Aufgabenblatt 1 beschrieben wurde, zurückgreifen.

Aufgabe 3 — Varianz in einem Durchlauf - 20 Punkte - Theorieaufgabe
Beweist nun, dass die beiden Formeln 1 und 2 zur Berechnung der Varianz gleich sind. Startet dazu bei der Formel 1 und zerlegt mit Hilfe der binomischen Formeln den Summanden der Doppelsumme. Überlegt anschließend, was die drei einzelnen Teile bedeuten und wie sie vereinfacht werden können.
Tipp: Schreibt nach der initialen Zerlegung des Summanden die Formel mit drei Doppelsummen, um die einzelnen Teile einfacher zu untersuchen.

Aufgabe 4 — Histogrammausgleich - $5 + 20 + 15 = 40$ Punkte - Programmieraufgabe
Erlaubte (Sub-)Pakete: `numpy`, `skimage.io`, `matplotlib`, `time`
Zur Verbesserung des Kontrasts im Bild wurde in der Vorlesung der Histogrammausgleich (histogram equalization) vorgestellt. Dieser Histogrammausgleich soll im Rahmen dieser Aufgabe implementiert und auf verschiedene Bilder angewendet werden.
Hinweis: Bei allen Teilen dieser Aufgabe sollen zur Visualisierung der Bilder mit `matplotlib.pyplot.imshow()` die Parameter `vmin=0` und `vmax=255` gesetzt werden (siehe auch Aufgabenblatt 4, Aufgabe 1).

1. Ladet euch das `bildverbesserung.zip` aus dem Moodle herunter und ladet die beiden Bilder `bild1.png` und `bild2.png` in Python. Beide Bilder wurden bereits in Aufgabe 1 von Aufgabenblatt 4 so bearbeitet, dass der Kontrast verbessert wurde. Ermittelt für beide Bilder jeweils das normierte Histogramm über die Funktion `numpy.histogram()`. Plottet zudem jeweils die Histogramme mit Hilfe der Funktion `matplotlib.pyplot.hist()`.
2. Implementiert nun selbst eine Funktion, die auf einem Bild einen Histogrammausgleich durchführt (nutzt nicht die Funktion `skimage.exposure.equalize_hist()`). Verwendet dazu das normierte Histogramm und berechnet euch die Transformationsfunktion $T(r_k)$ für jeden Grauwert nach der Formel aus der Vorlesung. Die Transformationsfunktion selbst kann beispielsweise als 1D-Array oder Liste gehalten werden, wobei der Listen- bzw. Arrayindex dem x-Wert und der Listen- bzw. Arrayeintrag dem y-Wert der Transformationsfunktion entspricht. Achtet darauf, dass das Ergebnis am Ende wieder im Bereich $0, 1, 2, \ldots, 255$ liegen muss. Wendet euren Histogrammausgleich auf die beiden Bilder `bild1.png` und `bild2.png` an. Erstellt und visualisiert erneut die jeweiligen normierten Histogramme. Was hat sich verändert? Zeigt auch die jeweiligen Ergebnisbilder an.
3. Visualisiert nun die Transformationsfunktionen der beiden Bilder. Dazu könnt ihr die Transformationsfunktion je Bild als weitere Rückgabe eurer Funktion zum Histogrammausgleich setzen. Das Plotten selbst kann wie in folgendem Beispiel dann über die Funktion `matplotlib.pyplot.plot` durchgeführt werden:

```
import matplotlib.pyplot as plt

mapping = [2,5,3,5,6] #Funktion als: 0->2, 1->5,...

def f(x): #Funktion als Pythonfunktion definiert
    return x**2

plt.figure(1) #erste Figure, ggf.erhöhen
plt.plot(range(len(mapping)), mapping) #Plottet Punkte, die verbunden werden;
    erster Parameter sind die x-Werte, zweiter Parameter die y-Werte
plt.plot(range(len(mapping)), list(map(f, range(len(mapping))))) #map wendet
    eine Funktion, hier f, auf alle Werte einer List an
plt.show()
```

   Vergleicht nun die Ergebnisse eures Histogrammausgleichs auf den Bildern `bild1.png` und `bild2.png` mit dem Ergebnissen (ggf. der Musterlösung) aus Aufgabe 1 von Aufgabenblatt 4. Was fällt euch auf? Vergleicht auch die Transformationsfunktionen mit den Intensitätstransformationen der Musterlösung aus Aufgabe 1 von Aufgabenblatt 4. Für `bild1.png` war dies

$$255^{0.8} \cdot \mathrm{I}(x, y)^{0.2}$$

   und für `bild2.png`

$$\frac{255 \cdot e^{\frac{\mathrm{I}(x,y)}{255} \cdot 16 - 10}}{1 + e^{\frac{\mathrm{I}(x,y)}{255} \cdot 16 - 10}},$$

   adaptiert auf den Wertebereich $0, \ldots, 255$ eines Bildes $\mathrm{I}(x, y)$. Zum Vergleich bietet es sich an, die jeweilige Transformationsfunktion sowie die jeweilige Intensitätstransformationen zu plotten und die Kurvenformen zu vergleichen.

Zusatzaufgabe 5 — Lokale Bildverbesserung - $5 + 15 + 10 = 30$ Punkte - Programmieraufgabe
Erlaubte (Sub-)Pakete: `numpy`, `skimage.io`, `matplotlib`
Die Anwendung von des Histogrammausgleichs global auf das ganze Bild mit einer Transformationsfunktion ist manchmal nicht optimal, da unterschiedliche Bildbereiche verschiedene Transformationsfunktionen

zur Optimierung des Kontrasts benötigen. Aus diesen Anforderungen lassen sich Formen der lokaler oder adaptiver Histogrammausgleiche entwickeln, bei denen verschiedene Bildbereiche mit unterschiedlichen Transformationsfunktionen behandelt werden.
Hinweis: Bei allen Teilen dieser Aufgabe sollen zur Visualisierung der Bilder mit `matplotlib.pyplot.imshow()` die Parameter `vmin=0` und `vmax=255` gesetzt werden (siehe auch Aufgabenblatt 4, Aufgabe 1).

1. Ladet zunächst das Bild `moon.png` aus dem Moodle in Python und visualisiert es. Wendet eure Funktion zum Histogrammausgleich aus Aufgabe 4 auf das Bild an und visualisiert das Ergebnis.
2. Führt nun eure Funktion zum Histogrammausgleich aus Aufgabe 4 auf einzelnen Teilbereichen des Bildes unabhängig voneinander durch. Zerlegt das Bild dazu virtuell in 4×4 Kacheln zu je 64×64 Pixeln, indem ihr nur die entsprechenden Teilbereiche des Bildes an eure Funktion zum Histogrammausgleich übergebt. Verfahrt mit den Ergebnis entsprechend, sodass jede Anwendung eurer Funktion zum Histogrammausgleich nur einen Teilbereich des Ergebnisses ermittelt. Achtet darauf, dass sich die Kacheln nicht überlappen. Wie verändert sich das visuelle Ergebnis? Ignoriert dabei die teilweise entstandenen künstlichen Kanten innerhalb des Bildes.
3. Verkleinert nun die Kachelgröße auf 32×32, 16×16, 8×8, 4×4 und 2×2. Die Anzahl der Kacheln verdoppelt sich pro Dimension entsprechend bei jedem Schritt. Wie verändert sich das visuelle Ergebnis mit jedem Schritt? Versucht eine Erklärung zu finden.