

Aufgabenblatt 6

Übung zu Einführung in die Bildverarbeitung

Laszlo Korte, Thang Le, Juri Schalück, Maris Hillemann, Tim Rolff, Christian Wilms
SoSe 2023

Ausgabe: 2. Juni 2023 - **Abgabe bis: 12. Juni 2023, 10:00**

Abgabe per Moodle

Juri Schalück (Gruppe 2)	oschalue@informatik.uni-hamburg.de
Thang Le (Gruppe 3)	phuoc.thang.le@uni-hamburg.de
Laszlo Korte (Gruppe 4)	9korte@informatik.uni-hamburg.de
Maris Hillemann (Gruppe 5, 6)	maris.nathanael.hillemann@studium.uni-hamburg.de
Tim Rolff (Gruppe 1)	tim.rolff@uni-hamburg.de
Christian Wilms (Gruppe 1)	christian.wilms@uni-hamburg.de

Aufgabe 1 — Varianz in einem Durchlauf - 30 Punkte - Theorieaufgabe

Wie in der Vorlesung bereits angedeutet lässt sich die Varianz eines Bildes

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

auch mit nur einem Durchlauf durch das Bild berechnen:

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

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.

Beweist nun, dass die beiden Formeln 1 und 2 zur Berechnung der Varianz gleich sind. Startet dazu bei der Formel 1 und zerlegt den Summanden der Doppelsumme. Überlegt anschließend, was die drei einzelnen Teile bedeuten und wie sie vereinfacht werden können.

Tipp: Notiert nach der initialen Zerlegung des Summanden die Formel mit drei Doppelsummen, um die einzelnen Teile einfacher zu untersuchen.

Aufgabe 2 — Histogrammausgleich - 5+25+15 = 45 Punkte - Programmieraufgabe

Erlaubte (Sub-)Pakete: `numpy`, `skimage.io`, `matplotlib`, `time`, `math`

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.

1. Ladet euch das `bildverbesserung.zip` aus dem Moodle herunter und ladet die beiden Bilder `bild1.png` und `bild3.png` in Python. Beide Bilder wurden bereits in Aufgabe 1 von Aufgabenblatt 4 so bearbeitet, dass der Kontrast verbessert wurde. Die Invertierung von `bild3.png` aus Aufgabe 1 von Aufgabenblatt 4 wurde bereits durchgeführt. Ermittelt nun für beide Bilder jeweils das normierte Histogramm über die Funktion `numpy.histogram()`. Plottet zudem jeweils die Histogramme mit Hilfe der Funktion `matplotlib.pyplot.hist()`.

Hinweis: Die Funktion `matplotlib.pyplot.hist()` erwartet als Eingabe kein Histogramm, sondern das Bild als 1D-Array (s. Folien).

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 auch 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 $0, 1, 2, \dots, 255$ liegen muss. Wendet euren Histogrammausgleich auf die beiden Bilder `bild1.png` und `bild3.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 `bild3.png` mit den Ergebnissen der Musterlösung aus Aufgabe 1 von Aufgabenblatt 4 (s. Abb. 1). Was fällt euch auf? Vergleicht auch die Transformationsfunktionen mit den Intensitätstransformationen der Musterlösung aus Aufgabe 1 von Aufgabenblatt 4. Für `bild1.png` war dies

$$255 \cdot \left(\frac{I(x, y)}{255} \right)^{0.3}$$

und für `bild3.png`

$$\begin{cases} I(x, y) & \text{if } I(x, y) < 64 \\ 64 & \text{if } I(x, y) \geq 64 \wedge I(x, y) < 128 \\ \frac{I(x, y) - 128}{195 - 128} (254 - 64) + 64 & \text{if } I(x, y) \geq 128 \wedge I(x, y) < 196 \\ 255 & \text{if } I(x, y) \geq 196 \end{cases}$$

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

Aufgabe 3 — Histogramme für Farbbilder - 5+10+5+5 = 25 Punkte - Programmieraufgabe Erlaubte (Sub-)Pakete: `numpy`, `skimage.io`

Für die Berechnung von Histogrammen über Farbbilder gibt es zwei Möglichkeiten. Zum einen können drei Histogramme erzeugt werden, die jeweils einen der Farbkanäle abdecken. Somit entsteht ein Histogramm über den Rot-Kanal, das die Rot-Werte im Bild repräsentiert, sowie analoge Histogramme zu Grün-Kanal und Blau-Kanal. Alternativ dazu kann ein dreidimensionales Histogramm erzeugt werden. Dies enthält für jeden der drei Farbkanäle eine Achse und hat die Form eines Würfels. Über die drei Werte für Rot (x -Achse), Grün (y -Achse) und Blau (z -Achse) wird der Behälter ermittelt, der inkrementiert werden muss. Im Extremfall gibt es daher für jeden RGB-Farbwert einen Behälter ($256 * 3$). Eine beispielhafte Visualisierung für ein dreidimensionales Histogramm mit $6 * 3$ Behältern ist in Abbildung 2 zu sehen.



(a)



(b)

Abbildung 1: Ergebnisse der manuellen Kontrastverbesserung aus Aufgabe 1 des Aufgabenblatts 4.

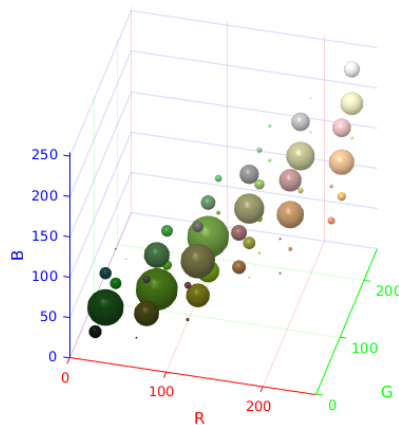


Abbildung 2: Beispiel eines dreidimensionalen Histogramms. Die Größe der Kugel repräsentiert die Anzahl der in diesen Behälter fallenden Pixel, die Farbe hingegen die mittlere Farbe des Behälters.

1. Ladet die beiden Bilder `nebel.png` und `farbverlaeuft.png` aus dem Moodle in Python. Die Bilder zeigen offensichtlich unterschiedliche Inhalte und haben auch farblich wenig Gemeinsamkeiten. Berechnet nun jeweils drei Farbhistogramme pro Bild: ein Histogramm über den Rot-Kanal, ein Histogramm über den Grün-Kanal und ein Histogramm über den Blau-Kanal.
2. Berechnet nun für beide Bilder jeweils ein dreidimensionales Histogramm, wie oben beschrieben. Implementiert dazu selbst eine Funktion, welche die Berechnung durchführt und nutzt nicht `numpy.histogramdd()`. Verwendet für die Erzeugung des dreidimensionalen Histogramms nur 4 Behälter je Achse, also 4^3 Behälter für das gesamte Histogramm. Folglich hat das Histogramm eine Shape von $(4, 4, 4)$.
3. Vergleicht nun die Histogramme der beiden Bilder. Sind die Histogramme für den Rot-Kanal, den Grün-Kanal und den Blau-Kanal jeweils gleich zwischen den Bildern? Wie sieht es mit den beiden dreidimensionalen Histogrammen aus? Findet eine Erklärung für die Ergebnisse.
4. Welcher Farbbereich ist in jeweiligen 3D-Histogrammen dominant? Stimmt dies mit eurem visuellen Eindruck überein?

Zusatzaufgabe 4 — Adaptive Bildverbesserung - $5+20+5 = 30$ Punkte - Programmieraufgabe Erlaubte (Sub-)Pakete: `numpy`, `skimage.io`, `matplotlib`

Die Anwendung 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 lässt sich ein lokaler, adaptiver Histogrammausgleich entwickeln, bei dem jedes Pixel mit einer individuellen Transformationsfunktion behandelt wird, die von seiner direkten Nachbarschaft abhängt. Daher wird für jedes Pixel eine quadratische Nachbarschaft der Größe $n \times n$, die um das Pixel zentriert ist, aus dem Bild extrahiert und mittels dieser eine Transformationsfunktion berechnet. Das zentrale Pixel wird dann mit dieser Transformationsfunktion verändert.

1. Ladet zunächst das Bild `moon.png` aus dem Moodle in Python und visualisiert es. Wendet eure Funktion zum Histogrammausgleich aus Aufgabe 2 auf das Bild an und visualisiert das Ergebnis.
2. Nutzt nun euren Code zum Histogrammausgleich aus Aufgabe 2 und führt den oben beschriebenen adaptiven Histogrammausgleich durch. Achtet darauf, dass Pixel am Rand keine vollständige Nachbarschaft der Größe $n \times n$ haben. Werte außerhalb des Bildes sollen dann ignoriert werden, sodass die Nachbarschaft kleiner wird. Wie verändert sich das visuelle Ergebnis im Vergleich zur ersten Teilaufgabe, wenn ihr für n einen Wert von 43 annehmt? Antwortet kurz schriftlich.
3. Variiert den Wert von n . Was passiert, wenn dieser substantiell größer oder kleiner wird?