Aufgabenblatt 3.2

Praktikum Computer Vision
WiSe 2023/24

Christian Wilms

Auf diesem Aufgabenblatt bietet es sich an, eine `ipynb`-Datei, also ein Notebook, für die Aufgaben 2 bis 4 zu verwenden. Die Ergebnisse der Teilaufgaben sollten in Kommentaren oder Markdown-Zellen notiert werden.

Aufgabe 1 — Erste Schritte mit Farbbildern

1. Ladet das Farbbild `cat.png` aus dem Moodle herunter.
2. Zerlegt das Bild in seine einzelnen Farbkanäle, addiert die Farbkanäle gewichtet zu einem Graustufenbild zusammen und lasst euch das Ergebnis anzeigen. Ist das Ergebnis so wie ihr es erwartet? Hinweis: Achtet auf Überläufe und ändert für die Berechnung ggf. den Elementtyp des Arrays mit der Methode `astype` des Arrays.
3. Was passiert, wenn ihr die Farbkanäle in einer falschen Reihenfolge wieder zu einem Farbbild-Bild zusammensetzt? Lasst euch das Ergebnis anzeigen. Nutzt die Funktion `np.dstack`, die einzelne 2D-Arrays, hier die Farbkanäle, zu einem 3D-Array zusammensetzt:

```
>>> b.shape
(64,64)
>>> c.shape
(64,64)
>>> d.shape
(64,64)
>>> a = np.dstack([b,c,d])
>>> a.shape
(64,64,3)
```

4. Wie verändert sich der visuelle Eindruck, wenn ihr das RGB-Bild invertiert? Invertieren bedeutet hier die Umkehrung der einzelnen Einträge eines Arrays im Wertebereich $0 \dots 255$.
5. Wie kann der kanalweise Mittelwert des RGB-Bildes berechnet werden, ohne das Bild explizit in Kanäle zu zerlegen? Der kanalweise Mittelwert ist ein Vektor/1D-Array mit 3 Elementen, wobei jedes Element dem Mittelwert über einem der Farbkanäle entspricht.
   Tipp: Schaut euch die entsprechende Funktion von NumPy sowie deren Parameter einmal genauer an.

Aufgabe 2 — Farbbasierte Bildklassifikation

Führt die Bildklassifikation aus Aufgabe 3.1.1 erneut durch, diesmal jedoch mit den Farbdaten (s. Moodle). Als Merkmal sollt ihr nun den kanalweisen Mittelwert nutzen. Beachtet erneut die Hinweise zum Einbinden und Nutzen der Daten. Verbessern sich die Ergebnisse?
Hinweis: Da der kanalweise Mittelwert ein Vektor/Array mit drei Komponenten ist, benötigt ihr hier in jedem Falle die euklidische Distanz, um das ähnlichste Trainingsbild zu finden.

Aufgabe 3 — Bildklassifikation mit Farb-Histogrammen

Verändert die Berechnung aus der vorherigen Aufgabe, sodass die drei einzelnen 1D-Farb-Histogramme zusammen den Merkmalsvektor ergeben. Nutzt dabei die Funktion `np.hstack`, um die einzelnen Histogramme zu einem Merkmalsvektor zu verbinden (s.u.). Welche Anzahl an Behältern (gleiche Anzahl Behälter für alle Farben) bringt die besten Ergebnisse in diesem Beispiel? Nutzt nur 2er-Potenzen für die Anzahl an Behältern, um eine Überanpassung zu vermeiden.

```
>>> b.shape
(10,)
>>> c.shape
(10,)
>>> d.shape
(10,)
>>> a = np.hstack([d,c,b])
>>> a.shape
(30,)
```

Aufgabe 4 — Zusatzaufgabe: Komplexere Merkmale und mehr Nachbarn

1. Verwendet nun als Merkmal **ein** 3D-Farb-Histogramm. Nutzt unbedingt eine kleine (≤ 8) Anzahl an Behältern je Dimension! Rollt das 3D-Farb-Histogramm in ein 1D-Array ab, bevor ihr es als Merkmal nutzt. Welche Histogramme funktionieren besser?
2. Anstelle des direkten nächsten Nachbarn, verwendet nun die k ($k = 3, 5, 8, 10$) nächsten Nachbarn zur Bestimmung der Vorhersage. Das Label wird dabei nach dem Mehrheitsprinzip unter den k Nachbarn gewählt. Was passiert, wenn ihr $k = 60$ setzt?
   Hinweis: Über 70% korrekte Zuordnungen sind möglich.