Aufgabenblatt 5

Praktikum Computer Vision
SoSe 2018

Christian Wilms

03. Mai 2018

Aufgabe 1 — Entrauschen durch Faltung

1. Ladet euch zunächst das Bild `noisyLenna.png` aus dem CommSy herunter.
2. Implementiert die Faltung des Bildes `noisyLenna.png` mit einem 3×3-Mittelwertfilter selbst (bspw. mit Hilfe von Schleifen). Behandelt den Rand so, dass die fehlenden Pixel mit dem Wert 0 aufgefüllt werden (Trauerrand um das eigentliche Bild). Es werden also immer 9 Pixel betrachtet.
3. Vergleicht eure Faltung nun mit der bereits in SciPy implementierten Faltung (`scipy.ndimage.filters.convolve`). Erstellt dazu ein 3×3-Array mit den Einträgen $\frac{1}{9.0}$. Messt dazu mit Hilfe der Funktion `time.time()` die Ausführungszeit beider Faltungen mit der selben Maske auf dem selben Bild. Die Funktion `time.time()` gibt die Zeit, die seit dem 01.01.1970 verstrichen ist, in Sekunden wieder. Ein Beispiel für die Nutzung von `time.time()` zeigt folgender Quelltext:

```
>>> import time
>>> tic = time.time()
>>> #your magic here
>>> toc = time.time()
>>> diff = toc-tic
```

4. Experimentiert mit der SciPy Faltung, indem ihr Masken verschiedener Größen nutzt, und vergleicht die Ergebnisse.
5. Zusatzaufgabe: Implementiert eure Faltung so, dass beliebige Masken mit ungerader Seitenlänge angewendet werden können.

Aufgabe 2 — Zusatzaufgabe: Gaußfilter und Hybridbilder

1. Um nicht alle Pixel gleichermaßen zu gewichten, werden die betrachteten Pixel oft mit einer auf dem mittleren Pixel zentrierten Gaußverteilung (2D-Glocke) gewichtet. Es entsteht der Gaußfilter, der in scikit-image unter `skimage.filters.gaussian` bereits implementiert ist. Wendet diesen auf das Bild `noisyLenna.png` an und experimentiert mit verschie-

denen Standardabweichung. Was sind die Auswirkungen? Unterscheidet sich das Ergebnis von der Anwendung des Mittelwertfilters?

2. Was geht bei der Faltung mit dem Gaußfilter oder Mittelwertfilter verloren? Findest es heraus, indem ihr das Ergebnis der Faltung vom Eingangsbild abzieht. Verändert vor der Faltung den Datentypen des Bild zu `np.float` und ändert den Wertebereich auf $0 \ldots 1$.
3. Ladet nun die Bilder `einstein.jpg` und `monroe.jpg` aus dem CommSy herunter. Wendet auf die Bilder Gaußfilter mit unterschiedlichen Standardabweichungen an und berechnet erneut die Differenz des Faltungsergebnis zum jeweiligen Eingangsbild. Kombiniert diese beiden Ergebnisse nun miteinander, jedoch über Kreuz: Differenz von Einstein mit Faltungsergebnis von Monroe und umgekehrt. Das Kombinieren kann durch eine Addition und eine Division durch 2.0 erfolgen.
4. Welcher Effekt entsteht? Tipp: Betrachtet die Bilder aus kurzer und weiter Distanz betrachtet.
5. Wie ist der Effekt zu erklären?

Aufgabe 3 — Kanten finden mit Sobel

1. Ladet das Bild `Lenna.png` aus dem CommSy herunter und wendet die Sobelfilter in x- und y-Richtung von scikit-image auf das Bild an. Visualisiert die beiden Ergebnisse.
2. Führt dasselbe auch für die Bilder `noisyLenna.png` aus Aufgabe 1 durch sowie für das Ergebnis der Faltung jenes Bildes mit einem Gaußfilter mit Standardabweichung 5 (`img2 = skimage.filters.gaussian(img, 5)`). Worin unterscheiden sich die Ergebnisse und wie ist das zu erklären?
3. Führt die jeweiligen Einzelergebnisse nun zusammen, indem ihr die Gradientenstärke berechnet und visualisiert.

Aufgabe 4 — Zusatzaufgabe: Histogram of Oriented Gradients

1. Benutzt euren Quelltext aus Aufgabe 3 und fügt die Berechnung für die Orientierung der Gradienten hinzu.
2. Schreibt nun eine Funktion, die gegeben die Gradientenstärken und deren Orientierungen ein Histogramm von Gradientenorientierungen (HOG) erzeugt.
3. Schreibt eine weitere Funktion, die ein Bild in $n \times n$ Kacheln gleicher Größe zerlegt, und für jede Kachel das HOG berechnet. Abschließend werden alle HOGs zu einem langen Deskriptor zusammengeführt.
4. Eignet sich das HOG auch als Deskriptor zur Klassifikation? Probiert es aus, indem ihr die Klassifikation der Graustufenbilder aus Aufgabenblatt 3.1 erneut durchführt, diesmal aber nur das HOG als Deskriptor nutzt.

Aufgabe 5 — Template Matching

1. Ladet die Bilder `Lenna.png` und `auge.png` aus dem CommSy herunter. Das Bild `auge.png` soll als Template dienen, das im Bild `LennaRGB.png`

gefunden werden soll. Werden dabei beide Augen gefunden oder nur das eine und was heißt überhaupt gefunden in diesem Zusammenhang?

2. Führt das Template Matching mit Hilfe der bereits in scikit-image enthaltenen Funktion `skimage.feature.match_template` durch. Welche Form hat das Ergebnis? Welchen Wertebereich hat es? Visualisiert das Ergebnis wie ein übliches RGB-Bild, d.h. ohne `cmap='Greys_r'`. Könnt ihr nun die Fragen aus der ersten Teilaufgabe beantworten?
3. Findet die Position der größten Übereinstimmung des Templates im Bild. Tipp: die Funktion `np.unravel_index` könnte sich zur Lösung als hilfreich erweisen.
4. Zusatzaufgabe: Markiert das Maximum im Bild mit einem Punkt.
   Tipp: Nutzt die `plot`-Funktion von matplotlib.
5. Nutzt nun euren Quelltext, um wirkliche Probleme zu lösen. Kennt ihr die Bilder aus den Where's Wally-Büchern? Es handelt sich dabei um typische Wimmelbilder, wobei sich unter den vielen dargestellten Personen auch stets Wally mit seiner rot-weißen Bommelmütze und seinem rot-weiß geringelten Pulli versteckt. Versucht Wally im Bild `whereIsWally1.jpg` aus dem CommSy zu finden. Nutzt dazu Template Matching und das entsprechende Template-Bild `wally.png` aus dem CommSy. Achtet darauf, dass es sich um ein RGB-Bilder handelt.
6. Warum wird das Gesicht von Wally auf der Briefmarke der Postkarte nicht gefunden?

Aufgabe 6 — Zusatzaufgabe: Leitwerke finden mit Template Matching

Im Rahmen dieser Aufgabe sollen Originalbilder (27 Bilder, `flugzeuge.zip`) von Flugzeugen Airlines (3 Klassen: LH,HK,Thai) zu geordnet werden. Dies lässt sich am einfachsten über die Leitwerke der Flugzeuge machen, da diese jeweils einen für die Airline charakteristischen Anstrich haben.

1. Findet zunächst die Leitwerke, indem ihr auf dem binären Bild `leitwerkMaske.png` Gradientenstärken mit Hilfe der Sobelfilter ermittelt und so das Kantentemplate eines Leitwerks erzeugt. Ermittelt die Gradientenstärken ebenfalls in den 27 Bildern ($\rightarrow$ Kantenbilder) durch Anwendung der Sobelfilter auf jedem Farbkanal einzeln. Nehmt als endgültige Gradientenstärke eines Pixels den jeweils maximalen Wert der drei einzelnen Gradientenstärken. Führt nun ein Template Matching mit dem Kantenbild und dem Kantentemplate durch. Das Kantentemplate sollte nun im Kantenbild an der Stelle am besten passen, an der im Originalbild das Leitwerk ist. Hinweis: Für gute Ergebnisse solltet ihr das Kantenbild auch gespiegelt verarbeiten und das Kantentemplate in skalierten Versionen mit dem Kantenbild matchen (Skalierungsfaktoren: $0.8, 0.81, \ldots, 1.19, 1.20$).
2. Findet so die Stelle im Originalbild, an der das Kantentemplate am besten passt: Koordinate der linken oberen Ecke sowie Höhe und Breite (achtet dafür auf die Skalierung des Kantentemplates!).
3. Erstellt nun ein binäres Bild, das erstmal nur Hintergrund enthält und die gleiche Größe wie das Originalbild hat. Skaliert nun das binäre Bild `leitwerkMaske.png` mit dem Parameter`order=0` auf die Größe des op-

timal passenden Kantentemplates und setzt die skalierte Version in das neue binäre Bild ein. Es entsteht so eine Maske für das Leitwerk oder ein Teil des Leitwerks im Originalbild (→ Leitwekbereich).

4. Berechnet nun den bandweisen Mittelwert des Leitwerkbereichs im Originalbild und vergleicht ihn mit den Mittelwerten der Leitwerke der drei Airlines:

   LH `np.array([ 77.22019439, 63.79567862, 72.68281938])`

   HK `np.array([ 145.12244732, 58.59563909, 47.44520381])`

   Thai `np.array([ 109.28658384, 60.70566164, 131.30926374])`