Aufgabenblatt 9

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: 3. Juli 2020 - **Abgabe bis: 10. Juli 2020, 10:00**

```
 Gelöste Aufgaben:
(_) Aufgabe 1.1
(_) Aufgabe 1.2
(_) Aufgabe 2.1
(_) Aufgabe 2.2
(_) Aufgabe 2.3
(_) Aufgabe 2.4
(_) Aufgabe 2.5
(_) Aufgabe 3
(_) Aufgabe 4.1
(_) Aufgabe 4.2
(_) Aufgabe 4.3
(_) Aufgabe 4.4
(_) Aufgabe 4.5
(_) Zusatzaufgabe 5.1
(_) Zusatzaufgabe 5.2
(_) Zusatzaufgabe 5.3
```

Aufgabe 0 — Finale Evaluation und Terminfindung für Fragerunde vor Klausur

1. **NEU:** Nehmt bitte an der finalen Lehrevaluation zur **Vorlesung** über folgenden Link teil: Link zur Evaluation der **Vorlesung** .
2. **NEU:** Nehmt bitte an der finalen Lehrevaluation zur **Übung** über folgenden Link teil: Link zur Evaluation der **Übung** .
3. Wir wollen gerne eine letzte Fragerunde als Zoom-Meeting kurz vor der Klausur organisieren, um letzte Fragen zu klären. Damit möglichst viele von euch daran teilnehmen können, stimmt bis zum 10. Juli in folgendem Doodle über den Termin ab: Link zum Doodle .

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Abbildung 1: Grauwertbild für Aufgabe 1 mit Pixelwerten als Zahlen.

Aufgabe 1 — Kanten mit Laplace-Filter berechnen - $10 + 5 = 15$ Punkte - Theorieaufgabe
Gegeben seien für diese Aufgabe das Grauwertbild in Abbildung 1. Die nachfolgenden Teilaufgaben sind händisch durchzuführen.

1. Wendet den Laplace-Operator
$$l = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$
auf das Bild in Abbildung 1 an und normiert das Ergebnis auf den Wertebereich $-1, \ldots, 1$. Geht an den Rändern von einem zero-padding aus.
2. Wie könnt ihr aus diesem Ergebnis die Kanten des Bildes bestimmen? Beschreibt dies kurz und markiert in einer Skizze alle Pixel, die so als Kante bestimmt werden.

Aufgabe 2 — Laplace-Filter und Rauschen - 2+3+5+10+10 = 30 Punkte - Programmieraufgabe
Erlaubte (Sub-)Pakete: `numpy`, `skimage.io`, `skimage.filters`, `skimage.util`, `matplotlib`
Im Rahmen dieser Aufgabe wird die Anfälligkeit des Laplace-Filters gegenüber Rauschen untersucht und das Ergebnis des Laplace-Filters mit dem Sobel-Filter verglichen.

1. Ladet das Testbild `mandrill.png` aus dem Moodle in Python und bringt es in den Wertebereich $0, \ldots, 1$. Erstellt euch aus diesem Originalbild nun zwei weitere Varianten des Bildes. Eine Variante soll mit Hilfe eines Gauß-Filters ($\sigma = 3$) weichgezeichnet werden. Die andere Variante soll mit Gaußschem Rauschen ($\sigma^2 = 0.01$) verrauscht werden. Nutzt dazu erneut die Funktion `skimage.util.random_noise()` (s. Aufgabenblatt 7). So entstehen drei Varianten des Bildes: das Originalbild (Variante 1), eine weichgezeichnete Darstellung des Originalbildes (Variante 2) und eine verrauschte Darstellung des Originalbildes (Variante 3). Zeigt die drei Bilder auch an.
2. Wendet nun den Laplace-Filter (`skimage.filters.laplace()`) auf jedes der drei Bilder an und visualisiert die Ergebnisse. Beschreibt und begründet kurz, wie und warum sich die Ergebnisse unterscheiden.
3. Ist es möglich, das Ergebnis der Anwendung des Laplace-Filters auf das verrauschte Bild (Variante 3) zu verbessern? Probiert es aus, indem ihr das verrauschte Bild mit einem Gauß-Filter weichzeichnet und bestimmt ein gut passendes σ. Visualisiert das Ergebnis.
4. Extrahiert nun aus dem Ergebnis des Laplace-Filters auf dem zu Beginn weichgezeichneten Originalbild (Variante 2) die Kanten, indem ihr für jedes Pixel untersucht, ob jeweils zwei sich gegenüberliegende Nachbarn des Pixels unterschiedliche Vorzeichen aufweisen. Ist dies der Fall, soll für das zentrale Pixel eine Kante angenommen werden (1 im Ergebnis), sonst soll das Ergebnis 0 sein. Setzt außerdem einen Schwellenwert für den Betrag der Differenz zwischen den beiden verglichenen Nachbarn ein, um unwichtige Ergebnisse von wichtigen zu trennen. Gelingt es euch, die wichtigen Kanten des Gesichts mit einem geeigneten Schwellenwert zu extrahieren? Erhöht nun das σ im initialen Weichzeichnen des Bildes und beobachtet die Veränderungen auf das Ergebnis der Kantenerkennung. Passt dazu den Schwellenwert für jedes neue σ an. Welches σ liefert gute Ergebnisse in Bezug auf das Auffinden der wichtigen Kanten im Bild?
Hinweis: Betrachtet die Randpixel nicht bei der Untersuchung auf Kanten nach dem Laplace-Filter, da diese nicht unbedingt gegenüberliegende Nachbarn haben.
5. Vergleicht das Ergebnis nun mit der Gradientenstärke nach Anwendung des Sobel-Filters (`skimage.filters.sobel()`). Führt auch hier einen Schwellenwert ein, der unwichtige von wichtigen Kanten trennt. Wie unterscheiden sich die Ergebnisse?

Aufgabe 3 — Fourier-Transformierte berechnen - 20 Punkte - Theorieaufgabe
Ermittelt für die diskrete Sequenz
$$f(x) = \begin{cases} 1 & x = 0, 2 \\ 0 & x = 1, 3 \end{cases}$$
mit der Länge $M = 4$ die Fourier-Transformierte $F(u)$ für $u = 0, 1, 2, 3$. Nutzt dazu die eindimensionale Diskrete Fourier-Transformation (DFT). Gebt unbedingt auch den Rechenweg an.
Tipp 1: Nutzt die Eulersche Relation, um eine komplexe Zahl $r \cdot \mathrm{e}^{\mathrm{i}\varphi}$ in die Form $r \cdot (\cos\varphi + \mathrm{i} \cdot \sin\varphi)$ umzurechnen. Dies erleichtert euch die Interpretation.
Tipp 2: Das Ergebnis sollte nur noch Realteile beinhalten.

Aufgabe 4 — Interpretation der Spektren - 10+5+5+5+10 = 35 Punkte - Programmieraufgabe
Erlaubte (Sub-)Pakete: `numpy`, `skimage.filters`, `matplotlib`
Im Fourierspektrum (*magnitude spectrum*) der Fourier-Transformierten eines Bildes sind die Stärken verschiedener Frequenzen in jenem Bild repräsentiert. Daraus lassen sich Rückschlüsse auf Bildinhalte ziehen, wie ihr im Rahmen dieser Aufgabe feststellen werdet.

1. Schreibt zunächst eine Funktion, die ein komplett schwarzes Bild (Wertebereich $0, \ldots, 1$) der Größe 256×256 erzeugt. Zudem soll durch die Angabe eines Mittelpunkts (x- und y-Koordinate) sowie von Höhe und Breite beim Aufruf der Funktion ein weißes (Wert 1) Rechteck in das Bild eingefügt werden. Testet eure Funktion, indem ihr etwa in die Mitte des Bildes ein Rechteck der Größe 5×10 (Höhe $\times$ Breite) setzt und das erzeugte Bild visualisiert.
2. Ermittelt nun mit Hilfe der entsprechenden Funktion von NumPy die Fourier-Transformierte des zuvor erzeugten Bildes. Wie hoch ist der Wert der Fourier-Transformierten an der Stelle $u = 0, v = 0 \rightarrow F(0, 0)$? Interpretiert diesen Wert.
   Tipp: Nutzt die Formel für die DFT, um den Wert besser zu verstehen.
3. Bestimmt nun das zentrierte Fourierspektrum sowie das zentrierte Phasenspektrum der vorher generierten Fourier-Transformierten. Visualisiert beide Ergebnisse. Was seht ihr in den Spektren? Wie ist das Fourierspektrum zu interpretieren?
4. Führt nun folgende Veränderungen am zu transformierenden Bild durch und beobachtet die Veränderungen in den Spektren:
   - Generiert ein neues Bild, wobei Höhe und Breite des Rechtecks diesmal vertauscht sein sollen.
   - Generiert ein neues Bild, wobei Höhe und Breite des Rechtecks diesmal beide 20 betragen sollen.

   Habt ihr eine Erklärung für die Veränderungen?
5. Schreibt nun eine neue Funktion für das Erstellen von Bildern. Diesmal soll allerdings statt eines weißen Rechtecks ein weißer, gefüllter Kreis mit einem anzugebenden Radius erzeugt werden. Kreiert mit Hilfe dieser Funktion ein Bild, das einen Kreis mit Radius 10 etwa in der Bildmitte beinhaltet und berechnet bzw. visualisiert erneut das zentrierte Fourierspektrum sowie das zentrierte Phasenspektrum. Was für Effekte beobachtet ihr nun? Wie verändern sich die Spektren, wenn ihr das Bild mit dem weißen Kreis zunächst mit einem Gauß-Filter ($\sigma = 10$) weichzeichnet? Wie ist diese Veränderung zu erklären?

Zusatzaufgabe 5 — Filtern im Frequenzraum - 10 + 10 + 10 = 30 Punkte - Programmieraufgabe
Erlaubte (Sub-)Pakete: `numpy`, `skimage.io`, `matplotlib`
Für diese Aufgaben findet ihr im Moodle das Bild `mandrill_hBars.png` als Variante des bekannten Testbildes. Es unterscheidet sich vom bekannten Testbild durch periodische Bildstörungen in Form von dunklen horizontalen Streifen. In dieser Aufgabe sollen diese Bildstörungen im Frequenzraum entfernt werden.

1. Ladet zunächst das Bild `mandrill_hBars.png` in Python. Führt eine Fourier-Transformation auf dem Bild durch und visualisiert das zentrierte Fourierspektrum sowie das zentrierte Phasenspektrum. Sucht händisch im Fourierspektrum die vier Punkte, die etwa den periodischen Bildstörungen entsprechen und ermittelt händisch den ungefähren minimalen Abstand der Punkte zum Zentrum des dargestellten Fourierspektrums bei $(256, 256)$. Zieht nun 10 Einheiten von diesem kleinsten Abstand ab und setzt alle Punkte im Fourierspektrum auf 0, die einen Abstand von größer diesem Wert zum Mittelpunkt haben. Es bleibt also nur ein runder Bereich in der Mitte des Spektrums übrig, der Rest ist 0. Führt nun das manipulierte Fourierspektrum und das Phasenspektrum wieder zusammen. Achtet dabei darauf, dass ihr die Zentrierung der Spektren abschließend rückgängig machen müsst. Transformiert nun die so erhaltene Fourier-Transformierte zurück in den Ortsraum mit der inversen Fourier-Transformation. Visualisiert das Ergebnis. Wie sieht es aus? Welchen Filter habt ihr angewendet?
   Tipp: Nutzt **zum Anzeigen** des Fourierspektrums die Funktion `np.log()`, um das Spektrum zu skalieren.
2. Ersetzt nun diesen harten Unterschied, dass einige Teile des Fourierspektrums auf 0 gesetzt werden und andere Teile nicht verändert werden, durch eine weichere Lösung. Nutzt dazu die angegebene Funktion `gauss2d`, welche die Konstruktion einer zweidimensionalen Gaußfunktion mit dem Mittelpunkt in `mx,my` und der Standardabweichung `s` in beide Richtungen erlaubt.

```
def gauss2d(x, y, mx, my, s):
```

```
    return 1. / (2. * np.pi * s * s) * np.exp(-((x - mx)**2. / (2. * s**2.) + (y
     - my)**2. / (2. * s**2.)))
```

Ermittelt nun unter der Annahme, dass der Mittelpunkt der Gaußfunktion im Zentrum des dargestellten Fourierspektrums bei $(256, 256)$ liegt, für jedes Pixel ein Gewicht als Ergebnis der Gaußfunktion (bspw. mit $\sigma = 30$). Multipliziert nun die Einträge im Fourierspektrum mit den jeweiligen Gewichten, um ein neues manipuliertes Fourierspektrum zu erhalten. Erzeugt erneut die Fourier-Transformierte aus dem manipulierten Fourierspektrum sowie dem Phasenspektrum und transformiert diese zurück in den Ortsraum mit der inversen Fourier-Transformation. Visualisiert das Ergebnis. Was verändert sich? Welchen Filter habt ihr angewendet?

3. Entfernt jetzt nur noch kleine Bereiche um die vier Punkte im Fourierspektrum, die den Frequenzen der Bildstörung entsprechen. Ihr könnt dazu bspw. quadratische Regionen um diese Punkte mit der Größe 30×30 definieren und diese mit Nullen füllen. Was für Veränderungen sind nun im Ergebnis nach der inversen Fourier-Transformation zu erkennen?