3. Übung

Lineare Nachbarschaftsfilter

In dieser Übung werden Sie einen linearen Nachbarschaftsfilter auf einem Bild anwenden und die Rahmenbedingungen eines Filters kennenlernen. Wenden Sie die linearen Filter auf das Bild Lena.jpg an.

---

1. Lesen Sie das Kapitel 6 (*Filter*) aus dem Buch "Digitale Bildverarbeitung".
2. Implementieren Sie eine Funktion, die es erlaubt ein Bild (im sinnvollen Rahmen) mit frei wählbaren Filtermasken zu falten.
   - **Prototyp:** [out_image] = filter(in_image, filter, off)
     - **out_image:** Ergebnisbild (int) nach Faltung von in_image mit filter
     - **filter:** Filtermatrix (float)
     - **off:** Offset (int). Der Offset, auch bekannt als stride, gibt an, um wie viele Pixel die Filtermatrix verschoben wird. Bei einem offset > 1, wird die Bildgrösse deutlich verkleinert.
     - 8-Bit Graustufenbilder als Eingangs- und Ausgangsdaten.
     - Filtermatrix der Grösse (NxN) mit N = (2K + 1), K = 1, 2, ...
     - Beispielaufruf: filter(image, [1 1 1; 1 3 1; 1 1 1]/11, 1)
3. Erweitern Sie die Funktion *filter()* aus Aufgabe 2 um folgende Randbedingungen:
   - *min:* Setzt Bildpunkte auf den minimalen Wert (0). Dies wird auch als zero padding bezeichnet.
   - *max:* Setzt Bildpunkte auf den maximalen Wert (255)
   - *continue:* Setzt das Bild ausserhalb mit dem gleichen Pixelwert, wie das entsprechende am nächsten liegende Randpixel, fort.
   - Untersuchen Sie die Randbehandlungen auf ihr Verhalten bei Benutzung verschiedener Filter.

   - **Prototyp:** [out_image] = filter(in_image, filter, off, edge);
     - *edge:* Parameter zur Auswahl der Randbehandlung ('min') - String
4. Beantworten Sie folgende Fragen:
   a) Nennen Sie die Arten und Eigenschaften von linearen Filtern.
   b) Was ist der Unterschied zwischen linearen und nichtlinearen Filtern?

Nichtlineare Nachbarschaftsfilter

In dieser Übung werden Sie die Auswirkung nichtlinearer Filter auf ein mit *Salt and Pepper* verrauschtes Bild untersuchen.

---

1. Lesen Sie die Kapitel 6.4 (*Nichtlineare Filter*) und Kapitel 7 (*Kanten und Konturen*) aus dem Buch "Digitale Bildverarbeitung".
2. Implementieren Sie den Medianfilter.
   - **Prototyp:** out_image = medianFilter(in_image, filtersize, offset)
     out_image: Ergebnisbild nach Faltung von in_image mit filter
     in_image: Eingangbild (int); 8-Bit Graustufenbilder
     filtersize: Filtergrösse (int); Grösse (NxN) mit N = (2K + 1), K = 1, 2, ...
     offset: Offset (int)
     Beispielaufruf: medianFilter(image, 3, 1)
     Benutzung Sie für die Sortierung wenn möglich Heap Sort.
3. Beantworten Sie folgende Fragen:
   a) Vergleichen Sie die Ergebnisse der verschiedenen Filter miteinander und begrï¿½nden Sie diese.
   b) Warum ist es beim Medianfilter sinnvoll für die Sortierung Heap Sort zu verwenden?
   c) Untersuchen Sie, welche Effekte bei mehrmaligem Anwenden eines Filters auf das jeweilige Ergebnisbild auftreten.
   d) Welche Effekte treten bei grossen und bei kleinen Filtermasken auf?

Abgabe
Die Aufgaben werden per Git-Tag (https://git.ios.htwg-konstanz.de) bis jeweils zur kommenden Übungsstunde abgegeben. Zudem müssen die Lösungen in der nächsten Übungsstunde mündlich präsentiert werden. Es ist nicht nötig einen eigenen Branch pro Aufgabe zu erstellen.