4. Übung

Kantendetektion

In dieser Übung wird der Sobel-Operator zur Kantendetektion implementiert.

---

1. Lesen Sie das Kapitel 7 (*Kanten und Konturen*) aus dem Buch "Digitale Bildverarbeitung".
2. Beantworten Sie folgende Fragen:
   a) Was ist eine Kante und wie ist sie im Bild, in dessen Ableitung erkennbar?
   b) Was sind die **partiellen Ableitungen** eines Bildes? Was sagen sie aus?
   c) Was ist der **Gradientenvektor** (kurz: **Gradient**) eines Bildes?
   d) Wie kann die *Kantenstärke* berechnet werden?
   e) Wie kann die *lokale Kantenrichtung* berechnet werden?
3. Implementieren Sie den Sobel-Operator und wenden Sie Ihn auf das Bild *lena.jpg* an.
   a) Nutzen sie die Python-Datei *sobel.py* und definieren Sie die Filter in x und y Richtung. Vervollständigen Sie die Funktion *sobel()*, welche ihnen die erste Ableitung zurückgibt. Das Filterergebnis soll skaliert sein. Auf eine Randbehandlung kann verzichtet werden.
   b) Berechnen Sie die erste Ableitung in horizontaler Richtung und berechnen Sie die Ausführungszeit.
   c) Berechnen Sie die erste Ableitung in vertikaler Richtung und berechnen Sie die Ausführungszeit.
   d) Zusätzlich soll eine Funktion implementiert werden, die die *Kantenstärke* durch den Betrag des Gradienten berechnet.
4. Implementieren Sie den Sobel-Operator in C++. Hier lernen Sie kennen, wie Sie C++ Funktionen mithilfe von pybind11 aus ihrem Python Skript aufrufen können.
   a) Nutzen sie die Python-Datei *sobel_demo.py* und definieren Sie die Filter in x und y Richtung.
   b) Vervollständigen Sie die Funktion *sobel()* in der Datei *sobel_demo.cpp*, welche ihnen die erste Ableitung zurückgibt. Das Filterergebnis soll skaliert sein. Auf eine Randbehandlung kann verzichtet werden.
   c) Installieren Sie *python-dev* und *pybind11* in ihre Pythonumgebung.
   d) Kompilieren Sie unter Unix die *sobel_demo.cpp* mit folgendem Kommando:

```
c++ -I eigen-3.4.0 -O3 -Wall -shared -std=c++11 -fPIC
$(python3 -m pybind11 --includes) sobel_demo.cpp -o
sobel_demo$(python3-config --extension-suffix)
```

Windows-Nutzer benötigen hingegen CMake. Windows Nutzer, die dennoch Unix Kommandos ausführen möchten, empfehlen wir den *docker-anaconda* Container. Installieren Sie im Container die benötigten Conda-Pakete sowie die build-essential's. Ihr Verzeichnis können Sie mit folgendem Kommando einhängen:

```
docker run -it 4.Aufgabe:/home:rw <container name> bash
```

e) Berechnen Sie die erste Ableitung in horizontaler Richtung, indem sie in *sobel_demo.py* die *sobel()* Funktion aus *sobel_demo.cpp* aufrufen.

f) Berechnen Sie die erste Ableitung in vertikaler Richtung, indem sie in *sobel_demo.py* die *sobel()* Funktion aus *sobel_demo.cpp* aufrufen.

g) Messen Sie ebenfalls die Ausführungszeiten und vergleichen Sie mit Ihrer Python Implementierung.

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 nï¿½chsten Übungsstunde mündlich präsentiert werden. Es ist nicht nötig einen eigenen Branch pro Aufgabe zu erstellen.