Übung zur Vorlesung Digitale Bildverarbeitung
WiSe 2022/23

Blatt 2

Datentypen, Bild-Interaktionen

Erfordert die VL bis Kapitel	Image Statistics and Point Operations
Besprechung am	17. November 2022

Allgemeine Infos:

Wir empfehlen die Übungen in kleinen Teams durchzuführen. Das macht mehr Spaß und man lernt mehr.

Während des Moduls werden Sie verschiedene Bildverarbeitungsanwendungen implementieren. Diese Aufgaben sind in der Regel Stand-Alone Applikationen die nicht direkt aufeinander aufbauen. Achten Sie während der Bearbeitung auf möglichst aktuelle Software- und Paket-Versionen. Das Framework für die jeweils aktuelle Aufgabe finden Sie unter *Detaillierter Zeitplan und Downloads* auf der Kurswebseite unter `https://graphics.tu-bs.de/teaching/ws2223/DBCV`.

Die Lösung der Aufgabenblätter wird jeweils eine Woche nach Herausgabe, **Donnerstags 09:45 Uhr** in der Übung besprochen.

Die Aufgaben sind optional aber wir empfehlen dringend diese zu bearbeiten, da sonst die Gefahr besteht das Abschlussprojekt nicht erfolgreich bestehen zu können.

In diesem Übungsblatt werden wir eine Reihe von Hilfsfunktionen implementieren, die Ihnen später sehr nützlich sein können für die weiteren Aufgaben. Dafür haben wir eine weitere Datei *openCVTools.py* erzeugt, in welcher die Hilfsfunktionen definiert werden sollen. Mittels `import openCVTools as cvt` kann diese Datei importiert werden und die Funktionen im Namespace `cvt` verwendet werden. Mittels des Schlüsselwortes `as` können Bibliotheken in Python unter einem beliebigen Namen eingebunden werden, was den Code deutlich lesbarer machen kann.

Aufgabe 1 — Datentypen

> Ein Farbbild ist in (Python-)OpenCV eine 3D-Matrix: `img.shape = (Zeilen, Spalten, Farbkanäle)`
> Grau-/Schwarzweißbilder sind hingegen "nur" 2D-Matrizen: `img.shape = (Zeilen, Spalten)`
> Der Datentyp der einzelnen Pixelwerte darin ist standardmäßig `uint8` mit Werten im Bereich [0-255].

In der Datei *02_HelperFunctions.py* wird ein Bild in die Variable `img` geladen.

Erstellen Sie eine 1-Kanal Schwarzweiß(Grauton)-Kopie des Bildes.

Passen Sie die Funktionen `imageStats(..)` und `showImage(..)` in der Datei *openCVTools.py* so an, dass sie sowohl Grau- als auch Farbbilder korrekt anzeigen.

Erweitern Sie die Funktion `imageStats(...)`, so dass der Datentyp der Pixelwerte mit ausgegeben wird.

Konvertieren Sie das originale Bild von seinem Standarddatentyp (`uint8` pro Farbkanal) in `float64` und zeigen Sie es mittels der `showImage(...)`-Funktion an.

Voraussichtlich wird das Bild komplett weiß sein. Versuchen Sie herauszufinden woran das liegt und passen Sie das Bild entsprechend an.

Aufgabe 2 — Pixelzugriff

Zeichnen Sie ein rotes Rechteck mit 110×220 Pixeln genau in die Mitte des Bildes. Versuchen Sie dazu den Slicing Operator aus Python zu nutzen.

Aufgabe 3 — Mouse Callback

Es ist bereits sehr hilfreich, insbesondere beim Debuggen, dass Matplotlib die Pixelwerte unter dem Mauszeiger ausgibt. Trotzdem kann es für manche Anwendungen hilfreich sein den Pixelwert unter dem Mauszeiger im Programm selbst nutzen zu können.

Hängen Sie eine Callback-Funktion `mouse_move(event)` an das Bild im Fenster "Red box", die bei jeder Mausbewegung aufgerufen wird. Sie soll die Bild(!)-Koordinaten des Mauszeigers (x, y) und evtl. gedrückte Tasten ausgeben. Sie benötigen dazu u.U. die Referenz auf das Zeichenfeld des Fensters (vom Aufruf `plt.figure(title)`) aus der Funktion `showImage(..)`.

Optional: Entfernen Sie die Callback-Funktion, sobald auf das Bild geklickt wird.

Hilfreiche Informationen:

```
fig = plt.figure(TITLE)
...
fig.canvas.mpl_connect('EVENT', CALLBACK_FUNCTION)
```

Aufgabe 4 — Vorher / Nachher

Ebenso ist es sehr angenehm, wenn man mehrere Bilder gleichzeitig anzeigen kann, bspw. die Eingabe und die Ausgabe eines Algorithmus.

Schreiben Sie dazu eine Funktion `showImageList(...)`, die Bilder auch als Liste annimmt und diese horizontal nebeneinander anzeigt.

Optional: Deaktivieren Sie die Darstellung der X- und Y-Achsen. Erweitern Sie die Funktion, um die Anordnung der Bilder flexibler zu kontrollieren, z.B. nicht nur horizontal sondern in einem Gitter (`grid`), oder durch Anpassung der Abstände.

Aufgabe 5 — Contrast Stretching

Erweitern Sie die Funktionen `showImage(...)` und `showImageList(...)` um einen weiteren Parameter `normalize` vom Typ Boolean. Wenn dieser auf `False` gesetzt ist, dann soll die Funktion wie bisher funktionieren, wenn dieser auf `True` gesetzt wird, dann soll der Kontrast jedes Eingabebildes mittels Contrast Stretching verbessert werden, damit der gesamte Wertebereich ausgenutzt wird.

Optional: Skalieren Sie Parameter des Contrast Stretchings so, dass die 10% der hellsten Pixel auf den größtmöglichen Farbwert abgebildet werden und 10% der dunkelsten Pixel auf den dunkelsten Wert. Das ist hilfreich, damit Contrast Stretching trotz Ausreißern sinnvolle Ergebnisse liefern kann.

Nützliche Links

Die folgenden Links können Ihnen bei der Bearbeitung der Aufgaben behilflich sein.

- Die Google Suche `https://google.de`: Geben Sie dort einfach `python opencv <meinSurchwort>` ein und Sie werden sehr oft schnell fündig.
- OpenCV `https://opencv.org/`: Der Industriestandard für Bildverarbeitung.
- Matplotlib `https://matplotlib.org/`: Visualisierung mit Python.
- OpenCV Python Tutorial `https://docs.opencv.org/4.5.0/d6/d00/tutorial_py_root.html`