



Interactive Systems

Lineare Algebra und Computer Vision Basic

WiSe 2017/18

Ziel dieser Übungsaufgabe ist eine tiefere Einarbeitung in NumPy und OpenCV und Bildverarbeitung. Laden Sie den vorgegebenen Skelett-Sourcecode von Moodle und implementieren Sie die notwendigen Funktionen. Erweitern und verändern Sie den Sourcecode nach Belieben. Das ist nur eine anfängliche Unterstützung.

Wichtig: Für die erfolgreiche Bearbeitung der Aufgabe ist es unbedingt erforderlich, dass Sie sich selbst mit OpenCV, Python und Numpy vertraut: z.B. http://docs.opencv.org/3.1.0/d6/d00/tutorial_py_root.html, <https://www.stavros.io/tutorials/python/>) oder <https://docs.scipy.org/doc/numpy-dev/user/quickstart.html>. Datentypen und -formate sind bei der Benutzung von NumPy und OpenCV sehr wichtig. OpenCV Bilder werden häufig im *uint8*: 0 - 255 geladen, bearbeitet und gespeichert. Viele Berechnungen finden dann aber im *float* Format statt. Umrechnungen können u.a. mit NumPy *np.float32(image)* durchgeführt werden.

Aufgabe 1: Linear Algebra und Numpy Basics (4 Punkte)

Die Bibliothek *Numpy* ist sehr umfangreich und wir werden auch nicht jede Funktionalität brauchen, aber es ist wichtig mit den grundlegenden Datenstrukturen vertraut zu sein. Ich habe hier eine Liste von kleinen Aufgaben zusammengestellt, die Sie in der Datei *numpy_basics.py* beantworten sollen.

Suchen Sie sich insgesamt 15 Sterne zusammen und implementieren Sie diese. Das ist häufig ein Einzeiler oder Zweizeiler. Lesen Sie sich selbstständig in die Dokumentation ein und versuchen Sie die Fragen mit den notwendigen Numpy-Funktionen zu beantworten. Sollten Sie keine Funktion finden, implementieren Sie Lösung selbst in einer kleinen Funktion. Nutzen Sie dafür die vorgegebene Struktur der Datei. **Bitten geben Sie die Lösung jeweils auf der Console aus (mittels print).**

- a) (*) Erzeugen Sie einen Vektor mit Nullen der Länge 10 (10 Elemente) und setzen den Wert des 5. Elementes auf eine 1.
- b) (*) Erzeugen Sie einen Vektor mit Ganzahl-Werten von 10 bis 49 (geht in einer Zeile).
- c) (*) Drehen Sie die Werte des Vektors um (geht in einer Zeile).
- d) (*) Erzeugen Sie eine 4x4 Matrix mit den Werten 0 bis 15 (links oben rechts unten).
- e) (*) Erzeuge eine 8x8 Matrix mit Zufallswerte und finde deren Maximum und Minimum und normalisieren Sie die Werte (sodass alle Werte zwischen 0 und 1 liegen - ein Wert wird 1 (max) sein und einer 0 (min)).
- f) (*) Multiplizieren Sie eine 4x3 Matrix mit einer 3x2 Matrix
- g) (*) Erzeugen Sie ein 1D Array mit den Werten von 0 bis 20 und negieren Sie Werte zwischen 8 und 16 nachträglich.
- h) (*) Summieren Sie alle Werte in einem Array.
- i) (**) Erzeugen Sie eine 5x5 Matrix und geben Sie jeweils die geraden und die ungeraden Zeile aus.



Abbildung 1: 2.1. Laden des Bildes

- j) (**) Erzeugen Sie eine Matrix M der Größe 4×3 und einen Vektor v mit Länge 3. Multiplizieren Sie jeden Spalteneintrag aus v mit der kompletten Spalte aus M . Schauen Sie sich dafür an, was *Broadcasting* in Numpy bedeutet.
- k) (**) Erzeugen Sie einen Zufallsmatrix der Größe 10×2 , die Sie als Kartesische Koordinaten interpretieren können ($[[x_0, y_0], [x_1, y_1], [x_2, y_2]]$). Konvertieren Sie diese in Polarkoordinaten <https://de.wikipedia.org/wiki/Polarkoordinaten>.
- l) (***) Implementieren Sie zwei Funktionen, die das Skalarprodukt und die Vektorlänge **für Vektoren beliebiger Länge** berechnen. Nutzen Sie dabei NICHT die gegebenen Funktionen von NumPy. Testen Sie Ihre Funktionen mit den gegebenen Vektoren:

$$v_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}, v_2 = \begin{bmatrix} -1 \\ 9 \\ 5 \\ 3 \\ 1 \end{bmatrix}$$

- m) (***) Berechnen Sie $(v_0^T v_1) M v_0$ unter der Nutzung von NumPy Operationen. **Achten Sie darauf, dass hier v_0, v_1 Spaltenvektoren gegeben sind. v_0^T ist also ein Zeilenvektor.**

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 0 & 2 & 2 \end{bmatrix}, v_0 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, v_1 = \begin{bmatrix} -1 \\ 2 \\ 5 \end{bmatrix},$$

Aufgabe 2: Computer Vision Basics + OpenCV (6 Punkte)

2.1. (1 Punkt) Laden Sie das Bild *Lenna.png* unter Nutzung von OpenCV und zeigen es sowohl als Graustufenbild und als Farbbild nebeneinander an. Das Ergebnis sollte so aussehen, wie in Abbildung 1. Hinweis: *np.concatenate* kann mehrere Matrizen / Bilder miteinander verbinden. Dies können Sie auch hier benutzen, indem Sie ein neues Bild mit doppelter Breite anlegen. Das finale Bild muss dann aber 3 Farbkanäle haben, d.h. Sie müssen das Graustufenbild jeweils in die R/G/B Kanäle kopieren. Sie können die Bildgröße (shape) über *rows, cols = img.shape[:2]* bekommen.

2.2. (1 Punkt) Implementieren Sie in dem gegebenen Pythonscript die Möglichkeit keyEvents abzufangen und darauf zu reagieren. Implementieren Sie dann zusätzlich Funktionalität:

- Key 't' - das Verschieben beider Bilder als Translation in x-Richtung.
- Key 'r' - das Rotieren der Bilder um eine feste Gradzahl (z.B. 30 Grad wie in Abbildung 2 gezeigt).
- Key 'q' - zum Beenden der Demo.



Abbildung 2: 2.2. Verschieben und Rotieren des Bildes

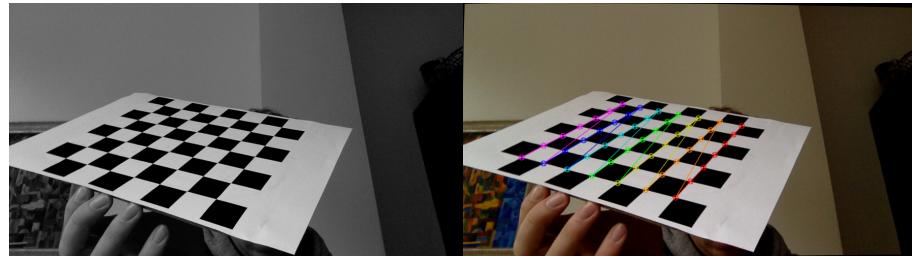


Abbildung 3: 2.4. Camera Calibration im Videostream.

Nutzen Sie für die Implementierung die OpenCV-Funktion `cv::warpAffine` http://docs.opencv.org/3.1.0/d54/group__imgproc__transform.html#ga0203d9ee5fc28d40dbc4a1ea4451983 und geben Sie die Transformationsmatrix M . Das Ergebnis sollte so aussehen wie in Abbildung 2.

2.3. (2 Punkt) Implementieren Sie die Faltung eines Bildes mit einer Filtermaske (Convolution) auf einem Graustufenbild. Nutzen Sie als Filtermaske ein Gaussianblur und die in der Vorlesung besprochenen Sobelfilter (x- und y-Richtung) zur Berechnung der Bildgradienten. Berechnen Sie daraus ein Bild, dass die Gradientstärken darstellt (Magnitude of Gradients). Hinweis: Ihre Implementierung kann durchaus langsam sein (mehrere Sekunden pro Bild) und das ist für die Aufgabe auch völlig in Ordnung. Die Implementierung von OpenCV läuft dann in Echtzeit, da Arrayzugriffe noch sehr stark optimiert werden können.

2.4. (1 Punkte) Experimentieren Sie mit folgenden - in der Vorlesung angesprochenen - Bildverarbeitungsalgorithmen in OpenCV:

- a) Wechsel von Farträumen (HSV, LAB, YUV)
- b) Adaptives Thresholding bitte in den beiden folgenden Varianten Gaussian-Thresholding und Otsu-Thresholding.
- c) Canny-Kantenextraktion

Nutzen Sie dafür die vorgegebene Datei `01_opencv_experiments.py` sodass man auf Tastendruck die oben genannten Funktionen wechseln kann.

2.5. (1 Punkt) Wie in der Vorlesung besprochen, ist eine Kalibrierung der Kamera für viele Anwendungen notwendig oder hilfreich. Diese Kalibrierung kann mittels des mitgelieferten Sourcecodes durchgeführt werden. Passen Sie den Sourcecode so an, dass Sie die Kalibrierung mit Ihrer Kamera durch einen Live-Videostream auf Tastendruck durchführen können. Schauen Sie sich dafür `cv2.VideoCapture` an. Nutzen Sie für die Kalibrierung dann das mitgelieferte Schachmusterpdf. Diese sollte auf einem ebenen Untergrund befestigt sein.

Abgabe Dies ist *Übungsaufgabe 1*; die Bearbeitungszeit der Aufgabe ist für ca. 2 Wochen ausgelegt. **Die Abgabe soll via Moodle zum angegebenen Zeitpunkt erfolgen.** Geben Sie bitte jeweils nur eine einzige .zip-Datei mit den Quellen Ihrer Lösung ab. Verspätete Abgaben werden mit einem Abschlag von 3 Punkten je angefangener Woche Verspätung belegt.

Demonstrieren und erläutern Sie Ihre Lösung in der nächsten Übung nach dem Abgabedatum. Die Qualität Ihrer Demonstration ist, neben dem abgegebenen Code, ausschlaggebend für die Bewertung. Viel Spaß und Erfolg bei der Bearbeitung.