

Bildverarbeitung in Matlab bzw. Python

Hinweise zur Nutzung des Computerraums

Im Computerraum C5-11 können Sie sich mit Ihrer LDAP-Kennung an einem PC Ihrer Wahl anmelden.

Die Unterlagen zur Übung und insbesondere die benötigten Beispielbilder können Sie entweder aus dem Moodle-Kurs herunterladen oder sich eine Kopie der Dateien vom Laufwerk $\square X:/$ BV (read only) holen.

Das Laufwerk $\square N:/$ (read/write) ist mit Ihrer Kennung benannt. Hier können Sie das Semester über Ihre Dateien und Projekte speichern.

Hinweise zu den Programmiersprachen und Software-Tools

Die Übungen und das Praktikum des Wahlfachs sind so gestaltet, dass Sie die Aufgaben zur Bildverarbeitung sowohl in Matlab als auch in Python bearbeiten können. Die Auswahl ist Ihnen freigestellt – es sind keine besonderen Vorerfahrungen nötig und in der Klausur wird kein Wissen abgefragt, das spezifisch für eine der Programmiersprachen wäre.



Matlab ist eine kommerzielle Software des US-amerikanischen Unternehmens MathWorks für die Lösung mathematischer Probleme. Es ist insbesondere für numerische Berechnungen mithilfe von Matrizen ausgelegt, woher sich auch der Name ableitet: MATrix LABoratory.

In Matlab wird mit einer proprietären, mathematisch orientierten Skriptsprache programmiert.

de.mathworks.com/help/matlab

Sie können Matlab auch auf Ihrem privaten Computer nutzen. Nähere Informationen zur Lizenz und eine Installationsanleitung finden Sie auf den Seiten der Campus-IT.

www.hochschule-bochum.de/cit/hard-software/uebersicht

Achten Sie bei der Installation darauf, dass Sie mindestens die im nächsten Abschnitt genannten Toolboxes auswählen!



Python ist eine weit verbreitete, freie und allgemein einsetzbare Programmiersprache mit klarer Syntax.

docs.python.org/3

Wenn Sie im Rahmen des Wahlfachs erstmals mit Python arbeiten und/oder noch keine einschlägige Erfahrung mit einer bestimmten Entwicklungsumgebung haben, empfehle ich Ihnen die minimalistische, freie Entwicklungsumgebung Thonny. Sie ist auch auf den PCs im Computerraum verfügbar.

thonny.org

Bei der Installation von Thonny auf Ihrem privaten Computer haben Sie den Vorteil, dass eine aktuelle Version von Python bereits enthalten ist.

Hinweise zu den verwendeten Bibliotheken



In Matlab gibt es diverse Funktionsbibliotheken (*Toolboxes*) zu verschiedenen Themenbereichen (z. B. Statistik, Signal- und Bildverarbeitung). Diese müssen bereits bei der Installation ausgewählt werden und können leider nicht ohne Umstände nachinstalliert werden! Die im Wahlfach verwendeten Funktionen stammen aus den folgenden Toolboxes:

- Image Acquisition Toolbox
- Image Processing Toolbox
- Medical Imaging Toolbox
- Symbolic Math Toolbox

Vorwiegend werden wir mit Funktionen aus der Image Processing Toolbox arbeiten.

de.mathworks.com/help/images

In Matlab-Skripts können die Funktionen aus installierten Toolboxes direkt verwendet werden. Ein expliziter Import o. ä. ist nicht nötig.



In Python stehen für viele Anwendungsbereiche (z. B. Statistik, Signal- und Bildverarbeitung) leistungsfähige Bibliotheken zur Verfügung. In Thonny können Sie diese über das Menü **Werkzeuge > Verwalte Pakete...** nachladen.

Im Rahmen des Wahlfachs verwenden wir insbesondere folgende Bibliotheken:

- NumPy (*Numerical Python*) mit Funktionen zum Umgang mit mehrdimensionalen Arrays bzw. Matrizen
- OpenCV (*Open Source Computer Vision Library*) mit Funktionen zur digitalen Bildverarbeitung
Paketname in Thonny: opencv-python
- Matplotlib mit Funktionen zur Erstellung verschiedener mathematischer Darstellungen
- scikit-image mit Funktionen zur digitalen Bildverarbeitung

numpy.org/doc/stable

docs.opencv.org

matplotlib.org/stable

scikit-image.org/docs/stable

In den nachfolgenden Anleitungen wird davon ausgegangen, dass diese Bibliotheken wie folgt importiert werden:

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import skimage as ski
```

Beachten Sie, dass Sie den Namen der Bibliothek als Präfix vor den Funktionsnamen setzen müssen. Beispielsweise nutzen Sie die Funktion `zeros()` aus der Bibliothek `numpy`, indem Sie `np.zeros()` aufrufen.

Des Weiteren werden wir einzelne Funktionen aus den Bibliotheken `os`, `pydicom`, `scipy` und `simpleitk`.

Aufgaben zu Lektion 1

Erste Schritte in Matlab bzw. Python

Aufgabe 1.1

Öffnen Sie Matlab und machen Sie sich mit der grafischen Benutzeroberfläche vertraut. Testen Sie insbesondere die Funktion des Command Window, indem Sie nacheinander die folgenden Anweisungen eintippen und jeweils mit Enter bestätigen:

```
x = 1;  
y = 1;  
z = x + y
```

Welche Funktion hat das Semikolon am Ende einer Anweisung? Welche Informationen können Sie dem Fenster „Workspace“ entnehmen?

Aufgabe 1.2

Matlab-Skripte werden mit der Dateiendung *.m gespeichert. Kommentarzeilen, die mit einem % beginnen, werden bei der Ausführung ignoriert. Mit doppelten Kommentarzeichen %% können Sie Ihr Skript in Abschnitte gliedern. Erstellen Sie ein neues, leeres Skript, fügen Sie die Anweisungen aus Aufgabe 1.1 ein und speichern Sie es in Ihrem persönlichen Arbeitsverzeichnis. Führen Sie das Skript dann aus.

Aufgabe 1.1

Öffnen Sie die Entwicklungsumgebung Thonny und machen Sie sich mit der grafischen Benutzeroberfläche vertraut. Testen Sie insbesondere die Funktion der Kommandozeile, indem Sie nacheinander die folgenden Anweisungen eintippen und jeweils mit Enter bestätigen:

```
x = 1  
y = 1  
z = x + y  
print(z)
```

Wofür benötigt man die Anweisung `print()`? Welche Informationen können Sie dem Fenster „Variablen“ entnehmen?

Aufgabe 1.2

Python-Skripte werden mit der Dateiendung *.py gespeichert. Kommentarzeilen, die mit einem # beginnen, werden bei der Ausführung ignoriert. Erstellen Sie ein neues, leeres Skript, fügen Sie die Anweisungen aus Aufgabe 1.1 ein und speichern Sie es in Ihrem persönlichen Arbeitsverzeichnis. Führen Sie das Skript dann aus.



Aufgabe 1.3

Oft ist es sinnvoll, am Anfang eines Skriptes einige Einstellungen vorzunehmen, die gewährleisten, dass das Skript fehlerfrei durchläuft.

Mit der Anweisung `clear` können Sie den Workspace und mit `clc` das Command Window leeren. Beides passiert beim Ausführen eines Matlab-Skripts nicht automatisch.

Sofern Sie nichts anderes angeben, liest und schreibt Matlab alle Dateien aus bzw. in das aktuelle Arbeitsverzeichnis. Welcher Order das ist, können Sie sich mit der Anweisung `pwd` ausgeben lassen. Mit der Anweisung `cd` können Sie das Arbeitsverzeichnis wechseln. Dabei kann der gewünschte Pfad fix als String (in Hochkommata) übergeben oder mithilfe der Anweisung `fullfile()` erzeugt werden.

Ergänzen Sie Ihr erstes Matlab-Skript mit den genannten Anweisungen. Was ist der Vorteil von `fullfile()` gegenüber der Angabe eines einzelnen Strings, um das Arbeitsverzeichnis anzugeben?



Aufgabe 1.3

Oft ist es sinnvoll, am Anfang eines Skriptes die benötigten Module zu laden und einige Einstellungen vorzunehmen, die gewährleisten, dass das Skript fehlerfrei durchläuft.

Um das OS-Modul zu laden, das Ihnen Zugriff auf einige Funktionen des Betriebssystems verschafft, müssen Sie zunächst die Anweisung `import os` ins Skript aufnehmen.

Sofern Sie nichts anderes angeben, liest und schreibt Python alle Dateien aus bzw. in das aktuelle Arbeitsverzeichnis. Welcher Order das ist, können Sie sich mit der Anweisung `os.getcwd` ausgeben lassen. Mit der Anweisung `os.chdir()` können Sie das Arbeitsverzeichnis wechseln. Der gewünschte Pfad kann fix als String (in Hochkommata) übergeben oder mithilfe der Anweisung `os.path.join()` erzeugt werden.

Ergänzen Sie Ihr erstes Python-Skript mit den genannten Anweisungen. Was ist der Vorteil von `os.path.join()` gegenüber der Angabe eines einzelnen Strings, um das Arbeitsverzeichnis anzugeben?

Aufgabe 1.4

Die Funktion `imread()` dient dazu, den Inhalt einer Bilddatei (z. B. `*.jpg`, `*.png`) als Matrix in den Matlab-Workspace zu laden.

Hat die eingelesene Bilddatei beispielsweise eine Speichertiefe von 8 Bit, so liegt nach dem Einlesen eine $(m \times n)$ -Matrix des Datentyps `uint8` vor.

Laden Sie das Beispielbild  `roentgen_thorax.jpg` in den Workspace. Wie groß ist die Bildmatrix und welche Speichertiefe hat das Bild? Nutzen Sie zur Beantwortung die angezeigten Informationen im Workspace-Fenster oder die Anweisung `whos`.

Aufgabe 1.5

Mit der Funktion `imshow()` können Sie sich die Bildinformationen, die Sie als Matrix in den Workspace geladen haben, auf dem Bildschirm anzeigen lassen. Das dabei erzeugte Fenster muss manuell geschlossen werden. Es kann daher hilfreich sein, die Anweisung `close` am Anfang eines jeden Skripts aufzurufen, um alle Fenster aus dem vorherigen Durchlauf zu schließen.

Lassen Sie sich das Beispielbild  `roentgen_thorax.jpg` ausgeben. Was sehen Sie auf dem Bild?

Aufgabe 1.4

Die Funktion `cv2.imread()` dient dazu, den Inhalt einer Bilddatei (z. B. `*.jpg`, `*.png`) als NumPy-Array zu laden. Sie sollten der Funktion als zweiten Parameter immer den Wert `cv2.IMREAD_UNCHANGED` übergeben, da sonst ggf. ungewollte Konvertierungen vorgenommen werden.

Hat die eingelesene Bilddatei beispielsweise eine Speichertiefe von 8 Bit, so liegt nach dem Einlesen ein NumPy-Array der Größe $(m \times n)$ und des Datentyps `uint8` vor.

Laden Sie das Beispielbild  `roentgen_thorax.jpg`. Wie groß ist die Bildmatrix und welche Speichertiefe hat das Bild? Nutzen Sie zur Beantwortung die folgenden Attribute des NumPy-Arrays:

```
I = cv2.imread("...", cv2.IMREAD_UNCHANGED)  
I.shape  
I.dtype
```

Aufgabe 1.5

Mit der Funktion `plt.imshow()` können Sie sich die Bildinformation, die Sie in ein NumPy-Array `I` geladen haben, auf dem Bildschirm anzeigen lassen. Gehen Sie dazu wie folgt vor:

```
plt.figure()  
plt.imshow(I, cmap='gray')  
plt.axis('off')  
plt.show()
```

Lassen Sie sich das Beispielbild  `roentgen_thorax.jpg` ausgeben. Was sehen Sie auf dem Bild?

Aufgabe 1.6

Mit der Funktion `imread()` können auch Farbbilder gelesen werden. Sie erkennen diese daran, dass nach dem Einlesen eine $(m \times n \times 3)$ -Matrix des Datentyps `uint8` vorliegt, da jeder Farbkanal in einer eigenen Matrix gespeichert wird. Mit der Funktion `rgb2gray()` können Sie das geladene Farbbild in ein Grauwertbild umwandeln.

Laden Sie das Beispielbild  `roentgen_kiefer.jpg` in den Workspace und wandeln Sie es in ein Grauwertbild um. Lassen Sie sich sowohl das Farb- als auch das Grauwertbild anzeigen.

Hinweis: Wenn Sie sich die beiden Bilder in verschiedenen Fenstern anzeigen lassen möchten, müssen Sie vor dem zweiten Aufruf von `imshow()` die Anweisung `figure` ausführen, damit Matlab ein weiteres Grafikfenster öffnet.

Aufgabe 1.7

Um eine im Workspace vorliegende Matrix als Bilddatei zu speichern, steht die Funktion `imwrite()` zur Verfügung.

Speichern Sie das in ein Grauwertbild umgewandelte Beispielbild  `roentgen_kiefer.jpg` unter  `roentgen_kiefer_grayscale.jpg` in Ihrem Arbeitsverzeichnis ab.

Matrizen



Bildinformationen werden in Matrizen gespeichert. Zur Umsetzung verschiedener Operationen der Bildverarbeitung werden wir daher auf die Möglichkeiten zurückgreifen, die Matlab bereitstellt, um mit Matrizen zu rechnen.

Aufgabe 1.6

Mit der Funktion `cv2.imread()` können auch Farbbilder gelesen werden. Sie erkennen diese daran, dass nach dem Einlesen ein NumPy-Array der Größe $(m \times n \times 3)$ und des Datentyps `uint8` vorliegt, da jeder Farbkanal in einer eigenen Matrix gespeichert wird. Mit der Funktion `cv2.cvtColor()` und dem Parameter `cv2.COLOR_BGR2GRAY` können Sie das geladene Farbbild in ein Grauwertbild umwandeln.

Laden Sie das Beispielbild  `roentgen_kiefer.jpg` und wandeln Sie es in ein Grauwertbild um. Lassen Sie sich sowohl das Farb- als auch das Grauwertbild anzeigen.

Aufgabe 1.7

Um Bildinformationen, die in Form eines NumPy-Arrays vorliegen, als Bilddatei zu speichern, steht die Funktion `cv2.imwrite()` zur Verfügung.

Speichern Sie das in ein Grauwertbild umgewandelte Beispielbild  `roentgen_kiefer.jpg` unter  `roentgen_kiefer_grayscale.jpg` in Ihrem Arbeitsverzeichnis ab.



Bildinformationen werden in NumPy-Arrays als Matrix gespeichert. Zur Umsetzung verschiedener Operationen der Bildverarbeitung werden wir daher auf die Möglichkeiten zurückgreifen, die NumPy bereitstellt, um mit NumPy-Arrays zu rechnen.



Aufgabe 1.8

Um eine Matrix zu erzeugen und in einer Variablen zu speichern, werden die Matrixelemente zeilenweise zwischen eckigen Klammern eingegeben, wobei ein Leerzeichen die Spalten trennt und mit einem Semikolon eine neue Zeile begonnen wird:

```
A = [1 2 3; 4 5 6; 7 8 9]
```

Erzeugen Sie die angegebene Beispielmatrix A. Welchen Datentyp hat die erzeugte Variable? Wandeln Sie die Matrix mit der Funktion `uint8()` in eine 8 Bit-Bildmatrix um.



Aufgabe 1.9

Testen und beschreiben Sie, welche Arten von Matrizen mit den Funktionen `eye()`, `zeros()`, `ones()` und `rand()` erzeugt werden können.

Hinweis: Bei der Angabe der Dimension einer Matrix gilt in Matlab, wie in der Mathematik üblich: „Zeilen zuerst, Spalten später“.



Aufgabe 1.8

Um ein NumPy-Array zu erzeugen und in einer Variablen zu speichern, werden die Elemente zeilenweise zwischen eckigen Klammern eingegeben:

```
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

Erzeugen Sie das angegebene Beispielarray A. Welchen Datentyp hat die erzeugte Variable? Wandeln Sie das Array mit der Anweisung `A.astype(np.uint8)` in eine 8 Bit-Bildmatrix um.



Aufgabe 1.9

Testen und beschreiben Sie, welche Arten von Arrays mit den Funktionen `np.eye()`, `np.zeros()`, `np.ones()` und `np.random.rand()` erzeugt werden können.

Hinweis: Bei der Angabe der Dimension eines Arrays gilt in NumPy, wie in der Mathematik üblich: „Zeilen zuerst, Spalten später“.



Aufgabe 1.10

Testen Sie die folgenden Anweisungen zur Selektion bzw. zum Überschreiben einzelner Elemente, Zeilen, Spalten und Teilmatrizen. Notieren Sie jeweils, was sie bewirken.

```
A = [1 2 3; 4 5 6; 7 8 9];  
A(2, 3)  
A(3, :)  
A(:, 1)  
A(1:2, 1:2)  
A(1, 1) = 9  
A(2:3, 2:3) = 1  
A(:, 3) = []  
A(:, 3) = [1;2;3]
```

Hinweise: Matlab beginnt die Zeilen- und Spaltennummerierung innerhalb einer Matrix mit dem Index 1. Links und rechts vom Doppelpunktoperator stehen die erste und die letzte Zahl einer Zahlenfolge (z. B. steht `2:5` für 2, 3, 4, 5).

Wenden Sie nun die Methoden zur Matrixselektion auf die Bildmatrix von Beispielbild [roentgen_thorax.jpg](#) an, so dass nur noch ein Lungenflügel zu sehen ist.



Aufgabe 1.10

Testen Sie die folgenden Anweisungen zur Selektion bzw. zum Überschreiben einzelner Elemente, Zeilen, Spalten und Teilmatrizen. Notieren Sie jeweils, was sie bewirken.

```
A = np.array([[1,2,3], [4,5,6], [7,8,9]])  
A[1, 2]  
A[2, :]  
A[:, 0]  
A[0:2, 0:2]  
A[0, 0] = 9  
A[1:3, 1:3] = 1  
A = np.delete(A, 2, axis=1)  
A = np.insert(A, 2, [1,2,3], axis=1)
```

Hinweise: NumPy beginnt die Zeilen- und Spaltennummerierung innerhalb eines Arrays mit dem Index 0. Links vom Doppelpunktoperator steht die erste Zahl einer Zahlenfolge; rechts steht die erste Zahl, die nicht mehr zur Zahlenfolge gehört (z. B. steht `2:5` für 2, 3, 4).

Wenden Sie nun die Methoden zur Matrixselektion auf das NumPy-Array von Beispielbild [roentgen_thorax.jpg](#) an, so dass nur noch ein Lungenflügel zu sehen ist.



Aufgabe 1.11

In Matlab können arithmetische Operationen wie $+$, $-$, \cdot und $/$ auch mit Matrizen durchgeführt werden. Beachten Sie jedoch, dass das Multiplizieren zweier Matrizen A und B – also `A*B` – eine Matrixmultiplikation zur Folge hat. Wollen Sie jeweils die einzelnen Elemente der Matrizen paarweise miteinander multiplizieren, müssen Sie die Punktsyntax `A.*B` nutzen.

Berechnen Sie aus Matrix A, die wie in Aufgabe 1.8 definiert ist, eine Matrix B, deren Elemente...

- a) jeweils um 2 höher sind als die Elemente in A.
- b) jeweils doppelt so hoch sind wie die Elemente in A.
- c) jeweils die Quadrate der Elemente aus A sind.

Erstellen Sie eine (3×3) -Einheitsmatrix (vgl. Aufg. 1.9) und maskieren Sie damit Matrix A, so dass die Diagonalelemente von A erhalten bleiben, alle anderen Elemente jedoch 0 werden.



Aufgabe 1.12

Die Dimensionen einer Matrix in Matlab können mit den Funktionen `ndims()` und `size()` ermittelt werden.

Schreiben Sie ein Matlab-Skript, das das Beispielbild [mri_colored.jpg](#) einliest, überprüft, ob es sich um ein Farbbild handelt, das Farbbild ggf. in ein Grauwertbild konvertiert und schließlich im Command Window die Anzahl der Zeilen und Spalten sowie die Größe der Bildmatrix ausgibt. Machen Sie sich dazu mit der Syntax der `if`-Abfrage und der Funktion `disp()` vertraut.



Aufgabe 1.11

In NumPy können arithmetische Operationen wie $+$, $-$, \cdot und $/$ auch mit Arrays durchgeführt werden. Beachten Sie dabei, dass das Multiplizieren zweier Arrays zur Folge hat, dass die einzelnen Elemente der Arrays paarweise miteinander multipliziert werden (d. h. es ist *keine* Matrixmultiplikation im mathematischen Sinne).

Berechnen Sie aus Matrix A, die wie in Aufgabe 1.8 definiert ist, eine Matrix B, deren Elemente...

- a) jeweils um 2 höher sind als die Elemente in A.
- b) jeweils doppelt so hoch sind wie die Elemente in A.
- c) jeweils die Quadrate der Elemente aus A sind.

Erstellen Sie eine (3×3) -Einheitsmatrix (vgl. Aufg. 1.9) und maskieren Sie damit Matrix A, so dass die Diagonalelemente von A erhalten bleiben, alle anderen Elemente jedoch 0 werden.



Aufgabe 1.12

Die Dimension eines Arrays A in NumPy kann mit der Anweisung `A.ndim` ermittelt werden.

Schreiben Sie ein Python-Skript, das das Beispielbild [mri_colored.jpg](#) einliest, überprüft, ob es sich um ein Farbbild handelt, das Farbbild ggf. in ein Grauwertbild konvertiert und schließlich in der Kommandozeile die Anzahl der Zeilen und Spalten sowie die Größe der Bildmatrix ausgibt. Machen Sie sich dazu mit der Syntax der `if`-Abfrage und der Funktion `print()` vertraut.