Zurich University
of Applied Sciences

HS 2023
Digital Image Processing
M. Loeser und F. Wadehn

# Lab 1 – Image Processing Basics

The purpose of this lab is to familiarize you with the basic image processing functions in Python. In particular you will learn how to load and display images using the `matplotlib`, `scikit-image` and other libraries.

## Learning Objectives

- You know how to load images with various file formats (dcm, tif, jpg, gif, png, bmp) and store them as arrays.
- You know how to convert between different data types (`uint8` and `float`).
- You know how the data of color images are organized.
- You can select, process and reassemble 2D and 3D image regions.
- You know how to change the resolution of images and discuss the resulting effects.

## 1 Tasks

### 1.1 Grayscale Images

1. Make sure to have the Python-packages `numpy` and `matplotlib` installed. Familiarize yourself with installing packages using `pip` or `conda`.
2. Load the color image `lena_grayscale.gif`. There are various ways to do that, e.g., using matplotlib (deprecated), scikit-image, or PIL (Python Imaging Library). Try different libraries and look at the return type of the function that reads in the image.
3. Determine the image size and the number of color channels.
4. Display the image. Again, there are several ways to do that. One option is employing `matplotlib.pyplot.imshow(...)`.
5. Save the image in different file formats such as .tiff, .png and .bmp file.

## 1.2 Color Images

1. Load the color image `lena_color.gif`.
2. Display the entire image and all three color channels (R,G,B).
3. Convert the image to grayscale.

## 1.3 Medical Images—DICOM File Format

### 1.3.1 Input, Output, Image Information and Data Types

In medical technology, the DICOM format is often used to store images. In this way, additional meta – information such as the patient's name, information about how the image was acquired, etc. can be stored.

1. Install the package `pydicom`.
2. Use the command `pydicom.dcmread(...)` to read the given DICOM-files. Which additional metadata are contained in the DICOM file `brain_001.dcm`?
3. Display the corresponding image.
4. Try to flip the image both horizontally and vertically.
5. Convert the image to `uint8`. Display the resulting image `g`. Was the conversion correct? If not—what might be the issue?
6. This time, prior to again converting the image `f` to `uint8`, modify the image in such way that all pixel values are in the range $[0, 255]$. Then convert the original image and display the result.
7. Convert the image `g` from `unit8` to a floating point representation and display the result.

### 1.3.2 Image Sequences

The directory *brain* contains 20 sectional images (slices) that were obtained from a CT scan of a human head. The slices begin at the chin (slice No. 1), end at the forehead and are in parallel to the transversal planes (see Fig. 1). This results in a 3d representation of the head—with high spatial resolution in two dimensions and poor spatial solution in vertical direction. For storing the images into a 3D array you can use the `np.stack(...)`. The result is a 3-D array, where the third index corresponds to the number of the image. Note: the patient's nose is pointing upwards.

1. Use a for-loop to display all individual images.
2. Save the image sequence as a .gif file (hint: use `Image.save` from the PIL library).
3. Display all the single images in a combined image.

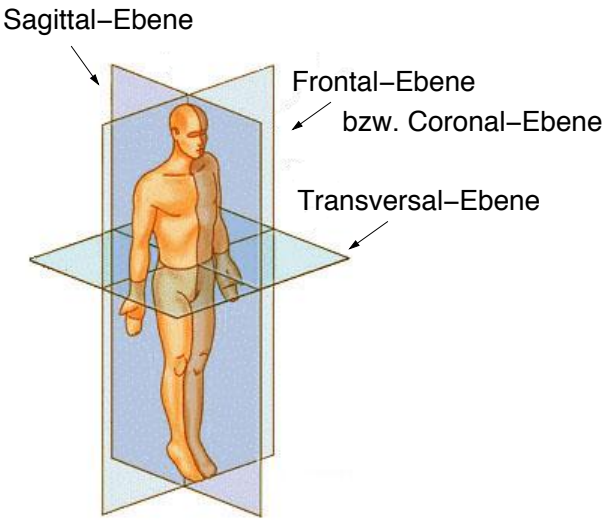


Figure 1: Different imaging planes.

4. Create an image parallel to the frontal plane (also called the coronal plane) through the center of the image. The image dimensions can be obtained for instance with numpy using the `[s, M, N] = image.shape` ($s$ number of slices, M in $x$-direction and N in $y$-direction) command.
   Note 1: Make sure to create a frontal view. To flip an image you can use the command `np.flip(...)`. Note 2: The distance between image slices is significantly larger than between pixels in the imaging plane. Resize or rescale the image (via interpolation) to obtain a square image.

5. Create an image parallel to the sagittal plane through the center.

6. **Optional:** Generate a series of images parallel to the sagittal plane and create an animation to view them all.