Image Analysis

Exercise 1

Introduction to Image Analysis using Python

Fall 2022

Introduction

This exercise introduces image analysis in Python.

Learning Objectives

After completing this exercise, the student should be able to do the following:

1. Install and use the `Anaconda` framework.
2. Create and activate a `conda virtual environment`.
3. Install Python packages in a virtual environment.
4. Import Python packages.
5. Read an image
6. Extract information about the dimensions of the image and the pixel type.
7. Display an image using both grey level scaling and using color maps.
8. Display an image histogram.
9. Extract individual bin counts and the bin edges from an image histogram.
10. Describe the (row, col) coordinate system used in scikit-image
11. Inspect pixel values in an image using (row, column) pixel coordinates.
12. Use NumPy slicing to extract and change pixel values in an image.
13. Compute a binary mask image based on an input image.
14. Use a binary mask image to extract and change pixel values in an image.

15. Inspect RGB values in a color image.
16. Extract and change RGB values in a color image.
17. Transfer images from a camera or a mobile phone to the computer so it can be used in Python image analysis scripts.
18. Rescale an image, where the width and height of the image are scaled with the same factor.
19. Resize and image, where the width and height of the image are scaled with the different factors.
20. Transform an RGB image into a grey-level image (rgb2gray).
21. Transform a gray-level image into an RGB image (gray2rgb).
22. Transform a pixel representation from floating points to unsigned bytes.
23. Visualize individual color channels in an RGB image.
24. Change and manipulate individual color channels in an RGB image.
25. Sample and visualize grey scale profiles from gray scale images.
26. Create 3D visualizations of an image, so it is seen as a height map.
27. Read individual DICOM files and get information about the image from the DICOM header.
28. Access the raw pixel data of individual DICOM files.
29. Visualize individual DICOM files and select appropriate gray value mapping limits to enhance contrast.

Anaconda and virtual environments

We strongly recommend to use the `Anaconda` framework for your Python installation and other tools. It can be installed from here `https://www.anaconda.com/distribution/`.

When you have installed Anaconda, you can start by creating a *virtual environment*. A virtual environment is a practical way to keep a specific Python interpreter, installed libraries and scrips isolated. So if you have several projects that each requires different libraries, they can be kept separate.

Start an `Anaconda prompt` and do:

```
conda create --name course02502 python=3.9
activate course02502
```

here a new virtual environment called `course02502` is created and activated.

Python scripts or Notebooks

In this course, there are basically two ways that you can do the exercises:

1. **Jupyter notebooks** is a web-based interactive platform, where you can run Python code. We recommend using `JupyterLab` that can be found trough the Anaconda framework. Please see Appendix B for details on figures in notebooks.
2. **Python scripts** are text files with the `.py` ending (for example `myscript.py`). Python scripts are normally edited and executed using a dedicated code editor (IDE). We recommend `Spyder` or `PyCharm`.

We provide the exercise code as pieces of Python code and it is your decision if you want to use them in a Notebook or in a script.

If you need an introduction or a refresher to Jupyter notebooks, Python, NumPY and Matplotlib, you can find a good introduction at: `https://github.com/nabriis/Jupyter_Python_Introduction`.

It is important that your Notebook, Spyder, Pycharm or other development tool is set to use the virtual environment that you created before.

Installing Python packages

In this course, we will use several Python packages. We will install them as needed. To install the package *scikit-image* (skimage)[1], start an `Anaconda prompt` and do:

```
activate course02502
conda install -c anaconda scikit-image
```

(see here `https://anaconda.org/anaconda/scikit-image` for further info).

More info on installing packages can be found in Appendix A.

Exercise data and material

The data and material needed for this exercise can be downloaded from

`http://courses.compute.dtu.dk/02502/`.

Start by creating an exercise folder where you keep your data, Python scripts or Notebooks. Download the data and material and place them in this folder.

[1] `https://scikit-image.org/`

Basic image handling

In this exercise, we will read images from the disk, display them and make some basic examinations of them.

Exercise 1

Start by importing some relevant libraries:

```
from skimage import color, io, measure, img_as_ubyte
from skimage.measure import profile_line
from skimage.transform import rescale, resize
import matplotlib.pyplot as plt
import numpy as np
import pydicom as dicom
```

Exercise 2

One image is a part of an X-ray image of a hand. X-ray examinations are extremely common and are used for example for bone fracture detection.

Start by reading the image:

```
# Directory containing data and images
in_dir = "data/"

# X-ray image
im_name = "metacarpals.png"

# Read the image.
# Here the directory and the image name is concatenated
# by "+" to give the full path to the image.
im_org = io.imread(in_dir + im_name)
```

then check the image dimensions by writing:

```
print(im_org.shape)
```

To check what type (unsigned int, boolean, double or something else) the pixels are, use:

```
print(im_org.dtype)
```

Exercise 3

Display the image by writing:

```
io.imshow(im_org)
plt.title('Metacarpal image')
io.show()
```

Try to use the simple viewer tools like *zoom* tool to inspect the finger bones. You can see the pixel values at a given pixel position (in x, y coordinates) in the upper right corner. Where do you see the highest and lowest pixel values?

Color Maps

Exercise 4

When working with gray level images, they are viewed using 256 levels of gray. There is another method of viewing them using a *color map* where each gray level is shown as a color. `Matplotlib` has several predefined color maps. Try

```
io.imshow(im_org, cmap="jet")
plt.title('Metacarpal image (with colormap)')
io.show()
```

Experiment with different colormaps. For example `cool, hot, pink, copper, coolwarm, cubehelix`, and `terrain`. A list of color maps can be found here: `https://matplotlib.org/stable/tutorials/colors/colormaps.html`.

Grey scale scaling

Exercise 5

Sometimes, there is a lack of contrast in an image. It possible to scale the way the image is visualized, by forcing a pixel value range into the `[all black, all white gray]` range.

By calling `imshow` like this:

```
io.imshow(im_org, vmin=20, vmax=170)
plt.title('Metacarpal image (with gray level scaling)')
io.show()
```

Pixels with values of 20 and below will be visualized black and pixels with values of 170 and above as white.

Exercise 6

Try to find a way to automatically scale the visualization, so the pixel with the lowest value in the image is shown as black and the pixel with the highest value in the image is shown as white.

Histogram functions

Computing and visualizing the image histogram is a very important tool to get an idea of the quality of an image.

Exercise 7

Compute and visualise the histogram of the image:

```
plt.hist(im_org.ravel(), bins=256)
plt.title('Image histogram')
io.show()
```

Since the histogram functions takes 1D arrays as input, the function `ravel` is called to convert the image into a 1D array.

The bin values of the histogram can also be stored by writing:

```
h = plt.hist(im_org.ravel(), bins=256)
```

The value of a given bin can be found by:

```
bin_no = 100
count = h[0][bin_no]
print(f"There are {count} pixel values in bin {bin_no}")
```

Here `h` is a list of tuples, where in each tuple the first element is the bin count and the second is the bin edge. So the bin edges can for example be found by:

```
bin_left = h[1][bin_no]
bin_right = h[1][bin_no + 1]
print(f"Bin edges: {bin_left} to {bin_right}")
```

Here is an alternative way of calling the histogram function:

```
y, x, _ = plt.hist(im_org.ravel(), bins=256)
```

Exercise 8

Use the histogram function to find the most common range of intensities? (hint: you can use the list functions `max` and `argmax`).

Pixel values and image coordinate systems

We are using `scikit-image` and the image is represented using a `NumPy` array. Therefore, a two-dimensional image is indexed by rows and columns (abbreviated to `(row, col)` or `(r, c)`) with `(0, 0)` at the top-left corner.

Exercise 9

The value of a pixel can be examined by:

```
r = 100
c = 50
im_val = im_org[r, c]
print(f"The pixel value at (r,c) = ({r}, {c}) is: {im_val}")
```

where r and c are the row and column of the pixel. What is the pixel value at (r, c) = (110, 90) ?

Exercise 10

Since the image is represented as a `NumPy` array, the usual *slicing* operations can be used. What does this operation do?

```
im_org[:30] = 0
io.imshow(im_org)
io.show()
```

Exercise 11

A *mask* is a binary image of the same size as the original image, where the values are either 0 or 1. Here

```
mask = im_org > 150
io.imshow(mask)
io.show()
```

a mask is created from the original image. Where are the values 1 and where are they 0?

What does this piece of code do?

```
im_org[mask] = 255
io.imshow(im_org)
io.show()
```

Color images

In a color image, each pixel is defined using three values: R (red), G (green), and B (blue).

Exercise 12

An example image `ardeche.jpg` is provided.

Read the image and print the image dimensions and its pixel type. How many rows and columns do the image have?

Exercise 13

What is the (R, G, B) pixel values at (r, c) = (110, 90)?

A pixel can be assigned a (R, G, B) by for example:

```
r = 110
c = 90
im_org[r, c] = [255, 0, 0]
```

Try to use NumPy *slicing* to color the upper half of the photo green. The number of rows in the image can be found using `rows = im_org.shape[0]` . Remember to *cast* your computed height into `int` before using it. For example `r_2 = int(rows / 2)` or use the *division floor operator*: `r_2 = rows // 2` .

Working with your own image

It is now time to work with one of your own images. It is assumed that you know how to either save an image from a digital camera on the disk or download an image. Copy the image to your relevant Python folder.

Exercise 14

Start by reading the image and examine the size of it. We can rescale the image, so it becomes smaller and easier to work with:

```
image_rescaled = rescale(im_org, 0.25, anti_aliasing=True,
                         channel_axis=2)
```

Here we selected a scale factor of 0.25. We also specify, that we have more than one channel (since it is RGB) and that the channels are kept in the third dimension of the NumPy array. The rescale function has this side effect, that it changes the type of the pixel values. What is the new type of the pixel? Try to show the image and inspect the pixel values. Are they still in the range of $[0, 255]$?

The function `rescale` scales the height and the width of the image with the same factor. The `resize` functions can scale the height and width of the image with different scales. For example:

```
image_resized = resize(im_org, (im_org.shape[0] // 4,
                        im_org.shape[1] // 6),
                        anti_aliasing=True)
```

Exercise 15

Try to find a way to automatically scale your image so the resulting width (number of columns) is always equal to 400, no matter the size of the input image?

Exercise 16

To be able to work with the image, it can be transformed into a gray-level image:

```
im_gray = color.rgb2gray(im_org)
im_byte = img_as_ubyte(im_gray)
```

We are forcing the pixel type back into *unsigned bytes* using the `img_as_ubyte` function, since the `rgb2gray` functions returns the pixel type as floating point numbers.

Exercise 17

Compute and show the histogram of you own image.

Exercise 18

Take an image that is very dark and another very light image. Compute and visualise the histograms for the two images. Explain the difference between the two.

Color channels

We are now going to look at the intensity values of the different channels of a color (RGB) image taken at DTU.

Exercise 19

Start by reading and showing the `DTUSign1.jpg` image.

You can visualise the red (R) component of the image using:

```
r_comp = im_org[:, :, 0]
io.imshow(r_comp)
plt.title('DTU sign image (Red)')
io.show()
```

Do this with the R, G, and B components.

Why does the DTU Compute sign look bright on the R channel image and dark on the G and B channels? Why do the walls of the building look bright in all channels?

Simple Image Manipulations

Exercise 20

Start by reading and showing the `DTUSign1.jpg` image. Now create a black rectangle in the image, by setting all RGB channels to zero. This is also an example of using NumPy slicing:

```
im_org[500:1000, 800:1500, :] = 0
```

Show the image again and save it to disk as `DTUSign1-marked.jpg` using the `io.imsave` function.

Try to save the image using different image formats like for example PNG.

Exercise 21

Try to create a blue rectangle around the DTU Compute sign and save the resulting image.

Exercise 22

Try to automatically create an image based on `metacarpals.png` where the bones are colored blue. You should use `color.gray2rgb` and pixel masks.

Advanced Image Visualisation

Before implementing a fancy image analysis algorithm, it is very important to get an intuitive understanding of how the image *looks as seen from the computer*. The next set of tools can help to gain a better understanding.

Exercise 23

In this example we will work with an x-ray image of the human hand. Bones are hollow and we want to understand how a hollow structure looks on an image. Start by reading the image `metarcarpals.png`. To investigate the properties of the hollow bone a grey-level profile can be sampled across the bone. Use the tool `profile_line` to sample profile across the bone:

```
p = profile_line(im_org, (342, 77), (320, 160))
plt.plot(p)
plt.ylabel('Intensity')
plt.xlabel('Distance along line')
plt.show()
```

What do you see - can you recognise the inner and outer borders of the bone-shell in the profile?

Exercise 24

An image can also be viewed as a landscape, where the height is equal to the grey level:

```
in_dir = "data/"
im_name = "road.png"
im_org = io.imread(in_dir + im_name)
im_gray = color.rgb2gray(im_org)
ll = 200
im_crop = im_gray[40:40 + ll, 150:150 + ll]
xx, yy = np.mgrid[0:im_crop.shape[0], 0:im_crop.shape[1]]
fig, ax = plt.subplots(subplot_kw={"projection": "3d"})
surf = ax.plot_surface(xx, yy, im_crop, rstride=1, cstride=1, cmap=plt.cm.jet,
                       linewidth=0)
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.show()
```

Use the mouse to rotate the view and find a good viewpoint. Notice how the white road markings are clearly visible on the 3D view.

DICOM images

Exercise 25

Typical images from the hospital are stored in the DICOM format. An example image from a computed tomography examination of abdominal area is used in the following.

Start by examining the header information using:

```
in_dir = "data/"
im_name = "1-442.dcm"
ds = dicom.dcmread(in_dir + im_name)
print(ds)
```

What is the size (number of rows and columns) of the DICOM slice.

This image has been *anonymized* so patient information has been removed. Else the patients name and diagnosis is sometimes also available. This makes medical images very complicated to share due to the need of protecting patient privacy.

Exercise 26

We can get access to the pixel values of the DICOM slice by:

```
im = ds.pixel_array
```

Try to find the shape of this image and the pixel type? Does the shape match the size of the image found by inspecting the image header information?

Exercise 27

We can visualize the slice using:

```
io.imshow(im, vmin=-1000, vmax=1000, cmap='gray')
io.show()
```

As can be seen, the pixel values are stored as 16 bit integers and therefore it is necessary to specify which value range that should be mapped to the gray scale spectrum (using vmin and vmax). Try to experiment with the vmin and vmax values to get the best possible contrast in the image.

A Additional package installations

matplotlib (Further info: `https://anaconda.org/conda-forge/matplotlib`)

```
activate course02502
conda install -c conda-forge matplotlib
```

pydicom (Further info: `https://anaconda.org/conda-forge/pydicom`)

```
activate course02502
conda install -c conda-forge pydicom
```

B Visualization in Jupyter Notebook and JupyterLab

It is not always that interactive visualizations work out-of-the-box in for example in Jupyter Notebook or in JupyterLab. Some additional experimentation might be needed.

It might be necessary to install `ipympl`:

```
conda install -c conda-forge ipympl
```

You might also need to change the *visualization backend* by adding this in a cell in a notebook:

For Jupyter Notebook:

```
%matplotlib nbgg
```

And in JupyterLab either:

```
%matplotlib widget
```

or

```
%matplotlib inline
```