Exercise 1

Installation and Setup Guide

IAML 2020

The purpose of this exercise is to guide you through the process of preparing the environment needed during the course. We use **Anaconda** for managing packages and environments. It makes it easy to use multiple versions of *Python* and packages on a single machine.

Notes

- The OpenCV docs are sadly only available for `C++` in their complete form. Therefore, we often refer to a number of well-written guides on using OpenCV with Python instead of the API docs. You can find the API docs as well as the guides at `https://docs.opencv.org/4.1.1/`. Additionally, we suggest you use Python's `help` function for OpenCV functions.

Exercise 1.1

Installation

Go to the anaconda webpage[1] and download the installer for *Python 3.7*. We use the *Python 3.7* version because it is the one being actively developed and maintained. **Windows** and **Mac OS** users get simple graphical installers with on-screen instructions.[2]. **Linux** users need to manually install a few libraries and use the terminal for installing Anaconda itself. Detailed installation guides can be found for all platforms (but is especially useful for Linux users) here[3].

Exercise 1.2

Prepare the Python environment

1. On Mac OS and Linux, open a terminal. On Windows, open the *Anaconda Prompt* application.

2. Navigate to the code folder of the downloaded exercise files using `cd <path to code>`.

3. Execute the following command

   ```
   conda env create -f iaml_env.yml
   ```

   This should create a new environment named *iaml* and automatically install the necessary packages used during the course, including `OpenCV`, `Numpy`, `Pytorch`, etc.

[1] `https://www.anaconda.com/distribution/`

[2] Windows users should check the *use Anaconda as default Python* checkbox when prompted.

[3] `https://docs.anaconda.com/anaconda/install/`

Using Anaconda

Anaconda enables easy switching between environments that may contain different versions of Python and packages. Although we only use a single environment for the course, you may want to create custom environments when developing your own projects. This ensures that you do not accidentally invalidate the code for other projects by, for example, updating a package with breaking API changes.

Environments need to be activated to be used. The command for activation is

```
conda activate <env-name>
```

When an environment is active, its specified Python version is available using the `python` terminal command. We have listed a number of commonly used Anaconda commands here:

The name of the active environment is shown in the terminal to the left of the usual prompt.

conda list Lists installed packages and versions in the current environment.

conda install <package> Install the specified package using Anaconda.

conda uninstall <package> Remove the specified package.

conda env create -n <name> python=<version> Create new environment with specified name and python version.

conda update <package/–all> Update specific package or all package in currently active environment.

Additional information can be found using `conda --help`, `conda <command> --help`, or the official Anaconda documentation[4].

Exercise 1.3
Setting default environment

Having to activate the same environment every time you open a terminal might get tiresome. Luckily, it is easy to set up a default environment.

For Windows users:

1. Open the `Activate.bat` in `C:\Users\Your_UserName\AppData\Local\Continuum\anaconda3\Scripts`. If it isn't there, try searching.
2. Add the foling line to the bat file `activate iaml`.
3. Test that the environment is automatically activated by opening a new anaconda prompt.

For Mac OS and Linux users:

(a) Open your `~/.bashrc` file (or equivalent if you are using another shell).

[4] https://docs.conda.io/projects/conda/en/latest/index.html

(b) Add the line `conda activate iaml` to the end of the file.

(c) Test by opening a new terminal and check that the environment is activated.

Editors and workflow

During the course, you may either want to use Python directly from the terminal using a text editor such as Visual Studio Code[5] or Sublime Text[6], or indirectly from an IDE such as PyCharm[7] or Wing[8]. These editors are either entirely free or have free editions for students. We recommend you to use whatever solution works best for you but note that the IDE's are likely the easiest starting point.

If you use the terminal for executing Python code, use the `conda activate` command to activate the `iaml` environment at the start of each session. If you use PyCharm or Wing IDE, the environment can be configured when creating new projects or running Python scripts. See their respective documentation pages for additional details.

Jupyter Lab

Jupyter Lab is a web-based IDE for developing interactive *Python* scripts in notebook format, making it possible to intertwine *markdown*-based text with code and visual output. We will be using Jupyter Lab for some of the exercises. Jupyter's main drawback is its inability to show GUI elements from OpenCV. Additionally, we believe you will learn more about application design by using regular source files. We therefore use it mostly for introducing more theoretical concepts as well as data exploration .

To start Jupyter Lab, either run `jupyter lab` from the terminal (remember to activate the *iaml* environment) or use the *Anaconda Navigator* application.

Exercise 1.4
Introduction to Python

This exercise aims to introduce you to a basic workflow using OpenCV and Python with either a text editor and terminal, or using the Jupyter Lab environment.

Task 1: *Editor workflow*

1. Open your favourite text editor and create a new document named `ex1_editor.py`.

2. Type in the following code snippet and save the changes

```
import cv2

img = cv2.imread(<path>)
```

[5] https://code.visualstudio.com/

[6] https://www.sublimetext.com/

[7] https://www.jetbrains.com/pycharm/

[8] https://wingware.com/

```
cv2.imshow(img)
cv2.waitKey()
```

`<path>` should be replaced with either the absolute or relative path to one of the images included in the exercise materials.

3. Now open a terminal (Anaconda prompt on Windows) and navigate to the directory where your script is saved.

4. Activate the *iaml* environment by running `conda activate iaml`.

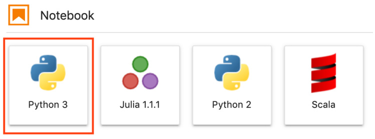
5. Now start your script by running `python ex1_editor.py`. You should see the image appear in a window. You can close the window by tapping any key (handled by the `cv2.waitKey(0)`[9] function).

Task 2: *Jupyter Lab workflow*

(a) Open a terminal (or Anaconda Prompt on windows) and activate the *iaml* environment using the command `conda activate iaml`.

(b) Start Jupyter Lab by running the command `jupyter lab`. This should automatically open Jupyter in a browser window.

(c) Create a new notebook by clicking the `Python 3` button under notebook in the center of the screen as shown to the right.



Alternatively, you can create new files by navigating to **File** ▷ **New** ▷ **Notebook** and then selecting `Python 3` as the kernel.

(d) The code snippet this time is a bit different since `cv2.imshow()` doesn't work in notebooks. Instead, we use **Matplotlib**:

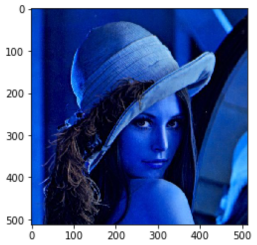
```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread(<path>)

plt.imshow(img)
```

You execute the code by pressing **Shift + Escape**.

(e) You probably noticed that the output looks wrong (compare to our result shown to the right).



This is due to OpenCV using the BGR color format as standard while Matplotlib uses RGB. To convert color spaces, add the following code:

```
cvt = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(cv2)
```

The image should now be displayed correctly.

(f) Try to write the code for converting to grayscale and showing the result. You can find the documentation for `cv2.cvtColor()` here. The documentation is for the C++ version but the API is nearly identical. The color codes in Python are accessed as `cv2.COLOR_<code>`.

[9] https://docs.opencv.org/4.2.0/d7/dfc/group__highgui.html#ga5628525ad33f52eab17feebcfba38bd7

Python REPL

Since Python is an interpreted language, it has a REPL (read-eval-print-loop) which allows you to write and execute code line by line. To start the REPL, simply run `python` in the terminal without any arguments. You should be presented with the following output:

```
Python 3.7.3 | packaged by conda-forge | (default, Dec  6 2019,
    ↪ 08:36:57)
[Clang 9.0.0 (tags/RELEASE_900/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more
    ↪ information.
>>>
```

You can then type any Python code you want, including importing modules.

Documentation and help

We have posted links to relevant documentation pages on the LearnIT page. Additionally, Python has a special function, `help`, which makes it possible to look up documentation for any `class`, `function`, and `module`. It is especially useful in combination with the interactive REPL and Jupyter Lab.

Exercise 1.5

Test scripts

We additionally provide a number of scripts that you can use as starting points when developing your own CV applications:

You can terminate the scripts by pressing *q* or *ESC* on your keyboard.

- `warmUpImage.py`: Opens a JPG figure and show it in an OpenCV window.
- `warmUpGrayscale.py`: Opens a image in the graycale color space and show it in an OpenCV window.
- `warmUpConvert.py`: Opens a colored image, convert it to grayscale, and show both images in two OpenCV windows.
- `warmUpMatplotlib.py`: Opens a grayscale image and show the same image using OpenCV and Matplotlib windows.
- `warmUpMultiple.py`: Opens multiple images and show them as a vector and a matrix using a Matplotlib window.
- `warmUpVideo.py`: Opens a MP4 video file and show it in an OpenCV window.
- `warmUpCapture.py`: Captures a video stream from your web camera and show the frames in an OpenCV window.
- `warmUpRecord.py`: Records an image sequence from your web camera.

Tasks:

1. For each script, get an overview of the code and run the script.
2. If something doesn't work as expected, make sure you installed everything correctly and ask one of your TAs if the problem persists.