CS 763/CS 764: Lab 01

Lab01a: Setup and Ground rules

- Announced 14/01. Due date is the same as that for `lab01b`.
- Please write (only if true) the honor code. You can find the honor code on the web page. If you used any source (person or thing) explicitly state it.
- **This is an individual assignment**

Because deep learning problems are nowadays most easily accessed in Python, the goal of this lab is to get you started with OpenCV using Python bindings.
We know that you can always learn many of these things by yourself anywhere and anytime. But why not schedule this learning? The lab hour 'forces' you to spend time periodically on activities curated for you. You are expected to spend time equivalent to the lab hours, but the exact schedule is left to you. We urge you though to schedule it.
While we expect you to learn on your own elsewhere, and of course in the lab, for the purposes of grading and awarding marks, there may be specific deliverables (sigh)! So make sure you understand the rules for assignments. Especially pay attention everytime to the submission rules – which are syntax, not content. Any violation for any reason whatsover is considered not-to-be-challenged. This is not the place to show creativity. Pay attention to the questions asked even if they are not highlighted in LaTeX . That will tell us that you have read the question paper.

1 Overview

The lab consists of two parts. Setting up the environment, and then learning via doing. The second part is discussed in a companion document.
Relevant in this document is the Ubuntu instructions for the setup, and general submission guidelines.

1.1 Setup

We will setup a Python-based development virtual environment. The purpose of this is to isolate your changes from other things going on in your system.
As an aside, you might have heard about `anaconda` which also supports a python-based virtual environment. We won't need the entire kitchen sink that `anaconda` offers (see, e.g., this page for the differences). You can always install it later. Note that lab submissions may be checked in a vanilla (non-Conda) environment, so it is to your advantage to know what you are using and doing at the basic level. We are also not discussing Python coding environments (e.g. spyder, ipython, pycharm and so on).
The following instructions are decidedly old since we don't have anyone I know using Windows or MacOS for this purpose. So use with caution. The Ubuntu section is kosher, though.
Choose the installation steps according to your operating system.

- Ubuntu
- MacOS
- Windows

1.2 Ubuntu

The following steps are for Ubuntu 18.04 or later (64-bit).

Install Python and venv

Python3 is usually pre-installed in recent Ubuntu. The following steps will install Python, pip package manager and venv (the default virtual env in the language)

```
$ sudo apt update
$ sudo apt install python3-dev python3-pip python3-venv
```

Create a Virtual Environment

Python virtual environments are used to (sort of) isolate package installation from the system. Create a new virtual environment by choosing a Python interpreter and making a venv directory to hold it:

```
$ mkdir foo; cd foo
$ python -m venv .  # create stuff on hard disk
```

Activate the virtual environment :

```
$ source bin/activate
```

When `venv` is active, your shell prompt is prefixed with (venv).
Install packages within a virtual environment without affecting the host system setup. Start by upgrading pip.

```
$ python -m pip install --upgrade pip
```

And to exit env later (this is important)

```
$ deactivate # only exit when not using the environement
```

When you create a new virtual environment, pip will be installed, but that's all. You'll need to install any other packages you want to use in the environment. You should keep in the root of the project a `requirements.txt` file that lists the requirements for the project. This way, if you need to recreate the virtual environment, you can reinstall all of the needed packages with the command `pip install -r requirements.txt`.

Install Libraries

Install Numpy, Matplotlib and OpenCV-Python:

```
$ pip install numpy matplotlib opencv-python  opencv-contrib-python
$ pip install numpy matplotlib opencv-python==3.4.8.29  opencv-contrib-python
    ==3.4.8.29
```

The `SIFT` algorithm is a popular algorithm that we might use but is patented. It is not included in newer versions of opencv-python.

1.3 MacOS

The following steps are for MacOS 10.12.6 (Sierra) or later (64-bit). [Not fully tested]

Install Python and Virtualenv

The following steps will install Python, pip package manager and Virtualenv. Install using the Homebrew package manager.

```
$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/
    master/install)"
$ export PATH="/usr/local/bin:/usr/local/sbin:$PATH"
$ brew update
$ brew install python  # Python 3
$ sudo pip3 install -U virtualenv
```

Create a Virtual Environment

Python virtual environments are used to isolate package installation from the system.
Create a new virtual environment by choosing a Python interpreter and making a venv directory to hold it:

```
$ virtualenv --system-site-packages -p python3 venv
```

Activate the virtual environment:

```
$ source venv/bin/activate
```

When virtualenv is active, your shell prompt is prefixed with (venv).
Install packages within a virtual environment without affecting the host system setup. Start by upgrading pip. :

```
$ pip install --upgrade pip
```

And to exit `virtualenv` later:

```
$ deactivate # only exit when not using the environment
```

Install Libraries

Install Numpy, Matplotlib and OpenCV-Python :

```
$ pip install numpy matplotlib opencv-python==3.4.2.16 opencv-contrib-python
    ==3.4.2.16
```

The `SIFT` algorithm is a popular algorithm that we might use but is patented. It is not included in newer versions of opencv-python.

1.4 Windows

The following steps are for Windows 7 or later (64-bit). [Not fully tested]

Install Python and Virtualenv

The following steps will install Python, pip package manager and Virtualenv. Install the *Microsoft Visual C++ 2015 Redistributable Update 3*. This comes with Visual Studio 2015 but can be installed separately:

1. Go to the Visual Studio Downloads
2. Select Redistributables and Build Tools
3. Download and install the *Microsoft Visual C++ 2015 Redistributable Update 3*

Make sure long paths are enabled on Windows. Install the 64-bit Python 3 release for Windows (select pip as an optional feature).

```
$ pip3 install -U virtualenv
```

Create a Virtual Environment

Python virtual environments are used to isolate package installation from the system. Create a new virtual environment by choosing a Python interpreter and making a venv directory to hold it:

```
$ virtualenv --system-site-packages -p python3 venv
```

Activate the virtual environment.

```
$ venv\Scripts\activate
```

When virtualenv is active, your shell prompt is prefixed with (venv).
Install packages within a virtual environment without affecting the host system setup. Start by upgrading pip.

```
$ pip install --upgrade pip
```

And to exit virtualenv later:

```
$ deactivate # only exit when not using the environement
```

Install libraries

Install Numpy, Matplotlib and OpenCV-Python:

```
$ pip install numpy matplotlib opencv-python==3.4.2.16  opencv-contrib-python
    ==3.4.2.16
```

The `SIFT` algorithm is a popular algorithm that we might use but is patented. It is not included in newer versions of opencv-python.

2 Questions

Answer the questions in a text file called `answers.txt`

1. Which 3 hour period did you reserve this week for doing the lab? Mention date and time in the canonical tab separated CSV format:

   ```
   2022-01-14   14:00:00  17:00:00.
   ```

   It's ok to have multiple rows. Note that it's ok to finish the lab early or late but a reservation must be done.

2. Which virtual environment did you use and why? Answer in no more than one paragraph.

3. How do you deactivate a virtual python environment? Where is this capability defined, and what are the visible and invisible effects of deactivation

4. Which version of opencv did you install?

5. How long did the steps take? Round to the nearest half hour.

6. Although this is not a group assignment, what is your understanding of the contribution section in the `readme.txt` file (see below).

3 Submission Guidelines

1. The top assignment directory should contain the submission as detailed below.

   (a) Do include a `readme.txt` (telling me whatever you want to tell me including any external help that you may have taken). Don't forget to include your honor code here . All members of a group are expected to (electronically) sign the honor code and the percentages (see below). This is a text file, not `pdf`, not `docx`. Do not change the name of the file to REadme.txt or README.txt or README and so on. This is not the place to show creativity – the attention to details might be minor for you but little drops make an ocean.

   The `readme.txt` will contain individual contributions of each of the team members. If the assignment is finally worth 80% as graded by the TA, then a contribution of the form `80, 100, 60` (in sorted order of roll numbers) will result in (respectively) marks `64, 80, 48`. Do this for each question separately. A person claiming 100% is basically saying that (s)he can reproduce the entire assignment without the help of the other team members (after the discussions, if any, are over).

   (b) `answers.txt`: This file should contain the answers to the questions posed.

   (c) `ReflectionEssay.pdf`: This file does not contain answers, but meta-answers. It provides a human perspective of what you learned by doing this. It is not intended to be mechanical, but rather an introspection, like a blog post. Explain what you learnt in this assignment, and how this assignment improved your understanding. What will someone who read this pdf gain? Can this be a blog post, which if read end-to-end someone not in your class (but in your batch) will understand? (Not all of this is relevant to this particular assignment).

   (d) A directory called `code` which contains all source files, and only source files (no output junk files). The mapping of code file to questions should be obvious and canonical. (May not be relevant to this assignment).

   (e) A directory called `data` on similar lines to code, whenever relevant. Note that your code should read your data in a relative manner. (May not be relevant to this assignment).

   (f) A directory called `results` on whenever relevant to store the concerned plots and images. (May not be relevant to this assignment).

   (g) Source files, and only source files (no output junk files). Mac users please don't include junk files such as `.DS_store`. We are not using MacOS.

   (h) Create a directory called `convincingDirectory` which contains anything else you want to share to convince the grader that you have solved the problem. We don't promise to look at this (especially if the code passes the tests) but who knows? This is your chance.

2. Once you have completed all the questions and are ready to make a submission, prepend the roll numbers of all members in your group to the top assignment directory name and create a submission folder that looks like (for group but you get the idea) this `130010009_140076001_150D50001_lab0X.tar.gz`. Notice the numbers are sorted, and any letters in the roll number are in Upper Case.

   Please stick to `.tar.gz`. Do not use `.zip`. Do not use `.rar` Do not use `.tgz`

3. Your inlab submission folder should look something like: (in this example `opencv` is the target):

```
130010009_140076001_150050001_lab01a/
├── ReflectionEssay.pdf
├── answers.txt
├── opencv
│   ├── code
│   │   └── file.py
│   ├── data
│   ├── results
│   └── convincingDirectory
└── readme.txt
```

4. Submission. Very very important.

   (a) The lexicographic smallest roll number in the group should submit the entire payload (with all the technical stuff).

   (b) All other roll numbers submit only `readme.txt` as discussed above.