EECS 442/504 Computer Vision: HW0

Term: Fall 2018
Instructor: Jason J. Corso, EECS, University of Michigan
Due Date: Not Due; Not to be turned in.
Version: 1 (September 10, 2018)

Constraints: This is an open-world voluntary homework assignment that is designed to your hands dirty in the programming environment that we will be using for the semester: Python and ETA. You can search for resources on the web; talk to your friends about the assignment; do anything you would do in a real-world setting.

Goals: Get you working in the programming environment to be used this semester. You will install software packages on your system and you will interact with them. These include git, python, python venvs, opencv, tensorflow, etc.

Problem 1 (0): Background Know-How.

This course requires use of a modern set of programming tools, with which you may not be acquainted. The professor and the GSIs are here to help you, but you want to be in the habit of self-learning for this type of thing as it will be required both in future studies and when you are working a job.

This is a list of tools and where you can find information/documentation. I have done everything I can to make this list comprehensive, but something may have been missed.

- **git**: This is the basic tool used in contemporary software development for version control. You need it at least to clone the software repository for the course. Furthermore, I recommend you use it for your project (extra points will be given). Documentation: `https://git-scm.com/docs`.
- **python**: The modern *language of the realm* when it comes to computer vision, machine learning, deep learning and data science. Learn it, use it. I recommend Python 3 over Python 2 since Python 2 is now deprecated (and Python 3 has support for types, which improves its power). Python virtual environments are recommended for this course. Documentation: `https://docs.python.org/3/`.
- **numpy**: The basic numerical and linear algebra package for python. Documentation: `https://docs.scipy.org/doc/numpy-1.15.1/user/`.
- **matplotlib**: Plotting tools for python, integrated with scipy and numpy. Documentation: `https://matplotlib.org/users/index.html`.
- **opencv**: A basic computer vision library that has core functionality needed for some of our work in the course. In some cases, we will reimplement methods from opencv for the pedagogical value, in others we may use methods out of the box. OpenCV is written in C++/CUDA and has python bindings. Documentation: `https://docs.opencv.org/3.4/`.
- **ETA**: A higher level python library for computer vision and machine learning that facilitates sequences of various operations into reusable modules. ETA sits atop all of the above. Documentation: `https://github.com/voxel51/eta` and note the existence of a `docs` subdirectory.

Problem 2 (0): Setting up your system.

You need a UNIX-based operating system to work on this class, including Linux, BSD or OS X. If you have Windows, install a (free) virtual machine tool, such as `https://www.virtualbox.org/` and install a (free) Linux on it, such as Ubuntu 16.04. The GSIs can help with this or you can look online.

Make sure that you have a python and git installed.

On Linux, you can execute

```
sudo apt-get install git python python3 python-dev python3-dev
```

On OS X, with HomeBrew `https://brew.sh/` you can execute the following, but you may not need to since Apple already packages python 2.7 and a git binary.

```
brew install git python python3
```

Problem 3 (0): Setting up your development environment.

ETA provides rich documentation and installation scripts for installing all of the necessary (and free) packages required for this course. So, we first need to get ETA.

Let us assume you are working in `WORK_DIR`. (You need a network connection for this.)

```
cd ${WORK_DIR}
git clone https://github.com/voxel51/eta
cd eta
```

Note that there is a `README.md` that includes installation instructions. However, these instructions do not use virtual environments, which I believe are your best option to create a maintainable installation. Instead, you can find instructions on the virtual environments in `docs/virtualenv_guide.md`, noting the `.md` means this is a markdown file and viewable in any text editor. Below are a condensed set of instructions for setting up a python3 virtual environment.

Please note that many of you may use Anaconda already; these instructions are assuming no anaconda. As long as you can execute the tests in Problem 4 below, you are welcome to use Anaconda and set it up how you want.

Assume you will install your virtual environments to a directory `venvs` off of your home directory. Let us refer to this as `ENV_DIR` and its value is something like `/home/username/venvs`. But, you can make `ENV_DIR` anything you like.

```
cd ${WORK_DIR}
pip install virtualenv
cd ${ENV_DIR}
virtualenv -p /path/to/bin/python3 eta
```

You should change the path to specify the appropriate python3 you want to use (find it, perhaps, via `which python3`).

I highly recommend you add the following code to your `/.bashrc` (Linux) or `/.bash_profile` (Mac OS X) for activating and deactivating the environment.

```
# Python environments
export ENV_DIR="/path/to/env" # modify this
eta() { source "${ENV_DIR}/eta/bin/activate"; }
exit() {
   case `command -v python` in
      ${ENV_DIR}/*) deactivate;;
      *) builtin exit;;
   esac
}
```

Finally, you can install ETA into the virtual environment by following these commands. Assuming you have sourced the bash resource file above or have logged out and logged back in.

```
eta
cd ${WORK_DIR}/eta
bash install_externals.bash
pip install -e .
exit
```

Now you are finished. You can always activate the virtual environment by typing `eta` and get to work.

And, one more thing: I highly recommend using ipython instead of python. It provides a more usable shell.

```
eta
pip install ipython
ipython
...
```

Problem 4 (0): Configuration and testing.

Now that you have a development environment set up, you need to test it. You, of course, have access to all of the raw tools that were installed. For example, numpy and opencv.

```
eta
ipython
>> import numpy as np
>> A = np.eye(3)
>> import cv2
>> img = cv2.imread('a_path_to_an_image.png', 0)
>> import eta.core.image as etai
>> img2 = etai.read('a_path_to_an_image.png')
```

You can do similar simple operations to load images and video with ETA, but first you need to configure the environment. For this, refer to the section "Setting up your execution environment" inside of the ETA `README.md` file for instructions on how to do this. It is straightforward.

Finally, run the ETA tests, which will make sure all aspects of the computer vision stack are ready for use. To do this, follow the crisp instructions in the ETA `README.md` in the section "Testing your installation."

Problem 5 (0): Getting ready for the semester.

The semester will incldue 4 homework assignments with programming and one substantial project. In all cases, you will be using the ETA-stack that you just setup. It is a mature, open-source computer-vision development environment. Do note, however, that is it probably more "professional" than you are used to in your cases. This is intentional: I want to better prepare you for your professional life outside of school.

To that end, you are encouraged to read some more of the documentation that accompanies ETA, especially the following list. All files are in `eta/docs`:

- `core_dev_guide.md`
- `pipelines_dev_guide.md`
- `modules_dev_guide.md`

In particular, understanding how the pipeline and modules system works will be necessary if you wish to add any functionality beyond the structured ones we provide in the homework assignments.

Finally, I want to explicitly encourage you to add new functionality to ETA. ETA is open-source. Add new examples. Add new methods. Add new modules. Add new pipelines. Contribute to the analytics community by doing so! If you do want do want to contribute, please note that you should follow the `style_guide.md`, and that you will need to submit a pull requestd on the open repository as you cannot merge directly (a pull request from a fork is probably best).