ECE 471/536: Computer Vision
Assignment 1: Image Processing

Due: January 28, 11:59 pm

January 14, 2019

Abstract

This assignment covers topics discussed in the first two weeks of class. The assignment is divided into two main parts: (1) mathematical and theoretical questions, (2) gentle Python programming for computer vision introduction. This assignment is designed to refresh the students of what they already know, as well as provide a quick background in case they are not familiar with the topics.

Instructions are provided first for a reason.
Please read the instructions carefully.

1 Instructions

1.1 Submission Package

All assignments are submitted to your `git repository` located on `git.engr.uvic.ca`. The (a) **last** submitted entry, or (b) the submission with "Final submission" in the commit comments will be considered your submission entry. Marking will commence 72 hours after the deadline on entries that do not have "Final submission" comments for students who are using their late hours. A GIT tutorial is posted on the course page.

1.2 Assignment Report

Reports will be submitted in `pdf` format as `assignX_report.pdf`. Any other format will **not** be accepted.

Do **not explain or submit code** in your assignment report. Reports are for discussing results when appropriate. You **should** provide comments in your code for the marking TA but deductions will not applied if no comments are provided. However, comments may aid the TA in understanding your code if you have provided an incorrect solution.

1.3 Code

All assignments should be in `Python 3`. Code that fail to run on `Python 3` will receive a 20% deduction on the final score. `IPython` or `Jupyter` notebooks are **not accepted**. Please do not change the provided script names, which have been provided for you.

It is **strongly encouraged** to follow PEP8. It makes your code much more readable for the marking TA. You will not, however, be marked on the quality of your code.

1.4 Delayed submission

Each student has a bank of **72 late hours** which they may use for any assignment. No deduction for lateness will be applied while you have late hours. These are deducted in 30 minutes segments. Otherwise, you will receive a deduction of **20% per day**.

1.5 Use of open source code

Short snippets of code on public websites such as StackOverflow may be used without an explicit license, but proper attribution should be given even in such case. This means that if you embed a snippet into your own code, you should properly cite it through the comments.

Please note that without proper attribution, it will be considered plagiarism.

We will be using the `OpenCV 3` open source library available through the python `pip` package manager. However, the use of APIs/classes that perform the bulk for the assignment task will be prohibited. If you are unsure if an OpenCV library is prohibited, just message me on Slack.

In addition, as the assignments are intended for you to learn, (1) if the external code implements the core objective of the task, no points will be given; (2) code (including pseudo code) or written answers from other ECE471/ECE536 students will count as plagiarism.

To be more clear on (1), with the assignment, we will release a requirements.txt file, that you can use with pip to setup your environment.

2 Theoretical (50 points)

2.1 Preliminaries

2.1.1 Linear Algebra (29 points)

NOTE: We will not cover this in class.

Q1) Given the following vector and matrix definitions, answer the following questions (show your work):

$$v = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \quad u = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \quad A = \begin{bmatrix} 1 & 2 & 3 \\ -4 & -5 & -6 \\ 7 & 8 & 9 \end{bmatrix}$$

a) $v + 1$ (1 mark)
b) $v + v^T$ (1 mark)
c) $v \cdot v^T$ (dot product) (3 marks)
d) $v \times u$ (cross product) (3 marks)
e) $v \bullet u$ (matrix multiplication) (3 marks) [NOTE: treat u as a multiple vector matrix (i.e. each row is a separate vector)]
f) $v \odot v^T$ (Hadamard product) (3 marks)
g) Calculate the eigenvalues for matrix A (15 marks)

2.1.2 Statistics (16 points)

NOTE: We will not cover this in class.

Q2) Short answer - Define and explain the following in 2-3 sentences (math definitions are not required, but may be used to expand on your answer) and/or by sketching where appropriate.

a) What is mean and variance? (2 marks)

b) What is the Central Limit theorem? (3 marks)

c) What is the *curse of dimensionality* and how does it apply to Computer Vision? (5 marks)

d) What are the three properties of a probability density function f(x) for a continuous random variable X? (6 marks)

2.1.3 Linear Filtering (5 points)

Q3) Cross-correlation and convolution are similar operations, but convolution is associative and commutative while, in general, correlation isn't. How might this improve computation time (how fast the program will run)?

3 Programming (50 points)

3.1 Setting up a python virtual environment (no points)

You may skip this section if you are already familiar with how Python virtual environments work, and you already know how to set it up. However, it is encouraged to setup a new environment for the assignments, as we will test your Python scripts in a virtual environment with the packages specified in the requirements.txt file.

This step-by-step guide will first go through the process of installing pip the Python package manager, and then installing virtualenv, and then finally setting up the your virtual environment.

3.1.1 Installing Python 3 and pip

In many cases Python 3 may not be installed in your OS. Installing Python 3 is highly dependent on the operating system. For this process, you will need root access.

Linux: If you are using a Linux distribution, use your package manager to install `Python 3` and pip. For example, on Ubuntu/Debian/Mint, it should be as simple as typing in the terminal:

```
sudo apt install python3 python3-pip
```

OSX: Use Homebrew. Install Homebrew by visiting brew.sh, and type in the terminal:

`brew install python3` (This also installs pip)

Windows: For Windows version, please install a Linux Virtual Machine using Virtual Box, or use a Linux server. Then follow Linux instructions. Alternatively, for Windows 10, install the required packages by following this Youtube video.

The following command-line commands should work for Windows 10 PowerShell (don't use the `cmd` terminal), but is currently untested.

3.1.2 Installing and setting up your virtual environment

After installing pip, you can easily install your virtual environment by opening a terminal and typing:

```
pip3 install --user virtualenv
```

Here, we have the –user flag as we are installing this for the current user. This way we wont need root privileges.

Then, you need to specify a directory for your virtual environments. For example, it could be `~/myvenvs`. We will use this directory for the time being, for easy explanation, but any directory is fine. Create that directory, and then all you need to do is:

```
virtualenv --python=$(which python3) ~/myvenvs/a1
```

On Windows you may need to replace `$(which python3)` with the path to you Python3 executable.

Note that we are setting up the virtual environment with python3 by designating the python executable. To work in (activate) this virtual environment, type:

```
source ~/myvenvs/a1/bin/activate
```

To leave the virtual environment, type `deactivate`. In most cases, you will see something on the left of your command prompt, showing that you are inside a virtual environment. **Once inside a virtual environment**, you should now be able to install the exact environment that you will be evaluated through:

```
pip3 install -r <path-to-requirements.txt>
```

For example, if you are already in the directory which you extracted the assignment package, you can type:

```
pip3 install -r requirements.txt
```

To check if everything is okay, you can try which python, that would return something like:

```
/home/adash/myvenvs/a1/bin/python
```

where `/home/adash/` is your home folder, that is, ~/. Also, at this point you should also have `ipython`, which is a prettier version of the python command line interface. Go ahead to try importing `numpy` after launching `ipython` to try things out.

3.1.3 Python IDE

You may use any IDE or text editor you prefer. I prefer `PyCharm`, `Spyder` (which can be installed through pip) or `Vim` (not for the faint of heart). PyCharm has built-in git integration while Spyder does not; however, Spyder is less resource-heavy and I like its `ipython` terminal. `Sublime` is another popular IDE.

3.2 Basic Image Handing (15 points)

3.2.1 Reading, display, and save an image

See provided boilerplate code. The code uses argparse to parse command-line arguments. You do not need to provide any arguments as the code will use default values, but is given as an example.

Your script will perform the following:

1. Reads `pear.png` provided in the assignment package, or any image your pass via the command-line.
2. Displays the image using `matplotlib` or `OpenCV` as a grayscale image.
3. Save an image as `sharpened.png`.

Your final submission will save the result of `imfilter2d`. For testing, I suggest saving the result of `cv2.filter2D` for debugging purposes.

Hint: Keep a careful eye on the data-types for each of your arrays. Matplotlib handles images of float type differently than unsigned integer – it assumes that float images are in the range of [0,1]. OpenCV doesn't like float type images for displaying at all. When saving an image, you should make sure your image is in the range of [0,255].

Useful functions:

- cv2.imread, cv2.imwrite
- cv2.imshow, cv2.waitKey
- plt.imshow, plt.show, plt.plot

Useful reads:

- Numpy ndarray documentation
- Matplotlib image tutorial
- Numpy tutorial by Justin Johnson

3.3 Image Filtering (35 points)

3.3.1 Sharpen image using an approximate Laplacian of Gaussian)LoG) filter

See provided boilerplate code and lecture notes. Use the code from the previous section to display and save your output. You should submit your saved result with your final submission.

Your script will perform the following:

1. Reads `pear.png` provided in the assignment package, or any image your pass via the command-line
2. Create an approximate LoG filter (see Lecture notes) of size `9x9`
3. Apply a border to the input image
4. Apply the filter to the image using convolution
5. Save the result of the convolution to an result array
6. Clip the result (i.e. values less than 0 are set to 0) so the image is in the range [0,255]
7. Save an image as `sharpened.png`. Your final submission will save the result of `imfilter2d`

8. **BONUS:** Write your code to handle any size of kernel

Hint: A Gaussian kernel is a linear kernel and is therefore *separable* and should sum to 1. You may use any border padding before performing convolution, but `cv2.BORDER_REFLECT` is set by default.

Useful functions:

- cv2.getGaussianKernel
- np.zeros, np.ones
- np.float32, np.uint8
- cv2.copyMakeBorder

Prohibited functions:

- cv2.filter2D (you can use it to test your `imfilter2d`)

Useful reads:

- OpenCV image filtering documentation

4 Final Submission

Your final submission in GitLab should include the following:

`a1/`

solution.py

assign1_report.pdf

sharpened.png