EEP 596 Computer Vision

HW #1

The purpose of this homework is to familiarize you with some basic image processing and computer vision concepts using Python and associated libraries. You are free to work with fellow students but should turn in your own work. Specifically, you should be able to explain anything in your assignment if asked.

You should turn in your assignment as a single PDF file, with code and text interleaved, in a neat, readable format. Your file should document the learning process.

Warning: Python is an easy language, but Python packaging / ecosystem is a mess. It is best to keep your expectations low.

The questions are as follows.

0. **Set up.** Install the latest version of Python, NumPy, PyTorch, Matplotlib, OpenCV. It is recommended to install Anaconda, as it includes several of these automatically.

   You can test your setup by running Python from the command line, then importing modules and printing their versions. Below is the output on my system.

```
import sys
import torch
import torchvision
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import cv2

print(sys.version)
print(torch.__version__)
print(torchvision.__version__)
print(np.__version__)
print(matplotlib.__version__)
print(cv2.__version__)
```

   →

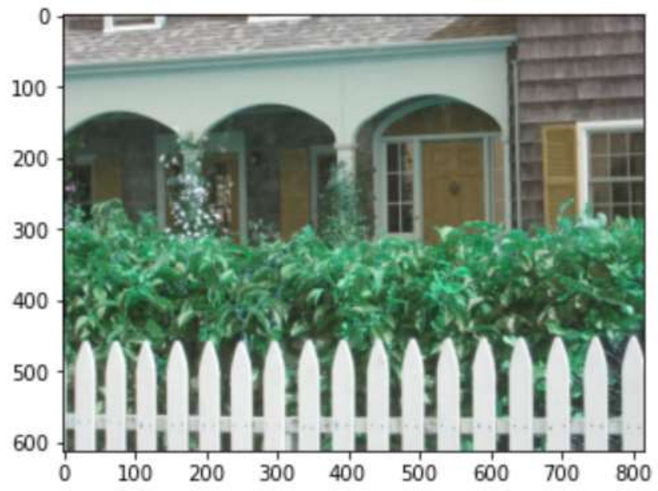
```
3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
1.6.0
0.7.0
1.18.5
3.2.2
4.0.1
```

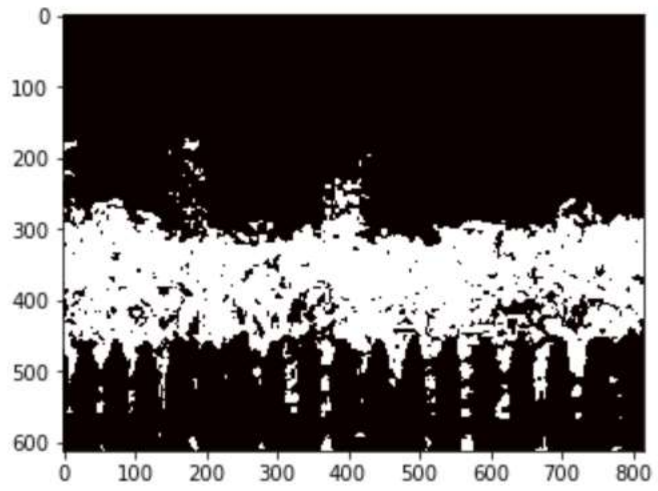
There are two popular ways to run Python:

- Command line. Edit code with your favorite editor (Visual Studio Code, or VSCode, is a good choice.) Save to `foo.py` (for example). Then, type `python foo.py` at the command prompt to run (or use the integrated interpreter via the IDE).
- Jupyter / IPython. Type `jupyter notebook` at the command prompt, and your browser should automatically open to a page that allows you to work interactively.

1. **Image load / save.** Load the `picket_fence.jpg` image provided. Print the data type of the image, and the data type of the pixels in the image. Print the image dimensions. Is the image accessed by (row,column) or (column,row)? What are the red/green/blue pixel values at (0,0)? In what order are the colors stored? What data type is the image? What data type are the pixels?
2. **Color channels.** Create a new RGB image by setting the green and blue pixel values to zero everywhere. The resulting image should look red.
3. **Photographic negative.** Create a new RGB image (from the original image) by subtracting each pixel value from 255. The resulting image should look like a photographic negative.
4. **Swap color channels.** Create a new RGB image (from the original image) by swapping the red and blue channels.
5. **Foliage detection.** Create a new binary image that contains an `ON` pixel wherever the original input pixel is approximately green, and `OFF` everywhere else. *(You will have to think about this a bit, but the solution is not hard.)* Below is my output for comparison. What data type is the output image, and what data type are the pixels? Save the image to a `.png` file on disk; load

the file in a separate image viewing program to verify that the output looks correct.





6. **Shift.** Translate the original image to the right by 200 pixels, and down by 100 pixels. Fill in the missing values with zero.
7. **Rotate.** Rotate the original image clockwise by 90 degrees.
8. **Similarity transform.** Write a function to apply a similarity transform to an input image, yielding an output image of the same size. The function should take scale, rotation angle, and translation as input (all as floating point values). Use nearest neighbor interpolation for simplicity, but be sure to use "inverse mapping" to ensure that every pixel in the output gets painted. Test this on the original image above by passing the values scale=2.0, theta=45.0 (degrees), shift=[100,100]. *(Yes, the order matters here, but the order is up to you, as is the specific interface of the function.)*
9. **Grayscale conversion.** Look up the most common formula to transform RGB to grayscale. (For example, look at OpenCV's `cv::COLOR_RGB2GRAY`.) Apply this formula to the original image. Although the result looks reasonable, it is technically not correct. List two reasons why this common formula is wrong. *(Hint: This is explained in the textbook.)* To convince yourself that the common formula is wrong, apply the formula to an image of all pure blue pixels, that is, (0,0,255). *(Extra credit: Implement the correct formula, assuming Rec. 709, and apply it to both the pure blue image as well as the image above.)*

10. **Moments.** Write a function to compute the first- and second-order moments (both standard and centralized) of a binary image. Apply the function to the image `glasses_outline.png`. Print the results.
11. **Binary image processing.** Using the function from the previous problem, write a function to compute the orientation and eccentricity of a binary image. Apply the function to the image `glasses_outline.png`, and draw an ellipse overlaid on the image showing the "best fitting ellipse". Explain two reasons why it is better to store `glasses_outline` as a PNG file rather than a JPEG. *(Hint: This is explained in the textbook, but you can also discover the answer on your own if you save the image as a JPEG and compare the resulting files.)*