COGS 225, Spring 2020: Homework 1

Due: April 14, 2020 11:59 PM PDT
Late policy: 5% reduction for the first day and 10% reduction afterwards for every extra day past due.
Instructions

1. The purpose of this assignment is to familiarize you with various image processing methods and software packages used in advanced computer vision research. To this end, you will use Python, OpenCV, PyTorch, and Kornia to complete this assignment.
2. Download the resources `Homework1.zip`. It contains the data and the Python code which will be required to complete this assignment. The data is a subset of BSD300 dataset [1].
3. Reference materials and some useful codes can be found at the end of the document.
4. Please submit your report describing about the experiments you run and the corresponding results. **You are encouraged to work on top of provided iPython notebooks and convert them into a pdf file.** Grading of your report will be based on standard metrics: (1) the quality of the writing, (2) your unique insights and ideas, (3) thoroughness of the experiments, (4) conclusive results, (5) references if any.

1 (5 points) Basic Matrix Operations with NumPy and PyTorch

NumPy and PyTorch will be primary software packages for this class. You may want to study the basics of NumPy by referring to the Python NumPy tutorial. PyTorch will be required for advanced computer vision applications that require GPU-compute. PyTorch has a similar interface to NumPy. Once you are familiar with NumPy, you can quickly familiarize yourself with PyTorch by referring to the official PyTorch tutorial.

In this problem, we will practice basic matrix/tensor arithmetic in NumPy and PyTorch.
Suppose $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}, B = \begin{bmatrix} -1 & 1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix}$.

1.1 (3 points) NumPy Arithmetic

We can convert A and B into NumPy arrays by

```
import numpy as np
A = np.array([[1,2], [3,4], [5,6]])
B = np.array([[-1,1], [1,-1], [-1,1]])
```

NumPy package uses `np.dot(A,B)` or `A.dot(B)` to compute the dot product between matrices A and B. Binary operators such as `*`, `/`, `+` and `-` compute the element-wise operations between two matrices. Please compute the following: (if the matrix multiplication is not possible, write 'impossible' in your answer).

1. $A + B =$

2. $A \circ B =$
   ('$\circ$' operator means element-wise product or Hadamard product)
3. $A^T B =$
4. $AB^T =$
5. $AB =$

1.2 (2 points) PyTorch Arithmetic

Please use PyTorch to solve the problems in Section 1.1.

2 (15 points) Basic Image Operations with OpenCV and Kornia

This problem will teach us a basic pipeline for image processing with PyTorch. First, we will open an image with OpenCV. The image will be saved as a NumPy array. Second, in order to process the image with PyTorch, we have to convert the NumPy array to a PyTorch tensor by invoking functions *e.g.*, `torch.from_numpy` or `kornia.image_to_tensor`. Finally, we process the tensor with PyTorch modules. Additionally, you might want to convert the tensor back to NumPy array in order to use packages that run on top of NumPy.

For OpenCV-Python, please refer to this OpenCV tutorial.

Kornia [4] is a recently introduced computer vision package for PyTorch. For Kornia, example codes in the official GitHub repository will be helpful.

For problems in this section, start from the skeleton code given in `Problem_2.ipynb`.

2.1 (4 points) Image Read/Write with OpenCV

Read any image from the dataset in the resource folder (data/img) in color, display it in RGB using pyplot and write it in RGB and Grayscale.

2.2 (6 points) Image Smoothing with OpenCV and Kornia

Take any 3 images from the dataset in the resource folder (data/img) and perform average smoothing (also known as box blur) and Gaussian smoothing. Try with at least 3 filter sizes including $3 \times 3, 21 \times 21, 45 \times 45$. For the sigma value of the Gaussian filter, please use this. Report the results with both OpenCV and Kornia. What do you observe? Do you observe the same results on both packages?

2.3 (5 points) Denoising with Kornia

For this question use the images present in (data/snp) folder. The images in this dataset contains salt and pepper noise which can effectively be removed using **Median Filtering**. Denoise all the images in the dataset and report your results with Kornia. Also report the filter size that you have used.

3 (20 points) Histogram Equalization with OpenCV

In this problem, you will implement contrast adjustment in OpenCV. You may want to study the basics of histogram equalization in *e.g.* Wikipedia. For problems in this section, start from the skeleton code given in `Problem_3.ipynb`.

3.1 (5 points) Histogram with OpenCV

Take any 3 images from the dataset in the resource folder (data/img) and show their histogram for R, G and B color intensity values. For each image, histograms for R, G and B has to be in the same plot. What can you say about the spread of the color intensities for these images?

3.2 (5 points) Global Histogram Equalization with OpenCV

Perform Global Histogram Equalization on the images used in the previous part. Report the original and enhanced images and their corresponding histogram.

3.3 (10 points) Contrast-Limited Adaptive Histogram Equalization (CLAHE) with OpenCV

Again take the images used in the previous part. This time perform Contrast Limited Adaptive Histogram Equalization with Contrast Limiting. Report the results similar to the previous part. Also report the hyperparameters used. Did you find any difference between the two techniques?

4 (60 points) Edge Detection

In the lecture, we have learned the history of edge detection methods. Now, you will experiment with **three** methods from the following edge detectors: Sobel Operator, Canny [2], and Structured Forests [3], Holistically-Nested Edge Detection (HED) [5]. Test them on at least 4 of the images present in the resource folder (data/img). For the evaluation, use the Python script provided (Instructions to run the code are present in `Problem_4.ipynb`). Report the accuracy obtained (using the evaluate method) for all the edge detectors. Discuss any differences between different edge detectors based on what you have learned in class. Please report the implementations and the results of any improvements you have tried on all 3 of the selected method. You may submit your discussions either as a part of iPython notebook (markdown) or separate pages attached to the iPython notebook pdf file.

You may find the following Python implementations useful.

Sobel Operator

- OpenCV.
- SciPy.
- Kornia.

Canny

- OpenCV.
- skimage.

Structured Forests

- Python implementation by ArtanisCV.

HED

- OpenCV implementation by Adrian Rosebrock.
- OpenCV implementation by Ankit Sachan.
- PyTorch implementation by Weijian Xu.

References

[1] Pablo Arbelaez, Charless Fowlkes, and David Martin. The berkeley segmentation dataset and benchmark. 2007.

[2] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-8(6):679–698, November 1986.

[3] P Dollár and C L Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013.

[4] D. Ponsa E. Rublee E. Riba, D. Mishkin and G. Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *WACV*, 2020.

[5] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *ICCV*, 2015.