EECS 504 Computer Vision: HW3

Term: Fall 2018
Instructor: Jason J. Corso, EECS, University of Michigan
Due Date: 11/07 23:59 Eastern Time

Constraints: This assignment may be discussed with other students in the course but must be written independently. Programming assignments should be written in Python using the open-source library ETA. Over-the-shoulder Python coding is strictly prohibited. **Web/Google-searching for background material is permitted. However, everything you need to solve these problems is presented in the course notes and background materials, which have been provided already.**

Data: You need to download `hw3.zip` to complete this assignment. All paths in the assignment assume the data is off the local directory.

Problem 1 (30): House Number Classification

Classify house numbers using an eigenbasis. Below are some example images in the Street View House Number (SVHN) Dataset.

First of all, to get the basis of the digits, we use the MNIST Dataset as training data. MNIST is a large database of handwritten digits. Download its training set images and labels from Yann LeCun's website. Make sure you store the .tar.gz files in the `data` folder. Every image contains a digit and has a size of 28×28. The original files downloaded are not in the standard image format (see how the pixels are arranged on the website). We have provided you code to read these files.

The SVHN test image dataset is present in `data/test.tar.gz`. Extract it. This should create a folder called `test`, which contains the test images as .png files and `digitStruct.mat`. `digitStruct.mat` stores the annotation of images, the image labels and the bounding boxes around each number. Refer to the official site for a more detailed description about the format of these annotations. We have provided you with a helper class located in `modules/dig_struct.py` to read this file. Please look through this class to know how to obtain the bounding boxes and any other information you may need.

You are required to implement a K nearest neighbor classifier to recognize the house numbers for the **first 300 images** in the **test dataset**. The number K is of your choice. Make use of the bounding box information to crop the image to the region of a specific digit and resize it to 28×28 (think about why). Note that there may be multiple digits in the same image and they are classified one by one to form a house number. Compare the house number generated by your program with the groundtruth and calculate the Digit Error Rate using the formula below.

$$\text{Digit Error Rate} = \frac{\text{number of wrong digits}}{\text{total number of digits in all images}}$$

You should test your basis on the MNIST test set first, i.e., select a subset of MNIST test images (first 500, for example) and report the error rate in the writeup. Then test it on SVHN.
Note that you need to store these two error rates in specifically named variables within the code. Comments are provided in the code with more information.

Submission requirement:

1. Visualize first 10 basis learned from MNIST that are associated with 10 largest eigenvalues; attach the visualization to your writeup. Note that the actual number of basis used for classification doesn't have to be 10.

2. Include one example of the hand-written digit in MNIST that can be correctly classified and one example that fails. Also include one example of the house number in SVHN that is incorrectly classified.

3. Report the error rate as described above on **both** MNIST and SVHN. The error rates are written in `out/error_rates.json`. Report the number of correctly classified house numbers. Discuss why or why not your program works well on SVHN.

You have to install h5py and matplotlib packages to run the code for this problem. Use the following command to install the packages:

```
pip install h5py
pip install matplotlib
```

You can run the pipeline using the following command:

```
eta build -r requests/svhn_request.json --run-now
```

Problem 2 (25): **EXTRA CREDIT (Optional)** Keypoints, Descriptors and Matching

Given below is an image, `data/image_of_cathedral.jpg`, of a beautiful cathedral in Italy captured by Prof. Corso. We are not providing you the name or GPS coordinates of the cathedral. In this question, you are asked to find the cathedral using Computer Vision. You are free to use any python packages like Opencv, Scipy etc.

Figure 1: image_of_cathedral.jpg

You need not use ETA for implementation. If you are using ETA, submit all the files you have written including modules, requests and pipelines. Your request file should be named `hw3p3.json`. If you are not using ETA, submit a .py file with name `hw3p3.py`. You should submit the name of the cathedral and also the matching image of the cathedral from Flicker in your writeup. Your work should be original and not a replica from any online repositories.

Hints: Create a web crawler to parse through Flicker Images https://www.flickr.com/. You can use the feature descriptors like SIFT to match the images.

Note: No partial marks for this problem

Submission Process: Submit a single typewritten pdf with your answers to these problems, including all plots and discussion. Submit the pdf to Gradescope.

For coding assignments, include your code verbatim in your writeup for each question. Pack the original program files into one zip file and upload it to Canvas. **Code should be well-commented for grading**

Grading and Evaluation: The credit for each problem in this set is given in parentheses at the stated question (sub-question fraction of points is also given at the sub-questions). Partial credit will be given for both paper and python questions. For python questions, if the code does not run, then limited or no credit will be given.