EECS 504 Foundations of Computer Vision: HW3

Term: Winter 2018
Instructor: Jason J. Corso, EECS, University of Michigan
Due Date: 2/23 23:59 Eastern Time

Constraints: This assignment may be discussed with other students in the course but must be written independently. Over-the-shoulder 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.**

Goals: Practice the usage of feature descriptors and deepen the understanding of image as points concept.

Data: You need to download `hw3.zip` to complete this assignment. All paths in the assignment assume the data is off the local directory. Besides, you need to download extra datasets for problem 3. Links are given in the spec.

Problem 1 (20+5): Matching balloons with SIFT

In HW2, you've learned how to find the feature points and their characteristic scales. Now we step further to describe and match the feature points.

Below are two images of colorful hot air balloons taken from different point of views. We want to match the balloons in the two images. Assume that we've already got the positions of feature points by using some fancy detectors; each point indicates one balloon at that position. To match those feature points in the two images, we need to measure the similarity between points by comparing the descriptors calculated from regions around them. Since there might be variance in scale, rotation or illumination, SIFT is a good choice to help us complete this task.

In this problem, we follow the SIFT implementation in David Lowe's paper. There are three main steps to do SIFT: feature points localization, orientation assignment and descriptor computation.

Localization means to find the location and scale of the feature points. You're provided a list of 5 feature points detected from each image, stored in `points1.mat` and `points2.mat`. To fix the scale of the balloons, you may use your `DoGScaleSpace.m` from HW2 or you could use `j_DoGScaleSpace.p`. It is recommended to use smaller increments of sigma and thus more levels in the scale space to get a good approximation. One possible setting is: `k = 1.1, s1 = 4*k, level = 25`. It is NOT the optimal combination. You should find what works for your program and report it.

Dominant orientation assignment basically makes your descriptor rotationally invariant. It is an optional task and you will get **5 bonus points** for implementing this step. `points3.mat` saves the feature points from a rotated image. Try matching `balloons1.jpg` and `balloons3.jpg` if you do this task.

There are a lot of details in descriptor computation. The major procedure, however, will be consistent with what was discussed in the class. Follow the comments in `sift.m` and fill it with your implementation. Refer to the paper for concrete instructions if necessary.

The main script is `matching_balloons.m`; it will call your `sift` function, match the points based on feature descriptors and display the matchings.

Example output:

Submission requirement:

Attach the output images (with matchings) to your writeup. Point out what parameters you're using in your program. Provide brief explanation of your idea if you do the optional task and show your matching result on the rotated image.

You do not need to include the code in your report for this problem. Submit the original program files to Canvas (see naming at the bottom of spec). Include all the files needed to run your program, including those provided to you.

Problem 2 (30): Haar Wavelet Transform

1. (12) **Transform computation** Compute the Haar wavelet representation for the following 2D image and show derivation. Answer how many wavelet bases we need to do the wavelet transform for this particular image, and draw the bases associated with the lowest-level detail coefficients. Your drawings should indicate the size and values of the basis. (Note: the highest-level detail coefficients are those computed immediately from the image and thus encoding the finest details.)

0	4	2	0
6	2	1	7
3	4	5	2
0	1	6	3

2. (7) **Wavelet compression** The wavelet transform provides a plausible means of compression by dropping certain coefficients. The most straightforward way of doing this is to cull the full set of coefficients of highest-level detail, i.e., set the coefficient layer to zero. The reconstruction is the same as what we do without compression. For the 1D image below, compute the reconstructed image after compression. Show your derivation.

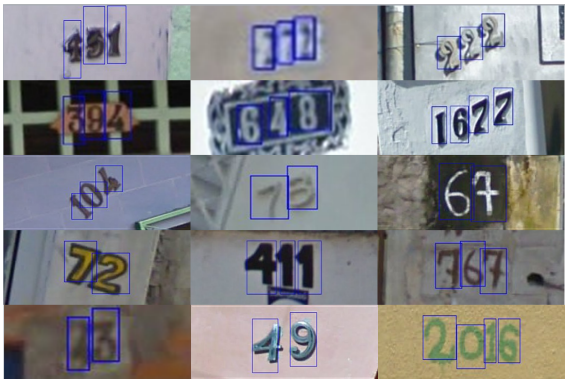
2	4	2	0	6	2	1	7

3. (5) **Reconstruction error** Consider a 1D image with 2^n pixels. Assume that the bit-depth is b (grayscale image values with range 0 through $2^b - 1$). Using the compression scheme above, derive an upper bound for the maximum error between the reconstructed image and original image. (The error here is computed by using l_2 norm).
4. (6) **Wavelets for description and matching** The wavelet transform could be considered as a representation of an image patch for the purpose of matching. Problem 1 in this assignment casts light on how we use the histogram of oriented gradient representation of a patch for matching. In fact, one could relate the operations in computing the Haar wavelet to the operations used in computing image gradient in SIFT representations associated with specific orientation.

   Make an argument for or against using the wavelet representation of the image patch to replace the SIFT descriptor. Use the ideas that we have discussed in the course in your argument; for example, is the wavelet transform shift invariant? Is the SIFT shift invariant? How do they deal with scale? There is a correct answer here (one has quite better properties than the other for the purposes of representing image patches for matching).

Problem 3 (30): House Number Classification

Recall the eigenface example discussed in the class. Now apply the same technique to classify house numbers. 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 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. 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). You may search for helper functions to read the images and labels (one can be found on this site).

Then download the SVHN test images from its official site. There are two formats of data, original images and cropped and centered images. Format 1 (`test.tar.gz`) is what we are going to use for classification. The annotation of images, including the labels and bounding boxes, are stored in `digitStruct.mat`, which can also be found in the downloaded folder. See detailed description about the format of the annotation on the official site.

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.

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

Also report how many house numbers you are able to recognize correctly.

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. If your program performs badly on SVHN, think about if there is something wrong in your code or if it is normal to get such a low accuracy.

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 can be correctly classified and one example that fails.

3. Report the error rate as described above on **both** MNIST and SVHN. Report the number of correctly classified house numbers. Discuss why or why not your program works well on SVHN.

4. **You do not need to attach the code in your report**. Submit the original program files to Canvas. Do not include the dataset. Add a README file to clarify the usage of your code.

Submission Process: Submit a single pdf with your answers to these problems, including all plots and discussion. Submit the pdf to Gradescope. The entry code of this course is **9GEG8X**.

For coding assignments, include your code verbatim in your writeup. Put original program files in separate folders corresponding to problems and name your folders as `hw3p1`, `hw3p2`, etc. Pack all the folders into one zip file named as `hw3_<unique_name>.zip` and upload it to Canvas. The problem description will clarify whether you need to attach your code verbatim and/or turn in the original files for that problem, or both. **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 Matlab questions. For Matlab questions, if the code does not run, then limited or no credit will be given.