ECS 174: Intro to Computer Vision, Spring 2019
Problem Set 3

Instructor: Yong Jae Lee (yongjaelee@ucdavis.edu)
TA: Xueyan Zou (zxyzou@ucdavis.edu)
TA: Yangming Wen (ymnwen@ucdavis.edu)
TA: Wei-Pang (Tyler) Jan (wjan@ucdavis.edu)

Due: Tuesday, June 4th, 11:59 PM

Instructions

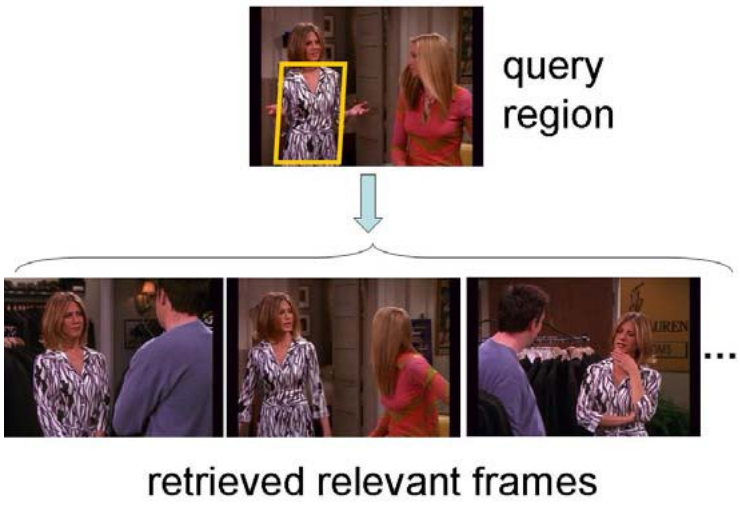
1. Answer sheets must be submitted on Canvas. Hard copies will not be accepted.
2. Please submit your answer sheet containing the written answers in a file named: FirstName_LastName_PS3.pdf.
3. Please submit your code and input/output images in a zip file named: FirstName_LastName_PS3.zip. Please do not create subdirectories within the main directory.
4. You may complete the assignment individually or with a partner (i.e., maximum group of 2 people). If you worked with a partner, provide the name of your partner. **We will be using MOSS to check instances of plagiarism/cheating.**
5. For the implementation questions, make sure your code is documented, is bug-free, and works out of the box. Please be sure to submit all main and helper functions. Be sure to not include absolute paths. Points will be deducted if your code does not run out of the box.
6. If plots are required, you must include them in your answer sheet (pdf) and your code must display them when run. Points will be deducted for not following this protocol.

1 Short answer problems [10 points]

1. What exactly does the value recorded in a single dimension of a SIFT keypoint descriptor signify?

2. A deep neural network has multiple layers with non-linear activation functions (e.g., ReLU) in between each layer, which allows it to learn a complex non-linear function. Suppose instead we had a deep neural network without any non-linear activation functions. Concisely describe what effect this would have on the network. (Hint: can it still be considered a *deep* network?)

2 Programming: Video search with bag of visual words [90 points]



For this problem, you will implement a video search method to retrieve relevant frames from a video based on the features in a query region selected from some frame. We are providing the image data and some starter code for this assignment.

Provided data

You can access pre-computed SIFT and deep features here:
https://drive.google.com/open?id=10yk7tvDfmge9fEVm2XbwAmaIRL9R7clK

The associated images are stored here:
https://ucdavis.box.com/s/ylxih5tgwja1azx78jkc0d5awcxla71m

Please note the data takes about 6 GB. Each .mat file in the provided SIFT data corresponds to a single image, and contains the following variables, where n is the number of detected SIFT features in that image:

```
descriptors    nx128    double   // SIFT vectors as rows
imname         1x57     char     // name of image file that goes with this data
numfeats       1x1      double   // number of detected features
orients        nx1      double   // orientations of the patches
positions      nx2      double   // positions of the patch centers
scales         nx1      double   // scales of the patches
deepFC7        1x4096   double   // AlexNet FC7 feature for entire image
```

Provided code

The following are the provided code files. You are not required to use any of these functions, but you will probably find them helpful. You can access the code here:

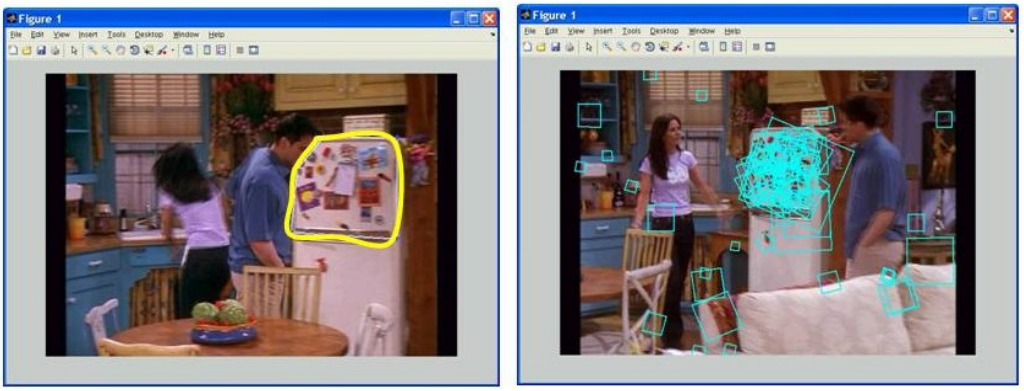
https://ucdavis.box.com/s/cll544a6gq4zaqgf6emn9uf3cq5gwy51

- `loadDataExample.m`: Run this first and make sure you understand the data format. It is a script that shows a loop of data files, and how to access each SIFT descriptor. It also shows how to use some of the other functions below.
- `displaySIFTPatches.m`: given SIFT descriptor info, it draws the patches on top of an image
- `getPatchFromSIFTParameters.m`: given SIFT descriptor info, it extracts the image patch itself and returns as a single image
- `selectRegion.m`: given an image and list of feature positions, it allows a user to draw a polygon showing a region of interest, and then returns the indices within the list of positions that fell within the polygon.
- `dist2.m`: a fast implementation of computing pairwise distances between two matrices for which each row is a data point
- `kmeansML.m`: a faster k-means implementation that takes the data points as columns

What to implement and discuss in the write-up

Write one script for each of the following (along with any helper functions you find useful), and in your pdf writeup report on the results, explain, and show images where appropriate. Your code must access the frames and the SIFT features from subfolders called 'frames' and 'sift', respectively, in your main working directory. However, **Do not include these two folders in your zip file as it will be too big for Canvas.**

1. **Raw descriptor matching [20 pts]:** Allow a user to select a region of interest (see provided `selectRegion.m`) in one frame, and then match descriptors in that region to descriptors in the second image based on Euclidean distance in SIFT space. Display the selected region of interest in the first image (a polygon), and the matched features in the second image, something like the below example. Use the two images and associated features in the provided file `twoFrameData.mat` (in the zip file) to demonstrate. Note, no visual vocabulary should be used for this one. Name your script `raw_descriptor_matches.m`



2. **Visualizing the vocabulary [20 pts]:** Build a visual vocabulary. Display example image patches associated with two of the visual words. Choose two words that are distinct to illustrate what the different words are capturing, and display enough patch examples so the word content is evident (25 patches per word displayed). See provided helper function `getPatchFromSIFTParameters.m`. Explain what you see. Name your script `visualize_vocabulary.m`. Please submit your visual words in a file called `kMeans.mat`. This file should contain a matrix of size kx128 called `kMeans`.

3. **Full frame queries [20 pts]:** After testing your code for bag-of-words visual search, choose 3 different frames from the entire video dataset to serve as queries. Display each query frame and its M=5 most similar frames (in rank order) based on the normalized scalar product between their bag of words histograms. Explain the results. Name your script `full_frame_queries.m`

4. **Region queries [20 pts]**: Select your favorite query regions from within 4 frames (which may be different than those used above) to demonstrate the retrieved frames when only a portion of the SIFT descriptors are used to form a bag of words. *Try to include example(s) where the same object is found in the most similar M frames but amidst different objects or backgrounds, and also include a failure case.* Display each query region (marked in the frame as a polygon) and its M=5 most similar frames. Explain the results, including possible reasons for the failure cases. Name your script `region_queries.m`

5. **Full frame queries, Part 2: comparing SIFT bag-of-words with Deep Features [10 pts]:** Use frames `friends_0000004503.jpeg` and `friends_0000000394.jpeg` to serve as queries. For each query display: the query frame and (1) its M=10 most similar frames based on the normalized scalar product between their bag of words histograms and (2) its M=10 most similar frames based on the normalized scalar product between their AlexNet fully-connected layer 7 activation features (stored as variable `deepFC7`). (The AlexNet was pre-trained on the 1000-class ImageNet classification task, and we are using it to extract the layer-7 activation features for each image.) Explain the differences between the retrieval results obtained using the SIFT bag-of-words features versus the pre-trained deep convolutional neural network features. Which does better? Why? Name your script `compare_bow_and_deep.m`

Tips: overview of framework requirements

The basic framework will require these components:

- **Compute nearest raw SIFT descriptors.** Use the Euclidean distance between SIFT descriptors to determine which are nearest among two images' descriptors. That is, "match" features from one image to the other, without quantizing to visual words.

- **Form a visual vocabulary.** Cluster a large, representative random sample of SIFT descriptors from some portion of the frames using k-means. Let the k centers be the visual words. The value of k is a free parameter; for this data something like k=1500 should work, but feel free to play with this parameter [see Matlab's `kmeans` function, or provided

`kmeansML.m` code]. *Note*: you may run out of memory if you use all the provided SIFT descriptors to build the vocabulary.

- **Map a raw SIFT descriptor to its visual word.** The raw descriptor is assigned to the nearest visual word. [see provided `dist2.m` code for fast distance computations]

- **Map an image's features into its bag-of-words histogram.** The histogram for image I_j is a k-dimensional vector: $F(I_j) = [freq_{1,j}, freq_{2,j}, \ldots, freq_{k,j}]$, where each entry $freq_{i,j}$ counts the number of occurrences of the i-th visual word in that image, and k is the number of total words in the vocabulary. In other words, a single image's list of n SIFT descriptors yields a k-dimensional bag of words histogram. [Matlab's `histc` is a useful function]

- **Compute similarity scores.** Compare two bag-of-words histograms using the normalized scalar product.

- **Sort the similarity scores** between a query histogram and the histograms associated with the rest of the images in the video. Pull up the images associated with the M most similar examples. [see Matlab's `sort` function]

- **Form a query from a region within a frame.** Select a polygonal region interactively with the mouse, and compute a bag of words histogram from only the SIFT descriptors that fall within that region. [see provided `selectRegion.m` code]

3 OPTIONAL: Extra credit (10 points)

- **Stop list and tf-idf**. Implement a stop list to ignore very common words, and apply tf-idf weighting to the bags of words. Discuss and create an experiment to illustrate the impact on your results.