Assignment 3: Content Based Image Retrieval
Vision and Image Processing

Søren Olsen and François Lauze

December 10, 2018

This is the third mandatory assignment on the course Vision and Image Processing. The goal for you is to get familiar with the Bag Of Words (BoW) principle and its use in content based image retrieval systems.

This assignment must be solved in groups. We expect that you will form small groups of 2 to 4 students that will work on this assignment. You have to pass this and the other 3 mandatory assignments in order to pass the course. If you do not pass this assignment, but you have made a SERIOUS attempt, you will get a second chance of submitting a new solution.

The deadline for this assignment is Monday 10/1, 2019 at 23:59. You must submit your solution electronically via the Absalon home page. For general information on relevant software, requirement to the form of your solution including the maximal page limit, how to upload on Absalon etc, please see the first assignment.

Bag of Visual Words Content Based Image Retrieval

The goal of the assignment is to implement a prototypical CBIR system. We recommend the use of the CalTech 101 image database `http://www.vision.caltech.edu/Image_Datasets/Caltech101/`. We recommend that you (for a start) select a subset of say 4-5 categories. When you have checked that everything works you may extend to say 20 categories. *Beware that with many categories and training images, the codebook generation via clustering might be very computationally intensive.* For each category, the set of images should be split in two: A training set and a test set (of equal size). The test set must not include images in the training set. When using few categories you may also limit the number of training images (to say 10) per category. Depending on your amount of computational power, for more categories, you may increase the number of training images to the double or more.

You should extract visual words using SIFT descriptors (ignoring position, orientation and scale) or similar descriptors extracted at interest points. To compute the descriptors, we recommend to use VLFeat's `sift`, but other options are possible.

1 Codebook Generation

In order to generate a code book, select a set of training images. Then Extract SIFT features from the training images (ignore position, orientation and scale). The SIFT features should be concatenated into a matrix, one descriptor per row, i.e., if you use n training images, you will concatenate the descriptors into one single matrix.

Then you should run the k-means clustering algorithm on the subset of training descriptors to extract good prototype (visual word) clusters. A reasonable k should be small (say between 200 and 500) for a small number of categories (say 5) and larger (say between 500 and 2000) for a larger number of categories. Also, a good value of k may depend on the complexity of your data. You should experiment with a few different values of k (but beware that this can be rather time-consuming).

Once clustering has been obtained, classify each training descriptor to the closest cluster centers) and form the bag of words (BoW) for each image in the image training set.

For extraction of the SIFT features in python, using Python3 and OpenCV, here a simple example from the WWW[1]:

```
$ python
>>> import cv2
>>> image = cv2.imread("test_image.jpg")
>>> gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
>>> sift = cv2.xfeatures2d.SIFT_create()
>>> (kps, descs) = sift.detectAndCompute(gray, None)
>>> print("# kps: {}, descriptors: {}".format(len(kps), descs.shape))
# kps: 274, descriptors: (274, 128)
```

For the k-means algorithm, you should use one of the implementations provided by OpenCV, `scipy.clusters.vq.kmeans` or Scikit-Learn implementation: `sklearn.cluster.k_means`.

2 Indexing

The next step consists in content indexing. For each image in the test set you should:

- Extract the SIFT descriptors of the feature points in the image,
- Project the descriptors onto the codebook, i.e., for each descriptor the closest cluster prototype should be found,
- Construct the generated corresponding bag of words, i.e, word histogram.

Please note that you have already performed the same steps for the training images during codebook generation.

The result should be saved in a table that would contain, per entry at least the file name, the true category, if it belongs to the training- or test set, and the

[1]https://www.pyimagesearch.com/2015/07/16/where-did-sift-and-surf-go-in-opencv-3/

corresponding bag of words / word histogram. The table need only be computed once and then used repeatably in the following retrieval experiments.

3 Retrieving

Finally, you should implement retrieving of images using some of the similarity measures discussed in the course slides. You may use:

- common words
- tf-ifd similarity
- Bhattacharyya distance or Kullback-Leibler divergence

Your report should show commented results for two experiments. In the first you consider retrieving training images. In the second you test how well you can classify test images. Otherwise the two test are identical. For each test you should count:

- The mean reciprocal rank (i.e. the average across all queries of $1/\text{rank}_i$, where rank_i is the rank position of the first correct category for the i'th query).
- How often (in per cent) the correct category is in top-3

Please note that the measures above are just two among a long list of possible performance measures. If you google *Information retrieval* you may find alternative measures.