CS 484, Fall 2019
Homework Assignment 2: Local Features

Due: November 20, 2019

1 Introduction

Content-based image retrieval (CBIR) is the problem of searching for images in large databases based on their contents rather than the metadata such as keywords, tags, or descriptions. This problem has been traditionally tackled by using holistically computed low-level visual features such as color or texture histograms. Another popular approach is the bag-of-words (aka. bag-of-features, bag-of-keypoints) model for image representation. The goal of this homework assignment is to perform CBIR by using such a representation based on local features.

2 Background

The bag-of-words model for image representation will be discussed during the lecture but you should read the following papers to learn more about the model (the papers can be found on the course home page):

- G. Csurka, C. R. Dance, L. Fan, J. Willamowski, C. Bray, "Visual Categorization with Bags of Keypoints," European Conference on Computer Vision, 2004.
- L. Fei-Fei, P. Perona, "A Bayesian Hierarchical Model for Learning Natural Scene Categories," IEEE Conference on Computer Vision and Pattern Recognition, 2:524–531, June 20–25, 2005.

3 Approach

The approach can be summarized in terms of the following steps.

3.1 Preparing data

Download the data set for this assignment from the course home page. It includes a subset of the object/scene data set developed at Stanford University:

- J. Z. Wang, J. Li, G. Wiederhold, "SIMPLIcity: Semantics-sensitive Integrated Matching for Picture LIbraries," IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(9):947–963, 2001.

The subset we will use contains 10 categories and 100 images in each category for a total of 1000 images. The images are organized into folders based on their categories. You can see example images in Figure 1.

3.2 Detecting local features

The first processing step is to run the detector of the Scale-Invariant Feature Transform (SIFT) on each image to obtain a set of interest points. The output of this step is, for each image, a set of keypoints represented with four numbers: x and y location, scale (for which the point is a

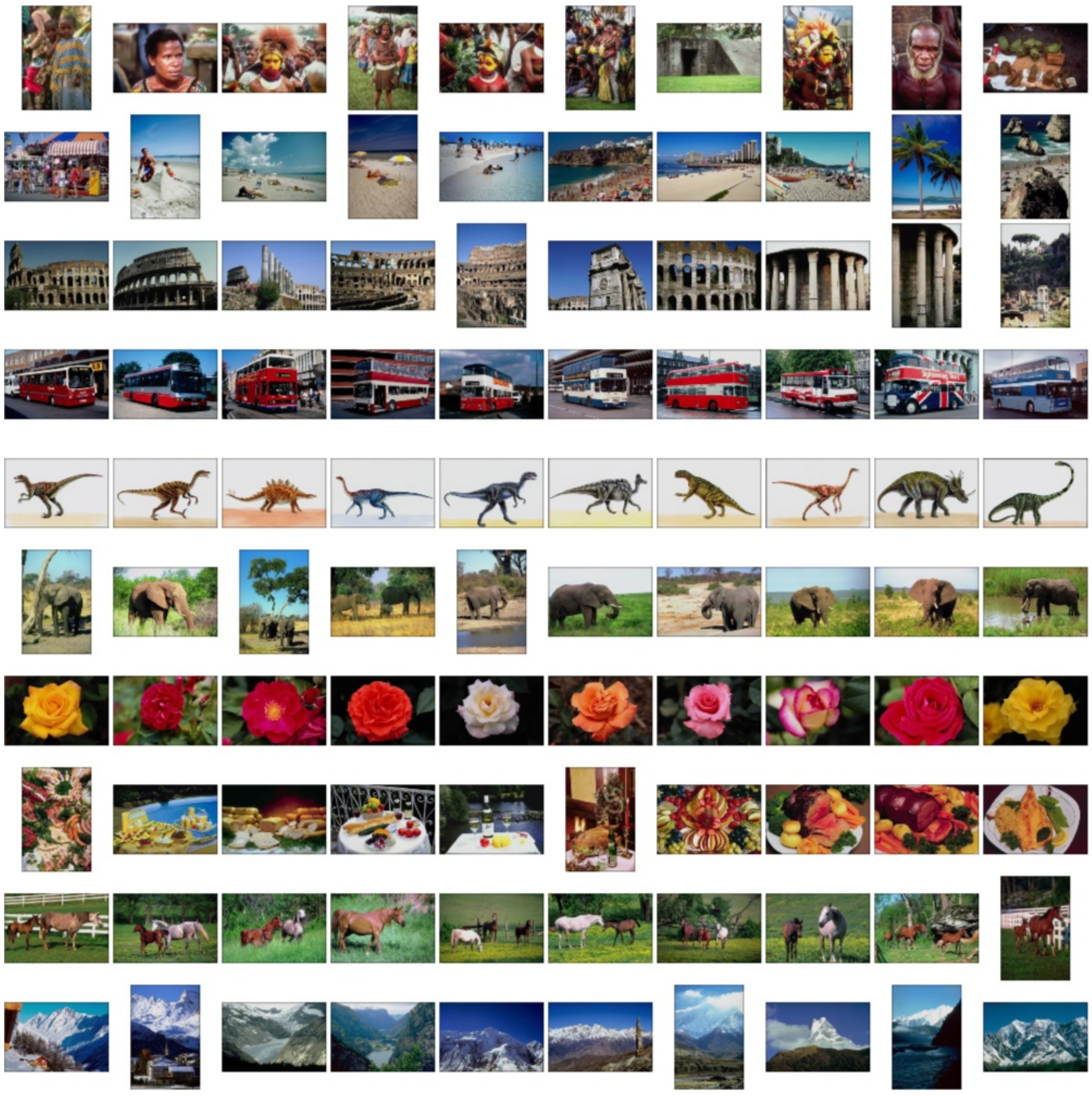

Figure 1: Examples from the object and scene category data set. Each row shows 10 images from a particular category. From top to bottom: Africa, beach, buildings, buses, dinosaurs, elephants, flowers, food, horses, and mountains.

local maximum), and orientation (the dominant orientation in the neighborhood of the point). Each image can contain a different number of interest points.

Note: You can use the VLFeat open source library (`http://www.vlfeat.org/`) for an implementation of the SIFT detector.

3.3 Describing local features

You will use two separate descriptors for each point:

- Gradient-based descriptors: compute the gradient-based descriptor of each local feature by using the descriptor step of the SIFT algorithm. Each point will have a SIFT descriptor of length 128.

- Color-based descriptors: compute a color histogram descriptor for each local feature. You should use the location, scale and orientation values computed in the previous step to form a scaled and rotated square around each point. Then, the color descriptor will be computed as a three-dimensional histogram of the RGB values of the pixels within the square neighborhood. You can use a $4 \times 4 \times 4$ histogram in the RGB space (i.e., each color channel is divided into 4 bins). The final one-dimensional descriptor can be obtained by *flattening* the three-dimensional histogram. Each point will have a color descriptor of length 64.

Note: You can use the VLFeat open source library (`http://www.vlfeat.org/`) for an implementation of the SIFT descriptor but the color-based descriptor step MUST BE your own implementation.

3.4 Finding visual words

The next step is the construction of the codebook of visual words. Here you can use the k-means clustering algorithm for quantizing the local descriptors. Typically, a large number of clusters (e.g, 500, 1000, 2000) has been used in the literature. You need to experiment with the number of clusters with respect to the number of local features obtained in the data.

In this step, you will compute three different codebooks:

- Using only the gradient-based descriptors (length of 128)

- Using only the color-based descriptors (length of 64)

- Using concatenated gradient-based and color-based descriptors (length of 192)

You must build two separate codebooks corresponding to two different sizes (i.e., two different values for number of clusters k) for each of the three options listed above. That is, you must have a total of six codebooks at the end of this step.

Note: Again, VLFeat and built-in MATLAB functions are good software resources for this step. If the set of all local descriptors that you extract does not fit into the memory of your computer (or if k-means is too slow), you may subsample them to run the k-means algorithm. Once you find the cluster centers, you can assign each local descriptor to the closest cluster center.

3.5 Building bag-of-words model

The next step is to compute the bag-of-words representation (i.e., histogram of visual words) for each image by using each of the six codebooks and the corresponding descriptors. At the end of this step, you will have six different bag-of-words representations for each image.

~~Optionally, you can ℓ_2 normalize each histogram by dividing the histogram by the norm of the histogram vector, in order to make image descriptors more robust against changes in the number of visual words (and some other factors). That is, for a bag-of-words histogram h, its ℓ_2-normalized version is given by $h/(\|h\| + 0.0001)$.~~

Note: This step MUST BE your own implementation.

3.6 Feature indexing and search

Now that you have feature representations of the images, you can perform a nearest-neighbor search among the feature vectors. The input of this step is a query image and its feature representation (i.e., a bag-of-words histogram). Given the feature representations of all images in the database, you rank them in ascending order of the distance between the feature representation of the query and the feature representation of each database image. You can use the Manhattan or Euclidean distance metrics to compute the distances between images. Since exhaustive search on large databases would be time-consuming, tree-based and hashing-based indexing methods are popular for this task. For this homework, you may use exhaustive search using arrays.

3.7 Evaluating retrieval performance

Evaluation measures for a retrieval system are used to assess how well the retrieval results satisfied the user's query intent. Precision at K (P@K) and Mean Average Precision (MAP) are two popular metrics. P@K is computed as the percentage of relevant images among the top K retrieved images, averaged over all queries. For this task, the images from the same category are considered relevant, and the images from other categories are considered irrelevant.

For evaluation you will split the data set into query and gallery images. Randomly selected 10 images from each category will serve as query images and the remaining 90 images from each category will serve as gallery images. In other words, there will be 100 query images, and the gallery for searching will consist of 900 images. For each query image, you will compute its distance to each image in the gallery and find the nearest K images in ranked order. You can compute all pairwise distances at once, or compute each pair as you go.

For quantitative evaluation, compute P@K scores for each query for K from 1 to 100. Then, take the average of these scores for all categories and for the whole gallery. You should plot the precision versus K (from 1 to 100) curve for the whole gallery to summarize the performance. Furthermore, prepare a table that consists of P@10 (for K = 10) scores averaged for each individual category and for the whole gallery (a total of 11 numbers).

You need to repeat this evaluation for each one of the six representations you have, and use these quantitative results in the discussion parts of your report. Plot the precision versus K curves for all representation on the same figure, and include the P@10 scores for all representations as separate rows of the same table.

Note: This step MUST BE your own implementation.

3.8 Discussion

You must discuss your implementation choices at each step and how these choices affect the final result in a report. This discussion must include example outputs for each step, such as local features detected for at least one image per category, example gradient-based and color-based descriptors (shown as bar plots) for several points from these images, example bag-of-words representations (also shown as bar plots) for these images, quantitative evaluation results (precision versus K curves and P@10 score table), and the final retrieval results showing several query images and corresponding top 10 retrieved images. The final results must include at least one query from each category. You must report how different descriptors and codebooks affect the final results, and discuss which categories are easier or more difficult to retrieve.

3.9 Software

You must provide well-documented code that implements all steps described above. For the steps that use external libraries, you must provide a README file that explains which libraries are needed and how they can be obtained and installed.

You must have a specific entry point (e.g., a function or script) to your solution to this homework assignment with the inputs listed below:

- A text file that lists a set of file names as the query images,
- A text file that lists a set of file names as the gallery images,
- A text file that provides the category id of each image in the query list,
- A text file that provides the category id of each image in the gallery list,
- An option for choosing a particular local feature descriptor (one of three options),
- An option for choosing the codebook size (one of two options for each descriptor).

The code for this entry point must be clearly described and documented in your solution. You are free to use any data structures and programming languages in your implementation. Note that we are going to test your code by using a separate image set that belongs to the same 10 categories.

Submit:

1. **A report (pdf file)** that contains the information requested above. You are also expected to provide a discussion of the results (e.g., which steps were easy and which were more difficult, what was possible and what was not).
2. **Well-documented code** for all steps that you implemented yourself. The specific entry point must be clearly indicated and documented.
3. Citations to all external resources that you used in your solution.

Make sure that all files you submit include your name and student ID.

Notes:
This assignment is due by midnight on Wednesday, November 20, 2019. You should upload your solutions as a **single archive file** that contains your **code** and **report** (a pdf file that contains the figures, descriptions of how you obtained them, and discussion of the results) using the online submission form on the course web page before the deadline. Please see the course syllabus for a discussion of the late homework policy as well as academic integrity. If you have any questions about what is allowed and what is not allowed in a solution, please check the course syllabus on the course web page.