University of Bucharest
Faculty of Mathematics and Computer Science
Department of Computer Science

Iuliana Georgescu
Bogdan Alexe
MSc in Computer Science
July 2021

Computer Vision - Project 3

Image retrieval using Bag of Visual Words

Objective

The goal of this project is to implement an image retrieval system. Your system should rank images from a database of images based on their similarity in content with a query image containing a specific landmark (building, monument, landscape, etc.). By retrieving the top ranked N images, the system should be able to find images containing the same landmark but with variations in viewpoint, scale, etc. The system that you need to implement will use the Bag of Visual Words (BOVW) model described in the course.

Data

The data for this assignment can be found at the following link: http://tinyurl.com/CV-2021-Project3.

The directory *database* contains images of 50 landmark classes, each class has 10 images of a specific landmark. Figure 1 shows all the 10 images for class 45. There are in total 500 images which form the database. You will use this database to retrieve images similar with a query image.

The directory *queries-validation* contains 50 *validation* query images, one query image for each landmark class. None of the query images is contained in the original database with

Figure 1: *Class 45 from the database contains* 10 *images with the same landmark.*

500 images. We will evaluate the performance of your system based on some *test* query images similar to these ones. The files in the *queries-validation/output* folder show how to format your output for an image query: create a txt file with the name given by the image name of the query and list all $N = 25$ retrived images from the database, listing each image name on one row. Ideally, for each query image, your system will return in the first 10 of the $N = 25$ retrieved images the ones from the corresponding class in the database.

The directory *queries-test* is currently empty. The test query images will be made available after the deadline for submitting the solution code. These images will be put in this folder.

The directory *logos* contains a logo and a video. The task here is to localize up to an image and bounding box the logo in the corresponding video.

Implementation using the BOVW model

Following the BOVW model, we represent each of the 500 images from the database and each query image as a normalized histogram of visual words. Image retrieval is reduced then to comparing the BOVW representations of the query image with the BOVW representations of the images from the database and retrieving the ones using a similarity measure.

Obtaining the visual words. The visual words form the visual vocabulary used for obtaining the BOVW representations of each image. They are obtained by clustering SIFT features[1] that describe the visual content of small regions (patches). These regions (features) have the property that they can be detected also in other images containg the same landmark, as the detector is robust to moderate changes in scale, viewpoint, scene illumination and color. For obtaining the descriptors you can use any interest point detector invariant to affine changes. For describing the visual content of the detected feature you can use the SIFT descriptor (other choices include SURF, ORB, etc.). The library *VLFEAT*[2] provides in this sense helpful code[3]. Usually, for each image from the database you will obtain around 3000-4000 SIFT descriptors. In total, there will be around 2 million SIFT descriptors for the 500 images from the database. For clustering you can use the *k*-means algorithm (other choices include mini batch k-means if the traditional *k*-means might require too much memory space), where the dimension k of the visual vocabulary should be chosen such that we have a good trade-off between speed and accuracy of the system. A small k value (for example 1000) would mean that each visual word is obtained as the average of 2000 descriptors. As in the database there are 10 images containing the same landmark most probable choosing $k = 1000$ might not lead to a good performance of the system, which will not be able to retrieve the most similar images in content. A recommended value of k is 10000 or 100000.

[1] see the article presented in the course: https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

[2] `http://www.vlfeat.org/`

[3] http://www.vlfeat.org/matlab/vl_covdet.html

BOVW representation. We use the k visual words to represent an image as a histogram of length k. We obtain the histogram asigning to each extracted SIFT descriptor the closest visual word, using the Euclidean distance. For comparing images with different number of visual words we normalize the histograms such that they have the l_2-norm equal to 1.

TF-IDF weighting of the visual words. A visual word is discriminative for a landmark if it appears many times in the images containing the landmark and in just a few images that do not contain the landmark. For a query image, detecting such visual words is a strong cue that it contains the searched landmark. So, instead of representing images just as simple normalized BOVW histograms by counting the appearance of each visual word, it is recommended to weight each visual word in a TF-IDF (term frequency - inverse document frequency) manner. According to this weighting scheme, each visual word will get the following weight:

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i},$$

where

- n_{id} represents the number of appearances of the i^{th} visual word in the image (document) d;
- n_d represents the number of visual words in the image d;
- N represents the number of total images in the database (in our case $N = 500$);
- n_i represents the number of images in which appears the i^{th} visual word.

The obtained tf-idf histograms should also be normalized.

Similarity measure. For searching similar images in content from a database based on a query image we use a similarity measure between the normalized tf-idf BOVW histogram of the query image and the normalized tf-idf histograms of the images from the database. As a similarity measure you can use the cosine similarity, given by the following formula:

$$sim_{cosinus}(h_{query}, h_i) = \frac{< h_{query}, h_i >}{\|h_{query}\|_2 \cdot \|h_i\|_2} =< h_{query}, h_i >,$$

where h_{query} is the histogram of the query and h_i is the histogram of the image i from the database.

Efficient computation. The searching time based on a query image in the database of 500 images mostly includes times corresponding to each of the following stages: (i) extracting descriptors from the query image; (ii) asigning to these descriptors the closest visual words based on computing the Euclidean distances between each descriptor and all visual words in the vocabulary; (iii) computing the tf-idf normalized BOVW histogram for the query image; (iv) computing similarities between the query tf-idf normalized BOVW histogram

Figure 2: *Results for image retrieval using the first image (the one from left) as a query image. The figure presents most similar (în the sense of the cosine similarity)* 25 *images from the database. Among the first* 10 *retrieved images only* 8 *contain the same landmark as the query image. However, among the first* 25 *retrieved images all the* 10 *images from the database are found. Images* 12 *and* 18 *are the other* 2 *images that contain the landmark*

and the 500 tf-idf normalized BOVW histograms which are precomputed for the images in the database. The searching time is dominated by the time needed to compute the Euclidian distances in order to determine the closest visual word from the visual dictionary to the current descriptor. The exact computations for an image with 3-4000 desciptors may take around a few minutes, as the complexity is linear in the number of descriptors, number of visual words from the dictionary and the dimensionality of the descriptor (128 for SIFT). For increasing the computational efficienty it is recommended using data structures such as kd-trees which offers a very good approximation in computing the nearest neighbors in the space of dimension 128 where the SIFT descriptors and visual words lie. The library *VLFEAT* provides functions[4][5] for efficient computations (under a second) of the nearest neighbors.

System's results. The results of the implemented system with all the above components (BOVW representation with $k = 100000$, SIFT descriptors extracted from regions detected by an afine invariant detector, the TF-IDF weighting, cosine similarity) are presented în Figure 2.

[4]http://www.vlfeat.org/matlab/vl_kdtreebuild.html
[5]http://www.vlfeat.org/matlab/vl_kdtreequery.html

Figure 3: *Results for image retrieval using the first image (the one from left) as a query image and geometric verification applied to the initial* 25 *retrieved images (from Figure 2). The obtained results are now perfect, the 10 images from the database are among the first 10 images based on the new score.*

Geometric verification. The system containing the components presented until now achieves a moderate performance in retrieval. We can increase the performance of the retrieval system by adding a new component, namely the geometric verification step. This step is used to filter wrong matches found between descriptors. Sometimes, regions that appear in images containing different landmarks are described by SIFT features assigned to the same visual word. These accidental matches can be removed by geometric verification. Based on some pairs of corresponding features of two images we can compute the affine transformation that would transform a region in the correponding region. Using the affine transformation we can find the number of inlier points (other corresponding features in the two images) that will follow the affine trasformation with some tolerance. In this way accidental matches can be removed. By using geometric verification as a rescoring method on the top of the first 25 retrieved images based on the old components of the system we obtained a better performance of the system. Figure 3 illustrates the results obtained by including this component.

Logo detection in video. Using as a starting point the image retrieval system you can detect a logo in a video. For example, for the Coca Cola logo run a modified version of your system on the provided video in the directory ”logo” that will allow detecting (Figure 2) up to a frame or a bounding box the logo in different frames for the provided video.

Figure 4: *Logo detection. The left logo is detected in one of the frames of the input video.*

Tasks

This project worths 6 points (as the other ones during the semester). In order to take all the points you have to do the following:

- implement correctly the image retrieval system that computes similarities between the query tf-idf normalized BOVW histogram and the 500 tf-idf normalized BOVW histograms from the database. This task worths **3 points**. In order to obtain the 3 points your system should achieve good recall performance of at least 50% for the test query images. We will measure recall as the number of images of the target class returned in the first $N = 25$ retrieved images. Ideally, your system will return all 10 images of the target class among the first 25 images for each test query image resulting thus in a recall equal to 100%. A recall performance equal to 40% will lead to obtaining only 2.5 points, a recall performance equal to 30% will lead to obtaining only 2 points and so on.
- correct implementation of the geometric verification component showing that your system performs better by rescoring the top 25 retrieved images - **1 point**;
- logo detection in the provided video (Coca-Cola) - **1.5 points** up to a frame/bounding box. You system should be able to output all the frames containing the Coca-Cola logo (1 point) and in each such frame localize the logo up to a bounding box (0.5 points);
- ex officio - **0.5 points**.

In order to get your grade you will have to submit your code and results on the test query images and also present online your project. The recommended day for presenting the project is Monday, 26^{th} of July, but you can also present later (until Sunday, 1^{st} of August). When sending your code please specify when you would like to present your project.

Deadlines: Send your zip archive containing only your code until Saturday, 24^{th} of July to the email address `bogdan.alexe@fmi.unibuc.ro`. On Sunday, 25^{th} of July, we will make available the query test images. There will be 2 images for each class, so in total you will have 100 test query images. You will have to run your system on the test images

provided by us and send your results (the first 25 retrieved images for each query, following the format in the release, see the folder *queries-validation/output*) in the same day to the same email address.

If you have any questions regarding the project please send an email to `bogdan.alexe@fmi.unibuc.ro`.