Content-Based Image Retrieval

1 Practical work presentation

1.1 Goal

The goal of this practical work (PW) is to apply local image descriptors in order to compare images. The application consists in querying a database with a query image and retrieve the most similar images in the database.

1.2 Data and working environment

1.2.1 Image databases

Different image databases are proposed which exhibit different properties.

- **base1** : 89 views of Monceau park and Amsterdam
- **Flickr** : 100 images collected from Flickr
- **NISTER** : 400 images extracted from the NISTER database. This collection consists of images of objects, each taken in 4 slightly different views. There are then 100 images of different objects
- **COREL** : 400 pictures in total composed of 50 images from the COREL database, each having undergone 7 transformations (see Fig. 1.2.1) :
  1. JPEG compression with quality factor 80%
  2. median filtering of size 9x9
  3. noise addition
  4. crop and rotation 15°
  5. crop and rotation 30°
  6. scale change and rotation 15°
  7. scale change and rotation 30°

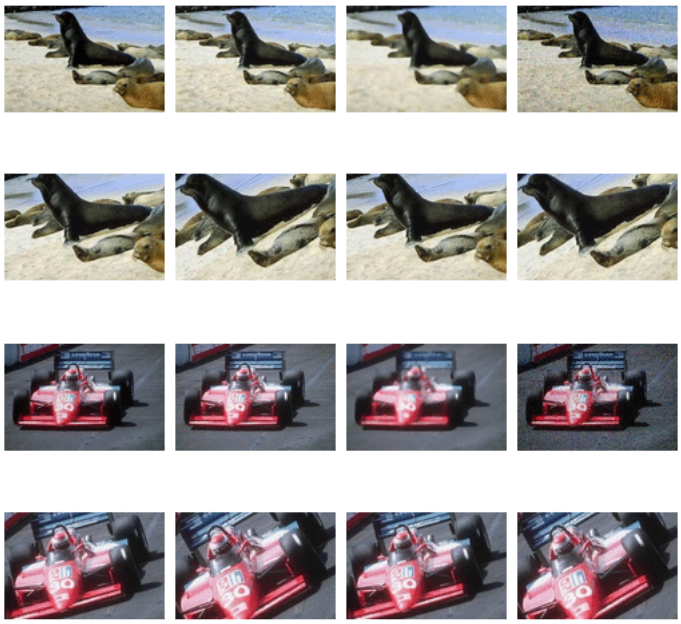

Figure 1: Example of an original image and its 7 transformations

1.2.2 Images

All the related images are in the archive: Images.zip which contains one sub-directory per database.[1]
For each database, is given:

- a corresponding directory containing all images
- the names of all the images listed in a text file: _liste_baseName
- a thumbnail : `index_baseName.gif`
- for the COREL and NISTER databases, the list of query images (which are not part of the database) : _liste_baseName_queries.txt

1.2.3 Source code

In the archive project.zip, there are 2 python scripts.

db_indexing.py: Given a database, it extracts the local descriptors for all images and index these descriptors.

query_search.py: Given a database and a query image, it searches the (indexed) database to retrieve the most similar images.

[1] Do not change the organisation, nor the names for the good working of the given programs

2 Experiments

In a first part, we will work with NISTER and COREL databases.

➔ Before running the programs, don't forget to set the correct path for variable img_dir in db_indexing.py (l. 29) and query_search.py (l.42).

2.1 Understanding the data

2.1.1 Local descriptor manipulation

1. Use the scripts feature_description.py and matcher.py to visualize the descriptors and matches.
   ✍ Explain which information can be visualized thanks to the images produced by the scripts.
   ✍ Which characteristics of an image influence the number of descriptors?

2. Visualize the matches between 2 images of a similar object, and between 2 images of a different object (NISTER database).
   ✍ What do you notice?

3. Similarly, visualize the matches between an original image from COREL database and each of its transformed versions.
   ✍ What are the most difficult transformations? Why?

2.1.2 Indexing the database

Be sure to understand what's going on, and what are the important parameters.

1. which local descriptors are used? what are their dimension?

2. how many descriptors in the database?

3. which indexing algorithm is used? exhaustive, sequential search? approximate search?

2.1.3 Querying the database

1. What is the matching criterion **between descriptors** used in this program? Explain its principle.

2. Explain how the matching between descriptors is turned into a score associated to each image of the database. What represent the score associated to each image?

3. Test the copy detection program on different query (original) images of the COREL database. By analyzing the query_name_ranked_list.txt output file:
   ✍ What are the most difficult transformations? Why?
   ✍ Is that coherent with the observations you made in the section 2.1.1?

2.2 Evaluation

To evaluate a system, a ground truth is needed. This is possible for NISTER and COREL databases using the image names (one image and its transformed versions share a same name).

➔ Complete the code in order to compute the recall and precision values. (l. 170-180)

➔ Complete the code in order to compute and display the Average Precision.

2.3 Analyzing the results

➔ do NOT hesitate to modify the given scripts. A lot of improvements can be made: define functions, call them in the right order, add parameters to the command line for shell scripts calls, add a loop on queries, etc. Just be careful to always keep a running version.

2.3.1 Comparisons

Perform a search with SIFT with 2 queries per base for **NISTER** and **COREL** (except the first two, randomly picked).

- Compare the results obtained with different local descriptors.
- Compare the results obtained with linear search or approximate search.
  ✍ Comment the results: quality, AP value, computation time in terms of indexing/search,...
  (➔ Note that the time to load descriptors and index is not important here !)
- Try different parameters (`knn` values) and see their influence.

Then pick 2 queries per base for **base1** and **Flickr**, and see the results.

2.3.2 Report

You will report the answers to the previous questions, as well as the results and analysis of your runs in a report. Be concise and precise. For each experiment, should be indicated:

- the query image (name, database, picture)
- which descriptors are used (and with which parameters, if applies)
- the indexing method, and its parameters (when applies)
- the results: image of top ranked images and their scores, recall-precision curve (at least for the best run), AP value, computation time in terms of indexing/search

Note that quantitative comparisons should be reported in a table, indicating all the useful information and parameters, to facilitate the analysis.

Conclude on which setup seems the most interesting regarding the datasets and the task, the behavior of the system (stability over different queries/databases, speed).

2.4 Comparing with global descriptors

- Adapt (simplify!) the code to run the search with a global descriptor (greylevel and color histograms for example).
- Try it with the same previously used queries, and compare the results with local description (quality, computation time).

2.5 Move to larger scale

Test with the COPYDAYS database.

- How many images? How many transformations per image?
- How many descriptors in the database?
- Comment the results (quality, computation time in terms of indexing/search,...).

➔ Send your report (`your_name.pdf`) by mail to: `ekijak@irisa.fr`, with subject: "[Module Name] Homework" .

Note that not respecting the instructions above (presentation, format and name of the file, etc) will be penalized.