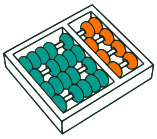

MC949/MO446 — Introduction to Computer Vision
INSTITUTO DE COMPUTAÇÃO — UNICAMP
2nd Semester, 2017

UNICAMP

Project 4

Prof. Adín Ramírez Rivera
adin@ic.unicamp.br

1 Description

In this project you will develop a content based image retrieval (CBIR) functions (system is a too wide word for this project). The goal of CBIR systems is to find the closest matches within a set of images given a query image.

The main idea is to create a descriptor that represents the characteristics of each image such that it can generalize the properties of the class, but can be different enough from the other classes. In this project you will work with regions obtained from the image, and will extract features from the regions. Then, you will develop a distance function that will obtain the similarity of these descriptors.

2 Data

The data for this project is given in a zip file. It consists of 40 images of 8 different categories.

3 Image Descriptor

For each image you need to compute the regions of the image. To do so, you will need to compute the K-means (or other clustering algorithm) on the color space of the image. And obtain K salient regions on the image.

Then you need to label the pixels on the image with their respective cluster and execute a connected components to extract the segmented image. ✎ Show examples of the clustering and the segmentation of selected images in your report.

You can improve the regions by removing noise from the regions, or by smoothing the region boundaries, or by filling the holes in the components. ✎ Explain in your report what you did, in case you decided to improve the regions.

Then, for each region you need to extract information of it and encode it in a data structure. Consider at least:

- Size of the region
- Mean color (in whatever color space you are working with)
- Co-occurrence texture features[1] (within a spatial relation, you can use any type of neighborhoods) like
  - Contrast
  - Correlation
  - Entropy
- Centroid of the region
- Bounding box (or other location representation)

You will need to create a data structure to store such information for access later on. We will call this structure the descriptor of the region, R, and the feature descriptor of the image is a set of region descriptors, such as

$$F = \{R_i\}_{0<i<K}. \tag{1}$$

Consider storing the processed descriptors, so you don't have to re-compute them every time you experiment with the distances (see § 4).

[1] https://en.wikipedia.org/wiki/Image_texture

4 Distance Measure

Once you processed all the images and stored all the image descriptors, you will need to develop a distance measure between two descriptors. That is you need to create a function $D : \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{R}$, where $\mathbb{F}$ is the space of the image features defined in § 3, and $\mathbb{R}$ are the real numbers. In other words, your function takes two feature descriptors and returns a score of their similarity or dissimilarity. I recommend to take the dissimilarity approach, as you can compute differences in the attributes instead of normalizing scores afterwards (see the following for more details). That is, if they are similar they will have a score close to zero, and if they are dissimilar will have a high score. ✎ Regardless of your choice, explain in your report your design decisions and your support for your distance.

To do this, you need to compute the correspondence greedily. That is, given two feature descriptors F_1 and F_2, for each region $R_i \in F_1$ you need to find the closest match $R_i^* \in F_2$. Note that the mapping need not to be one-to-one. You can do a full search for all the matches between the images. The final value of D should be a function of

- the difference in attributes of the corresponding regions, and
- the difference in the number of regions (in case you are running a clustering algorithm that returns different regions).

Experiment with at least two different distance measures and ✎ explain what they are and which one works better and why. Get creative with what and how you want to compare.

5 Experiments

Evaluate your method on the data provided (see § 2). Perform several experiments tweaking your parameters and evaluating different configurations, specially for your distances (in case they require some parameters) and for the clustering algorithm (in case you use K-means test different values of K).

✎ Show the top 3 results of querying the following images:

- `beach_2`
- `boat_5`
- `cherry_3`
- `pond_2`
- `stHelens_2`
- `sunset1_2`
- `sunset2_2`

Repeat this process for your different distance measures. Explain your results (see the requirements of § 4).

Note that when you are doing the search, you are searching against the whole database (40 images in our case).

6 Extras

If you are eager to do more and bring your method to a next level try the following:

- Use better features (up to +5% depending on proposal and results).
- Use adaptive distances and features (up to +5% depending on proposal and results).

Both extras should be demonstrated in the experiments and in the report to get the extra credit.

7 Evaluation

Your grade will be defined by the following aspects:

1. Feature descriptors 40%
2. Distances 40%
3. Overall report 20%

Each item corresponds to the questions and requirements defined in the previous sections. The overall report, point 3, refers to the evaluation of the details presented in your report, your way of presenting results, explanation and use of concepts and theory, references, etc. **Your English usage won't be graded**, but your ability to present your results, ideas, and how the are supported will be. Each other point will be evaluated according to the completeness and correctness of the requested items.

You will be eligible for extra points, defined in § 6, if you completed the required assignment, and you have an explanation of your extra development in your report.

8 Submission

You need to create a folder named `p4-XX-YY` where `XX` and `YY` are the RAs of the members of your team. Note that the RAs **must** be sorted.

Your submission must have the following subfolders:

- `input`: a directory containing the input assets (images, videos or other data) supplied with the project.
- `output`: a directory where your application should produce all the generated files (otherwise stated in the problem). This directory should be empty.
- `src`: a directory containing all your source code. You only need to submit files that are not derived from other files or through compilation. In case some processing is needed, prefer to submit a script that does that instead of submitting the files.
- `Makefile`: a makefile that executes your code through the docker image. An image is already built and available for use (`adnrv/opencv` at the docker hub registry). The code will be executed through a standard call to `make`, so other dependencies must be provided by you under that constraints.
- `report.pdf`: a PDF file that shows all your work for the given project, including images and other outputs needed to explain and convey your work. When needed include explanations to the questions given in the project.

The principal folder must be zipped into `p4-XX-YY.zip` and submitted through the Moodle website. No other files will be accepted. **Note that upon unzipping your file, the original folder** `p4-XX-YY.zip` **must be created.** That is, do not zip the contents, but rather the folder.

9 Notes

- Note that there are several implementations that you can find in the internet. This project is for **you to implement** the algorithms. Thus, do not submit code from others. And if you re-use code from someone for a non-restricted part, disclose it in your report and code.
- There **are no** restrictions of functions that can be used from OpenCV or other libraries, **as long as they do not solve the CBIR problem for you**.
- All the submissions must be self contained and must be executable in a Linux environment. Specifically, your code must execute in the docker image `adnrv/opencv`, available at docker hub (https://hub.docker.com/r/adnrv/opencv/).

- It is your responsibility to make sure your code compiles and executes correctly. No effort will be made to run your code besides executing `make` inside a docker.
- You must program in Python 3.6 (or higher) or in C/C++ with `gcc` version 6.2.0, using OpenCV 3.2.0. All available within the image. If you need to install other packages you must do so within your `Makefile` as automatic prerequisites.
- Check the `Makefile` provided with the project zero. It will be used to automatically execute your code. You must pass the full path of your project folder: `make SOURCE_DIR=$(realpath ./pA-XX-YY)`.