Running and Configuring the Baseline System of ImageCLEF 2013's Personal Photo Retrieval Subtask

Prerequisites

The following document assumes that you have followed the build instruction published on `www.imageclef.org/2013/photo/retrieval`.

We assume that you have created a subdirectory below the extracted sources called `build/`, which we will refer to as `builddir` in the following text.

After running "`make install`", you will end up with a directory structure as follows. The most important subdirectories are set bold.

builddir	**applications** (contains all executables)
	dbis (build files, can be ignored)
	include (header files if you want to compile against the system)
	libs (the dynamic libraries of the system)
	plugins (plugins providing feature extractors etc.)
	tests (various unit test, can be ignored)

If you are interested in the full documentation of the system, please use `doxygen` with the distributed Doxyfile to create it.

In order to run the system, you have to make the dynamic libraries known to your OS, e.g., by adding `builddir/libs` to `DYLD_LIBRARY_PATH` in case of Mac OS X. Additionally, you will have to create a new environment variable called `PYHTIA_PLUGIN_PATH` that has to point to `builddir/plugins`, which can be achieved as follows:

```
export PYTHIA_PLUGIN_PATH path/to/builddir/plugins/
```

Preparations

In this step, you will save the image collection and the query images in a structure that can be processed with the system.

Copy the test data set to a directory that we will refer to as `imagedir` from now on. Create a subdirectory called `queries` below `imagedir` in which you will store all query images. Your resulting file hierarchy should look as follows:

imagedir	file01.jpg file02.png file03.png ...	
	queries	query01.png
		query02.jpg
		...

Extracting the Features

In order to retrieve images, you will have to extract their low-level features first. The tool to extract features is called `extractFeatures` and can be found in `builddir/applications/`. Navigate to this directory.
Before you can use the feature extraction, you will have to create a configuration file by starting `extractFeatures` with the parameter `k`:

```
./extractFeatures -k
```

You will see the following output and a file named `configFE.cfg` will be created in your current directory.

```
Create configure file...
Done.
```

The configuration file `configFE.cfg` is pretty self-explanatory but the following parameters need your special attention:

`IMAGE_DIR`	Enter the path to imagedir ending with the path separator, e.g., `/path/to/imagedir/` if you use Un*x
`COLLECTIONS_DIR`	You can leave this directory unmodified. The variable contains the path that will be used for storing all extracted features and metadata. That is, a file called `collections.xml`, which you will need later.
`COLLECTION_NAME`	Choose an arbitrary collection name, e.g., "iclef2013".
`COLLECTION_ID`	Enter a UUID. We recommend to change the first digit into a 1 because you will need this UUID later and this UUID is easy to remember.
`PLUGIN_DIR`	If you have properly set the environment variable `PYTHIA_PLUGIN_PATH`, this variable should point to `builddir/plugins`. Otherwise you will have to change it manually and set up the enivironment variable.
`OUTPUT_TYPES`	This variable should only contain `dbis_xml`. If you leave it unmodified the extraction will consume more space and create additional XML files that more or less resemble MPEG-7. For the remainder, we will only use the `dbis_xml` format.
`FEATURES`	A list of features that will be extracted.

The actual contents of `FEATURES` will depend on your setup, e.g., if you have installed OpenCV before. Remove all features from the list that you do not want to extract.

Please note that the local features, e.g., SURF or BRIEF, take very long to process. The same holds true for the face detection. We recommend removing all local features and the face detection because the latter is not fully functional yet.

The resulting set of global features looks as follows:

```
AutoColorCorrelogram:NoDetector
BIC:NoDetector
CEDD:NoDetector
ColorHistBorder:NoDetector
ColorHistCenter:NoDetector
ColorHistogram:NoDetector
ColorLayout:NoDetector
ColorStructure:NoDetector
ContourShape:NoDetector
```

```
DominantColor:NoDetector
EdgeHistogram:NoDetector
FCTH:NoDetector
Gabor:NoDetector
RegionShape:NoDetector
ScalableColor:NoDetector
Tamura:NoDetector
```

The extractor parameters are more or less self-explanatory given you have access to the relevant papers.

After you have adjusted the configuration, you can simply start the extraction that will automatically use your altered `configFE.cfg`:

```
./extractFeatures
```

If you have not changed the collections' directory in the configuration file, you will find a new folder below the executable's path called `collections/`, which contains the actual extracted feature data for your collection and the metadata file `collections.xml`.

Calculating the Similarity between Query Images and the Collection

To calculate the similarity of each document in the collection with a set of query documents, you will have to use a tool called `calcSimilarities` that has *a lot* of parameters. We will explain the parameters below. If you need further information refer to the online help by calling `calcSimilarites --help`.

calcSimilarities, its Parameters and Utilities

```
./calcSimilarities -i {10000000-0000-0000-0000-000000000000} -
k 100 -t ./types.lst -q ./catmap/ -o 0 -e 0 -f 0 -x 1 -m
-c ./collections/collections.xml -r ./runs/
```

-i: Specifies the collection that you want to work on. You have to use the UUID that you have defined in the step before.

-k: Defines the top-k values of the results. In the example, each result file will contain 100 documents.

-t: Specifies the path to a text file that defines the retrieval types to be used. Roughly speaking, a retrieval type is a combination of a feature and a distance/similarity function.

There is a utility that will create a valid retrieval type file for you: `generateTypes`. Call it as follows and choose option `d` to get the standard set:

```
./generateTypes
```

```
Retrieval types (r), feature types (f) or default retrieval
types (d) ?
d
The types have been written to .../types.lst.
```

For our example we will modify `types.lst` and remove the local features. The result will look like this:

```
AutoColorCorrelogram:NoDetector_Manhattan:NoDistance:NoDistance
BIC:NoDetector_Euclidean:NoDistance:NoDistance
CEDD:NoDetector_Euclidean:NoDistance:NoDistance
ColorHistBorder:NoDetector_Euclidean:NoDistance:NoDistance
ColorHistCenter:NoDetector_Euclidean:NoDistance:NoDistance
ColorHistogram:NoDetector_Manhattan:NoDistance:NoDistance
ColorLayout:NoDetector_ColorLayoutSpecific:NoDistance:NoDistance
ColorStructure:NoDetector_Manhattan:NoDistance:NoDistance
ContourShape:NoDetector_ContourShapeSpecific:NoDistance:NoDistance
DominantColor:NoDetector_Euclidean:EMD:NoDistance
EdgeHistogram:NoDetector_EdgeHistogramSpecific:NoDistance:NoDistance
FCTH:NoDetector_Manhattan:NoDistance:NoDistance
Gabor:NoDetector_Euclidean:NoDistance:NoDistance
RegionShape:NoDetector_Manhattan:NoDistance:NoDistance
ScalableColor:NoDetector_Manhattan:NoDistance:NoDistance
Tamura:NoDetector_Manhattan:NoDistance:NoDistance
```

-q: Specifies the path to the catmap files.

This directory is expected to contain text files that are named after their topic ID. For instance, `1.txt` will contain all query images for topic #1. Each line of the files will contain one query image path relative to `imagedir`.

A sample file should look as follows. Although the first line has to be in the file, we can neglect its meaning for the scope of this document.

```
#Collection: abc (463); Category: 1
query/query01.png
```

-o; -e; -f; -x; -m: Can be ignored.

-c: The path to the `collection.xml` file. This file contains the metadata to find all extracted features etc. during the similarity calculations.

-r: Specifies the path to the directory in which all evaluation results in text form (file ending `.treceval`) will be stored. These files can be used directly with `trec_eval` to calculate various effectiveness metrics (`http://trec.nist.gov/trec_eval/`).

Two sample lines of a `.treceval` file:

```
Q1   Q0   _query/002_0001.jpg 1     1    AVG  QBE:
_query/002_0001.jpg
     Tamura:NoDetector_Manhattan:NoDistance:NoDistance=1
     DominantColor:NoDetector_Euclidean:EMD:NoDistance=1
```

```
Q101 Q0   P1000512.JPG   2    0.832689  AVG  QBE:
_query/002_0001.jpg
     Tamura:NoDetector_Manhattan:NoDistance:NoDistance=0.989578
     DominantColor:NoDetector_Euclidean:EMD:NoDistance=0.798366
```

The fields are separated by tabs with the following semantics (from left to right):

1. The topic ID
2. Can be ignored
3. Retrieved document's path (relative to `imagedir/`)
4. Document's rank
5. Document's similarity score with respect tot he query
6. Aggregation method (can be ignored, AVG stands for the arithmetic average of all feature similarities)
7. The query document
8. Similarity score with respect to the query using the specified feature...

The similarity scores are in [0;1]. Please note that these files will also contain the query images that you will have to remove manually. You see this in the first line of the example: column 3 equals column 7 and the similarity score is 1.0, i.e., a perfect match.

If you are short on hard-disk space you might want to add `-z` to enable compression of the result files.