Example: Automated test

This is a step to step tutorial with screenshots on how to perform an automatic query with a given annotated database.

In this example we use the Corel database and perform a query with random query images from the same database. The descriptor CEDD is used.

Table of Contents

1. Before using simple-cbir

1.1. Download Corel database

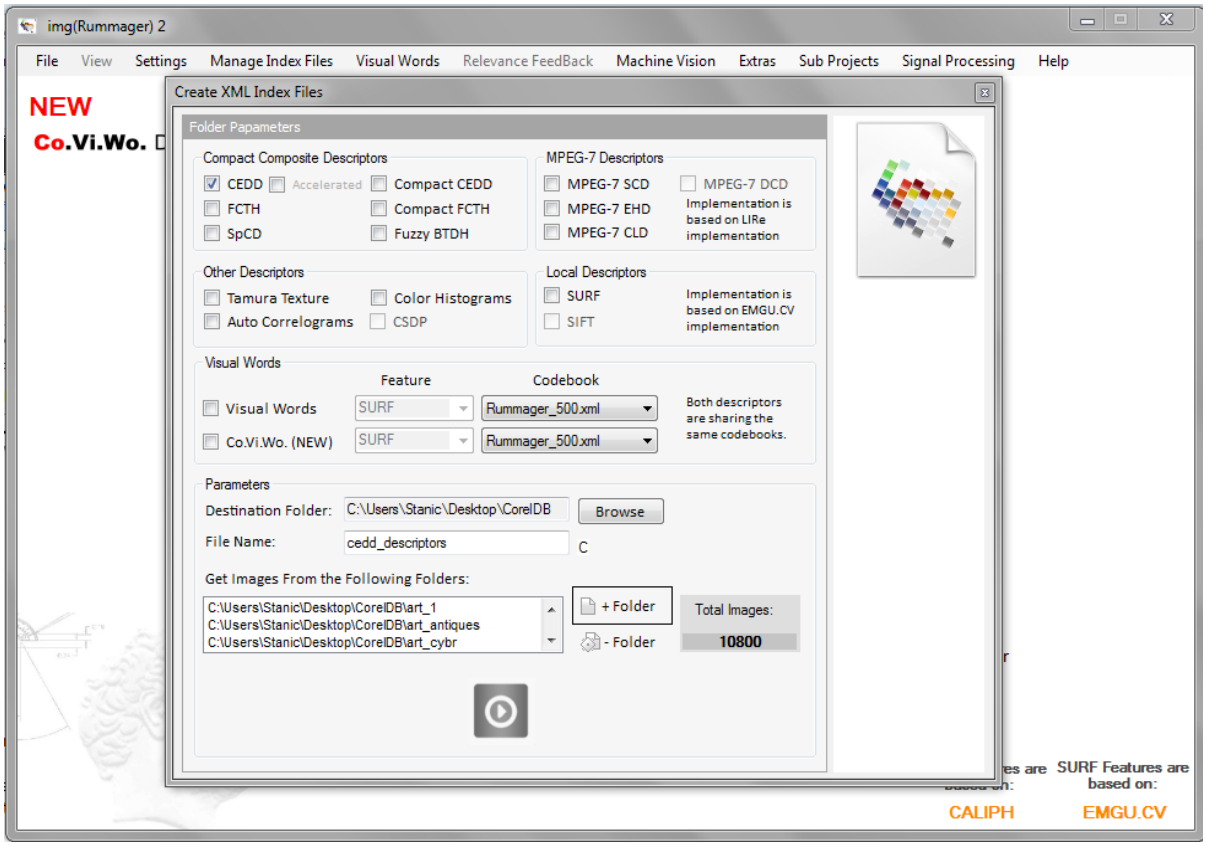
Download the Corel database here: https://sites.google.com/site/dctresearch/Home/content-based-image-retrieval Download the 7 Parts and unpack them. Save the database in the folder of your choice.

1.2. Extract descriptors via img(Rummager)

Now it's time to extract the descriptors from the images of the database. For extracting CEDD we use the tool *img(rummager)*. Download it here:http://chatzichristofis.info/?page_id=213 and install it if you haven't already.

Run *img(rummager)*, and select "Manage Index Files > Create Index Files" in the menu bar.

Choose CEDD as descriptor, then specify the destination folder and name for the descriptor file, ideally this is the database folder (CorelDB). Also choose the folders from which you want to compute the descriptor. In this case, these are all the the subfolders of CorelDB. There should be 10800 Images in total. Finally hit the play button on the bottom and wait until the tool finishes the work.



2. Adjusting settings

Now that the preparation is done, it is time to execute *simple-cbir*. If you haven't downloaded our code yet, download it via git: `git clone https://code.google.com/p/simple-cbir/`

Before you can execute the program you have to specify the correct settings in *src/cbir/AutomatedRFTest.java*. So open*src/cbir/AutomatedRFTest.java* with an (java) editor of your choice.

Specify following settings:

There are three settings which are important, otherwise the program may not execute.

- *output*: you have to specify the html output file path. In our case this is "C:\\Users\\Stanic\\Desktop\\CorelDB\\automatic_corel_cedd.html", but you can take the path of your choice.
- *type*: as we have extracted CEDD descriptors it's important to specify it as the descriptor type. So set type as `DescriptorType.CEDD`
- *xml path*: here you have to specify the path to the XML descriptor file created from *mg(rummager)*. In our case this would be: "C:\\Users\\Stanic\\Desktop\\CorelDB\\cedd_descriptors.xml"

```
public class AutomatedRFTest {
    /** The path of the output-file can be specified here. **/
    public static String outputfile = "C:\\Users\\Stanic\\Desktop\\CorelDB\\automatic_corel_cedd.html";
    /** The number of random queries that should be performed. **/
    public static int numOfQueries = 5;
    /** The number of RF iterations is specified here. **/
    public static int numOfRFIterations = 4;
    /** The number of results that are considered. **/
    public static int numOfResults = 20;
    /**
     * This boolean flag denotes if new random indices should be generated or
     * not.
     **/
    public static boolean newRandoms = true;
    /**
     * If newRandoms == false you can specify a file where the randoms should be
     * read from. Else the produced randoms will get stored in this file.
     **/
    public static String randomIndicesFile = "C:\\MBOX\\results\\corel\\randomindices50.txt";
    /** The metric that is used for the ranking. **/
    public static Metric metric = new WeightedEuclidean();
    /** The descriptor-type of interest for the ranking. **/
    public static DescriptorType type = DescriptorType.CEDD;
    /**
     * The normalization that is used on the descriptor of interest, this is
     * important for the merged descriptor!
     **/
    public static Normalization normalization = Normalization.GAUSSIAN;
    /** The path of the xml file containing the descriptors for your database. **/
    public static File xml_path = new File(
            "C:\\Users\\Stanic\\Desktop\\CorelDB\\cedd_descriptors.xml");
    /** Indicates whether you want to use indexing or not. **/
    public static boolean useIndexing = false;
    /**
     * The relevance feedback method that should be used has to be specified
     * here.
     **/
    public static RelevanceFeedback rf = new Bayesian();
```

The other parameters can be left as they are. Of course you can play around with the settings.

3. Executing the program

3.1. Compiling with *ant* and executing

```
1. download ant
2. build with ant:
  ant
3. run the AutomatedRFTest:
  java -classpath "./external;./bin" cbir.AutomatedRFTest
```