Content-based retrieval of visual information

Oerlemans, A.A.J.

Appendix A

RetrievalLab

A.1 Introduction

RetrievalLab is a tool for illuminating content based retrieval. It can be used in research and in educational workshops to explore, compare, and demonstrate the use of features, databases, images and evaluation methods in content based retrieval tasks. Regarding education, the intention is that students will be able to learn about content based retrieval without spending numerous months creating a custom system. In addition, RetrievalLab has a plugin architecture that makes it possible for users to add new functionality.

A.2 Related work

There has been significant prior work in content based retrieval (CBR) systems. See Datta et al. [12] for a recent survey. A few examples of such systems include the the GNU Image Finder and imgSeek.

RetrievalLab was inspired by Matlab [15], except that the native data structures and functionality are specifically designed toward facilitating content based retrieval (CBR) research. For example, in retrieval contexts, we have the notion of a database of multimedia objects which is common to most CBR systems. The typical usage in Matlab would be to load the pictorial, feature, and tag information into separate arrays. In RetrievalLab, there is the fundamental notion of a database object so the user can issue a command like

```
MyDatabaseObject = loaddatabase("MyImageDirectory")
```

More information will be given in Section A.3, but the fundamental notion is that databases are now native objects which can be updated, copied, manipulated, and analyzed or used as sets of positive and negative examples in machine learning.

RetrievalLab provides the following:

- Programming interface - Matlab-like algebraic/functional.
- Database data structure is a fundamental and native object.
- Free (unlike Matlab or Mathematica)
- User extendable plug-in architecture with sample plug-ins.
- Basic functionality for all research stages: loading databases, feature extraction, machine learning, visualization of results, and quantitative benchmarking and evaluation.

While there are many research systems which provide some of the items above, we are currently not aware of any other system which provides all of them.

A.3 Example usage

This section will demonstrate a few uses of the RetrievalLab program, by showing the commands that are needed to accomplish certain tasks. A more extensive description is available at http://press.liacs.nl/researchdownloads/retrievallab/.

A.3.1 Image retrieval

In a typical image retrieval task, a database is loaded into a variable and both ground truth tags and features are loaded into memory. We added support for loading ground truth for the MIRFLICKR [31] [32] and IMAGECLEF [55] datasets.

```
> db=loaddatabase("D:/MIR-Flickr/")
> loadmirflickrtags(db, "D:/tree_r1.txt")
> loadfeature(db, "hsv")
```

After that, an image is loaded and the same feature as was used for the database is calculated.

```
> im=loadimage("D:/Trees/tree1.jpg")
> updatefeature(im, "hsv")
```

With the database and the image, a search query can be executed that results in an index.

```
> index=searchimage(db, im)
> displayindex(index)
```

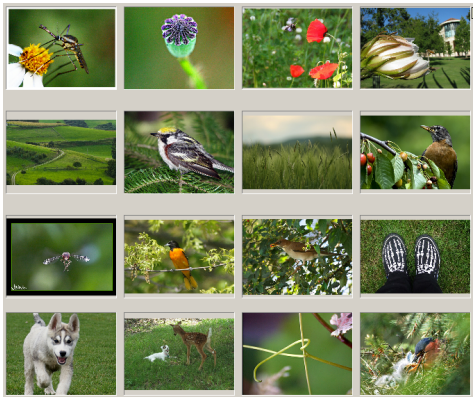

Figure A.1: Result of the 'displayindex' function.

Results can be displayed in the form of a standard grid, or a 2D view with results centered around the best matching image, where the feature distance is used to determine the distance in the image.

```
> displayindexmap(index)
```

Figure A.2: Result of the 'displayindexmap' function.

Finally, we evaluate the results. In this case, we evaluate the tree_r1 tag using:

```
> evaluateindex(index, "tree_r1")
```

which returns the MAP (mean average precision) value with respect to the ground truth.

A.3.2 Visual concept detection

In a typical visual concept detection query, two databases with positive and negative examples are loaded into variables and after that, the images are segmented and features are calculated. With these two databases, a visual concept can be created based on the selected classifier.

First, the positive database is loaded and the enhanced wavelet representation feature [57] is added to the image segments.

```
> dbpos=loaddatabase("D:/Trees/")
> segmentimage(dbpos, "sift")
> updatefeature(dbpos, "hsv")
> updatefeature(dbpos, "lbp")
> updatefeature(dbpos, "wavelet")
```

Note that each image segment now has three different features attached to it. All three features will be used. For the negative examples, we do the same. (Segmenting and feature extraction is omitted for brevity.)

```
> dbneg=loaddatabase("D:/NotTrees/")
```

The concept can then be learned with a selected classifier. Currently, nearest neighbor, SVM and neural network classifiers are supported.

```
> concept=learnvisualconcept(dbpos, dbneg, "svm")
```

After this an image is loaded and segments and features are added. We can now apply our visual concept to this image.

```
> findvisualconceptlocations(im, concept, "tree")
> displayimage(im)
```

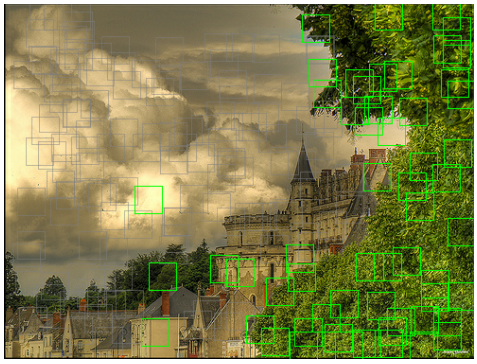

Figure A.3: Result of detecting a concept.

A.4 Discussion, conclusions and future work

The current system gives a programming interface to content based retrieval functionality, which is both user extendable and focuses on the typical CBR components including diverse features [12] [74] [80], distance measures [76], and classifiers [76]. We think that having a database as a native object facilitates many typical database operations. In addition, RetrievalLab has built-in support for the MIRFLICKR [31] [32], IMAGECLEF [55] test sets.

In future work, we will add plug-ins for video database analysis. Note that our framework was designed for generality regarding media types and should be able to accommodate most kinds of media.