Project – ICV

Content Based Image Retrieval (CBIR) with Wavelet features

Duration: 3 weeks
Start date: 12th March 2018
End Date: 9nd April 2018, 5PM (after Easter vac)

Description

CBIR systems take in a *query image* and try to find the most 'similar' images in a databases of images. Similar usually means semantically similar i.e. a query of an animal on a field should return only images of animals on fields. This is surprisingly difficult. Most modern approaches are based on complicated Deep Learning systems, where the internet (through image collections like Flickr and Facebook) is used to provide a huge number of training images. An alternative is to use an explicit feature-based approach: rather than learning a feature/description for an image, as done by a machine learning system, we compute a descriptive feature using some mathematical process.

Your task is to implement and evaluate a CBIR system that uses the Discrete Wavelet Transform to generate its core image feature representation. This approach generates a wavelet descriptor for each image in the database and then compares the wavelet descriptor of a query image to find the set of images in the database that are most similar. In the simplest case, one can use a Euclidean distance metric, where you difference each corresponding element in the feature vectors, square them and add them all up. Images that are similar should have smaller Euclidean differences. You will implement a system based on [1]; this describes the basics of the implementation.

You must read the paper (which is one the reading group papers we will analyse) and implement the basic system. You **must** use OpenCV [2] to manage image manipulation (loading, saving, displaying), but can use any library to implement the wavelet transform. For wavelet implementations, look at https://wavelets.org/software.php I would suggest using the GNU Scientific Library for the wavelet transform, if you use C++.

Your system needs to implement at least the following:

1) specify a query image and a number of matches to return;
2) display the query image and the returned images, ranked from best to worst;
3) write out/save the returned images;
4) return the times required to compute the wavelet features and the system processing times

If the system works and implements the above requirements, you can score up to 85% of the implementation mark. The remaining 15% is for additional optimizations and explorations. These include, but are not limited to,

- acceleration structures for the key matching search e.g. kd-tree
- partial image query, as specified in the paper

- PCA dimensionality reduction to shrink the wavelet feature vectors.

Note that while the authors claim their system is fast, this is for a single query image, and they test this against every image in the database. This is not a good idea in general.

Report

In addition the implementation, you must write and submit a report to describe your implementation, how to use it, and the results you achieved on a reasonable test corpus.

The report should be no more than **8** A4 pages (single column, 11pt font), excluding images. The report must contain the following:

1) An introduction which outlines the problem and context, the solution used and also presents a (very) brief comment on your main findings.
2) A system section which explains details on how you designed and implemented your system – including software used, any assumptions made; any unimplemented features should be note here. This is more about design decisions and algorithms used – not UML etc. You can cite the paper to avoid having to repeat information on wavelets. This section is about your work, not simply a rehash of the paper.
3) An analysis and discussions section: you must present details of your test corpus (you can find standard test sets online if you use google). Explain how you ran your tests as well as the hardware you used. Present your results appropriately i.e. graphs, tables etc which summarise matching performance as well as subjective assessment (since the algorithm will often give bad results). You can use a subset of the database and run experiments on this – using both an image from the database as query, as well as those from elsewhere. Since you can determine whether a match is valid (visually), you can compute scores such as true/false positive rates, precision, recall and accuracy (you can look these up on Wikipedia if you are not familiar with them).
4) A conclusion which restates the original problem, how you solved it and summarises your key findings and limitations. You should also include a paragraph on possible future extensions.
5) A short system user manual should be appended to the report as an extra page.

The report will contribute 30% of the project mark; the remaining 70% will be from the implementation. The implementation will need to function as required and produce good results to score highly.

Handin

You must submit your code as well as your written report (a PDF) in **one** zip archive.

Demo

You will need to demo your software to me, so that I can see it in action. I will also ask you questions about your implementation and findings. You must either bring your image database with you on a flashdrive, or have it resident on your system so we can run tests during the demo. You may use your own laptop for the demo.

References/Links

[1] “Content-based image indexing and searching using Daubechies wavelets”, Wang et al. 1997.

[2] OpenCV, http://www.opencv.org/