A Flexible Image Retrieval Framework

Raoul Pascal Pein[1,2], Zhongyu Lu[2]

[1] Multimedia Systems Laboratory (MMLab),
Faculty of Engineering and Computer Science,
Hamburg University of Applied Sciences,
Berliner Tor 7, 20099 Hamburg, Germany
[2] Department of Informatics,
School of Computing and Engineering, University of Huddersfield,
Queensgate, Huddersfield HD1 3DH, United Kingdom

Abstract. This paper discusses a framework for image retrieval. Most current systems are based on a single technique for feature extraction and similarity search. Each technique has its advantages and drawbacks concerning the result quality. Usually they cover one or two certain features of the image, e.g. histograms or shape information.
The proposed framework is designed to be highly flexible, even if performance may suffer. The aim is to give people a platform to implement almost any kind of retrieval issues very quickly, whether it is content based or somehing else. The second advantage of the framework is the possibility to change retrieval characteristics within the program completely. This allows users to configure the ranking process as needed.

Keywords: Content-based image retrieval (CBIR); retrieval framework; feature vectors; query image; combined retrieval; improved result quality

1 Introduction

Most available programs which provide an image search have very limited possibilities. They lack a flexible design allowing to fine tune the software to the user's requirements. Often they are capable of searching based on only a few features. Especially the separation between common retrieval approaches and specialized Content based Image Retrieval (CBIR) is visible.

Based on this observation, a standard design for image retrieval systems has been developed, which can be implemented and extended as simple as possible. This paper presents a framework which is:

- platform independent
- extensible and capable of offering individual user interfaces
- based selectively on database or file system
- capable of ranking several features simultaneously
- useful locally and in the web
- allowing manual annotation to improve quality

The proposed framework is based on the system described in [12,13]. The emphasis is clearly placed on the properties mentioned above, rather than a high performance. Some design decisions are deliberately made to provide flexibility. The important background of CBIR systems can be found in several papers like the Northumbria report [4] and other related papers [15,14]. A standard design for CBIR is presented by by Veltkamp and Tanase [17] which represents the direct background for this paper. A combined indexing approach exploiting multiple features is put on top of that design.

In Section 2 the basic types of image retrieval concepts are presented. Each type has specific advantages and disadvantages which the proposed design tries to exploit and overcome. Section 3 gives a brief overview of how the system is generally designed and how the single components interact. Section 4 describes a currently working implementation. The prototype is still under development aiming at a better result quality and a servlet based user interface.

2 Related Work

Most image retrieval systems are based on one or more of these four basic indexing techniques: *Keywords*, *Tags*, *Semantics* or *Content*. All these techniques have specific advantages and disadvantages.

The "classic" approach is the use of keywords. This technique clearly is based on a considerable amount of knowledge. Its advantage is that much previous research from different areas can be reused, as most retrieval work has been done on text based documents. The major drawback of this approach is the difficulty to extract useful keywords from images automatically. Current search engines used locally (i.e. Beagle[16], Google Desktop Search[7]) or in the World Wide Web usually understand textual queries.

A second approach is to use tags which are used to generate clusters of similar images. This approach makes searching very simple and straightforward. The query consists of one or more tags and the engine only needs to filter out the matches. Adding tags to an image can be done manually with a maintainable effort. As simple tags are valid for many similar images at once, they can be selected and tagged very quickly. The drawback is the bad accuracy of discrimination. If thousands of images are tagged with the same tags, the retrieval process is rendered quite useless. F-Spot [5] is a personal retrieval software using tags based on "Emblem Tags" in Gnome [2]. A much more ambitious project is the web based Flickr [6]. This program also allows to attach "geotags" enabling the user to find pictures taken dependent on their location.

To introduce high quality results, the semantics of files can be used for retrieval. This technique allows to find data based on very specific content, eliminating ambiguities in the natural language. Unfortunately, the research in semantics and semantic web is currently in a very early state and not even text based engines are working satisfactorily. Also the annotation has to be done very carefully and by experts to achieve significant improvements. An example for semantic image retrieval is the Ontogator project [8] or the approach by Zhang and Izquierdo based on Bayesian Networks [18]. A more lightweight approach to semantics can be found in TopicSEEK [3], where less effort is put into mathematical correctness but in usability.

The fourth technique introduces image specific attributes. It concentrates on the actual content, represented by the pixels. This allows to use maths for extracting the relevant indexing data automatically. Ideally, there is no manual input required to build a large index. Many different algorithms are possible to extract features of many kinds. The difficulty is to analyse the real content of an image, as seen by a user. This would require a currently not available (and probably impossible to implement) recognition algorithm. In addition a special kind of user interface is required, as queries cannot be directly mapped to a string. Examples are given by QBIC [11] and Virage [1].

An approach combining content and semantics has recently been described by Lam and Singh [10].

3 Proposed Design

The proposed design (fig. 1) is based on similar projects as well as cognitions from the preceding prototype.

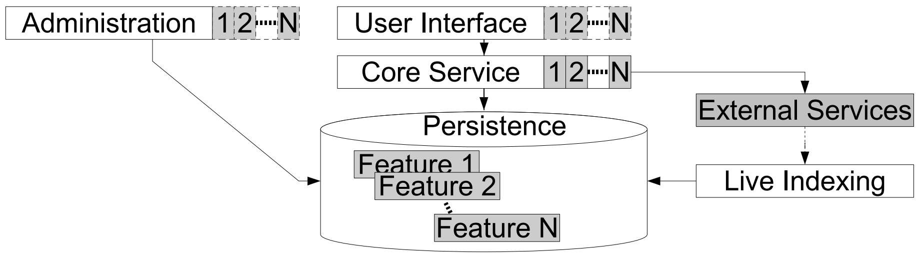

Fig. 1. The retrieval architecture consists of 6 basic modules. *Core Service* and *Persistence* can be enhanced by adding new features. Not content based features may also require an optional visualization in *User Interface* and *Administration*.

It is tried to model all components in the framework replaceable. Basically there are six different modules in the design: *Core Service*, *Persistence*, *User Interface*, *Administration*, *Live Indexing*, *External Services*.

Core Service This module contains the whole retrieval logic, optimized on CBIR. For each single feature one separate index is created. Every feature needs to provide only two different algorithms: extraction of the binary (or else) feature from a pixel image and the calculation of similarity mapped to a range between 0 and 1. Based on these two algorithms, almost any feature can be supported directly. Queries containing more than a single feature are merged internally.

The most important function is to hand over a query to the server and to get a result set, which contains information about all relevant images. The query should contain limitations where to cut off the ranking (minimal similarity, maximum result size) and some additional parameters to control each sub ranking for a single feature (especially CBIR related data). For each image x a combined and weighted ranking r_x is calculated based on the query. It is basically a weighted sum

$$r_x = \frac{1}{\sum_{f=1}^{n} w^f} * \sum_{f=1}^{n} w^f * r_x^f \tag{1}$$

where w^f is the weight/importance of feature f. r_x^f is the partial ranking for image x using feature f. The optimal values for the feature weights depend on the users requirements. By altering these values the search characteristics of the engine can be changed completely. Alternative calculations (e.g. intersection of the results) may also be interesting for fine tuning.

Persistence In the persistence layer all indexing data is stored. This can be a database, a file system or something completely different. There are only some basic requirements which are defined in the interface.

The basic access methods require opening and closing the source and read, update, delete and add of a single dataset. For searching, a sequential iteration over all datasets is required. To optimize the retrieval time, it may also be useful to have filtered access to a subset of an index.

User Interface The user interface may vary widely. As well fat clients using a middleware as thin clients providing a web interface are possible. The interface of the core server is supposed to offer every service required, like searching and some information about the provided features. For manual annotation, writing feature data for an image directly to the repository needs to be offered.

Administration A separate control instance is highly recommended to bar users from tinkering with important settings. This administration module needs direct access to the persistence module. It can be used to add new images into the repository and for setting basic properties for fine tuning the system.

Live Indexing In order to provide live results which correspondent to locally available resources, a daemon process can be set up. This process monitors changes in the file system and adds new images automatically.

A web based service might use a similar program for automated updates. As the topology of the world wide web is different to a file system, a spider like program may be implemented.

External Services Everything available from other retrieval services should be accessible. The core service is primarily designed to process image content, rather than textual queries. It is thought to exploit the techniques currently used in text based retrieval without re-implementing everything. A textual query can be used to get a list of relevant images which can be ranked with CBIR in the core system.

4 Implementation

A great deal of platform independence is already given by using a modern programming language like Java. Being aware of system specific properties like file names, the portability is reasonably high without much effort. Using further technologies requires also attention. To realize the prototype, components are chosen which are available on different platforms. For the underlying database, MySQL has been chosen, as it runs on most PCs. Further it is aimed not to use MySQL specific features to keep the migration effort to another data base low.

The current system is meant to be connected to an external engine to exploit existing text based search. The aim is to forward a search string from the user interface to this engine and get a list of matching files. This only requires a very simple interface, which will be realized as a small plug-in.

A promising engine seems to be the open-source project Beagle [16] which limits use of the extended image retrieval to Linux. The modular plugging of the external source allows to exchange the engine dependent on the user's needs.

To prove the design, support for a couple of different features is being implemented. Currently three different features are supported: keywords, histograms and spatially distributed histograms. Some other modules like a wavelet based one [9] and tags [2] are under development. Each module needs to implement two basic methods: *calculateFeatureVector(String fileName)* is required to extract the features from a file and *double calculateSimilarity(FeatureVector fv)* allows comparing two datasets. Optionally a specific GUI frame can be designed to provide a detailed query composition (e.g. entering keywords).

The realization of the histogram/wavelet based features provide gradual rankings anywhere between 0.0 and 1.0. Here a full scan over all datasets is adequate to achieve exact results, even if the required processing time is quite high. To improve speed, the index could also be put into a multi dimensional tree. In this case some otherwise good hits may be missing in the result set.

The text and tag based search is characterized for a much more distinctive ranking. Results may either be exactly 1.0 (hit) or exactly 0.0 (miss), slight differences may be realized with intermediate values. In the framework, these features are also easy to implement in the first place, especially for testing prototypes. Nevertheless a sequential search does not scale and the retrieval can be improved dramatically by adding an appropriate index structure (e.g. hashes, string based sorted trees) without losing any relevant result.

5 Results

The test data base contains information of 1709 images. To evaluate the efficiency of the combined ranking, the results of several queries are analyzed. The program combines three different result sets by using equation 1. As the *Keyword* component cannot handle query images, an additional query keyword is set. The query image shows a scene in Liverpool, therefore the query ”liverpool” is passed to the engine.

This keyword represents the not content based queries in section 2. The search engine now has two independent views to calculate the final ranking. Figure 2 shows the resulting image set. The weighted ranking is displayed in figure 3 listing the detailed ranking for the first 125 images. To refine the result set, the weigths w^f for each of the three modules have been adjusted manually.

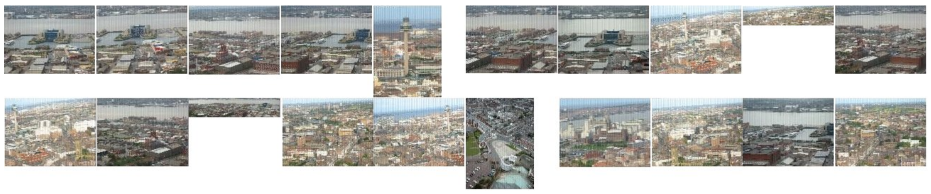

Fig. 2. The screenshot shows the top 20 results of a combined ranking. The query image (*upper left*) is ranked highest. Similarity decreases from left to right and top to bottom.

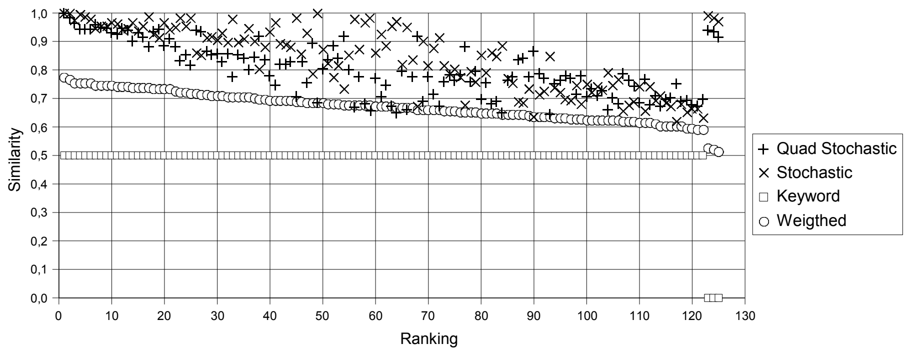

Fig. 3. The diagram visualizes details of the combined and weigthed ranking of fig. 2. The final ranking is based on the weighted similarity (*Weighted*).

6 Discussion and Analysis

The ranking generated by the retrieval service shows an eye-catching similarity between the first images in figure 2. A couple of brighter images with similar content got slightly higher ranks than some of the visually closer related images. This can easily be explained by the lossy compression of the histograms. An important fact is, that all results are somehow related to "liverpool". Queries without keyword restriction produce result sets containing several unrelated images in between the desired ones.

A way to overcome the drawbacks of each single indexing technique is to combine their strengths. Figure 3 shows the effect of combined retrieval. In this case, especially the *Keyword* module is a highly efficient filter to reject false positives. At the lower end (rank 123-125) some images had a very high similarity (> 0.9) when only considering the content based modules. All 122 top ranked images all contain the specified keyword, while the similarity drops from 1.0 to about 0.6. All of the first 24 images show houses of Liverpool from above. Turning off the keyword search leads to a reatively high amount of images completely out of context, as the content based modules have no way to filter them out. If the user is only interested in visually similar images, disabling keywords helps to find images in less keyword-related regions of the repository.

7 Conclusion and Future Work

This paper shows a possible way to develop a flexible image retrieval system. The prototype design is a trade off between extensibility and high performance with an emphasis on extensibility. It is assumed that a combined retrieval can be used much more widely than a highly specialized one.

Achievements The example query demonstrates the basic capabilities of the framework. It is possible to implement different feature modules supporting any of the types described in section 2. Tags and semantics are basically a special variation of keywords. The hard work with semantics need to be done in creating ontologies and adding certain values to an image, but this has nothing special to do with the framework itself.

Adding a new feature module can be done with minimal effort. There are only two important functions that need to be implemented plus some template-like ones for persistence issues (i.e. mapping from objects to low-level data types and back).

Future Work The future work focuses on implementing new feature modules and web support including active user annotation. Further it is aimed to improve the text search by using specialized external search engines. A layer capable of building fast and scalable index structures is also under development.

As soon as multiple highly distinctive features are available, a detailed survey is planned. The objective is to evaluate whether the proposed combination of features leads to noticeable improvements in the result quality. The results of single features will be compared to combined features on a basis of several thousand images and many test persons to gain a significant representative sample.

References

1. J. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Gorowitz, R. Humphrey, R. Jain, and C. Shu. Virage image search engine: an open framework for image management. In I. K. Sethi and R. C. Jain, editors, *SPIE, Storage and Retrieval for Image and Video Databases IV*, pages 76–87, February 1996.
2. James Barrett. Emblems and tagging in gnome, 2006. Available from: `http://live.gnome.org/EmblemTags`.
3. Andreas Christensen. Semantische Anreicherung von Suchanfragen auf Basis von Topic Maps., June 2005. Diplomarbeit.
4. J.P. Eakins and M.E. Graham. Content-based Image Retrieval. A Report to the JISC Technology Applications Programme. Technical report, University of Northumbria at Newcastle, January 1999. Available from: `http://www.unn.ac.uk/iidr/VISOR`.
5. Larry Ewing. F-spot - personal photo management, 2006. Available from: `http://f-spot.org`.
6. Flickr - Photo Sharing, 2006. Available from: `http://www.flickr.com/`.
7. Google desktop search, 2006. Available from: `http://desktop.google.com`.
8. Eero Hyvönen, Samppa Saarela, and Kim Viljanen. Intelligent image retrieval and browsing using semantic web techniques - a case study. Technical report, Helsinki Institute for Information Technology (HIIT) / University of Helsinki, 2003.
9. Charles E. Jacobs, Adam Finkelstein, and David H. Salesin. Fast multiresolution image querying. *Computer Graphics*, 29(Annual Conference Series):277–286, 1995. Available from: `citeseer.ist.psu.edu/jacobs95fast.html`.
10. Tony Lam and Rahul Singh. *Advances in Visual Computing*, volume 4292/2006 of *Lecture Notes in Computer Science*, chapter Semantically Relevant Image Retrieval by Combining Image and Linguistic Analysis, pages 770–779. Springer Berlin / Heidelberg, 2006.
11. W. Niblack, X. Zhu, J. Hafner, T. Breuel, D. Ponceleón, D. Petkovic, M. Flickner, E. Upfal, S. Nin, S. Sull, B. Dom, B.-L. Yeo, S. Srinivasan, D. Zivkovic, and M. Penner. Updates to the qbic system. *Retrieval for Image and Video Databases VI*, 3312:150–161, 1998.
12. Raoul Pascal Pein. Multi-Modal Image Retrieval, April 2005. Diplomarbeit.
13. Raoul Pascal Pein and Zhongyu (Joan) Lu. Content Based Image Retrieval by Combining Features and Query-By-Sketch. In Hamid R. Arabnia and Ray Hashemi, editors, *The 2006 International Conference on Information & Knowledge Engineering*, pages 49–55, 2006.
14. Monika Renz and Wolfgang Renz. Neue Verfahren im Bildretrieval. Perspektiven für die Anwendung. In R. Schmidt, editor, *Proceedings der 22. Online-Tagung der DGI*, pages 102–128, May 2000.
15. S. Shatford. Analyzing the Subject of a Picture: A Theoretical Approach. *Cataloging and Classification Quarterly*, 6:39–62, 1986.
16. Joe Shaw. Beagle desktop search, 2006. Available from: `http://beagle-project.org/Main_Page`.
17. Remco C. Veltkamp and Mirela Tanase. Content-Based Image Retrieval Systems: A Survey. Technical Report UU-CS-2000-34, Department of Computing Science, Utrecht University, October 2002.
18. Qianni Zhang and Ebroul Izquierdo. *Semantic Multimedia*, volume 4306/2006 of *Lecture Notes in Computer Science*, chapter A Bayesian Network Approach to Multi-feature Based Image Retrieval, pages 138–147. Springer Berlin / Heidelberg, 2006.