

A Generic Concept for the Implementation of Medical Image Retrieval Systems

Mark O Güld, Christian Thies, Benedikt Fischer, Thomas M Lehmann

Department of Medical Informatics, Aachen University of Technology (RWTH), Aachen, Germany

Abstract

This work presents mechanisms to support the development and installation of content-based image retrieval in medical applications (IRMA). A strict separation of feature extraction, feature storage, feature comparison, and the user interfaces is suggested. The concept and implementation of a system following these guidelines is described. The system allows to reuse implemented components in different retrieval algorithms, which improves software quality, shortens the development cycle for applications, and allows to establish standardized end-user interfaces.

Keywords: Medical Imaging; Medical Image Archive; Content-based Image Retrieval (CBIR); Picture Archiving and Communication Systems (PACS); Digital Imaging and Communication in Medicine (DICOM)

1. Introduction

The growing number of digital image acquisition and storage systems in clinical routine rises demands for new access methods. Still, most picture archiving and communication systems (PACS) only use manually entered textual information to access a patient's image data. Content-based image retrieval (CBIR) depends on automatically extracted content descriptions (features) for each image as well as their storage and comparison upon a query [1].

Considering the implementation of a CBIR system in medical applications, there is currently a gap between monolithic CBIR systems for general-purpose image retrieval, e.g. Blobworld [2], and programming tools for the development of image processing algorithms. Existing general-purpose CBIR systems closely couple feature extraction, feature storage, feature comparison, and the query interface. Since changes often affect all system components, this makes it difficult to extend them. However, existing image processing tools provide a huge number of routines useful for feature extraction and comparison, but they lack support for organising feature storage and easy deployment of algorithms to the end-user, who is not involved in technical details.

Extensibility and flexibility

TAGARE et al. have pointed out the specific demands and challenges of CBIR in the medical domain [3], which significantly differ from demands of general-purpose CBIR [4]. To provide satisfying query results, multiple levels of content abstraction are necessary [5], with each abstraction level using a completely different type of features as well as a rough detection of relevant image regions, e.g. global features for categorization, per-pixel features for local analysis and structural features to express spatial relationships between identified objects. The context in medical queries can also vary, e.g. attention to details of bones or tissue structure. Thus, most medical CBIR implementations focus on a certain domain [1]. Furthermore, medical knowledge continuously evolves and properties or quantifications used in medical language are sometimes difficult to express precisely in

computerized methods. To keep the retrieval system extensible, all algorithms must be divisible into reusable computation steps, e.g. pre-processing, feature extraction, and feature comparison, which can be easily parameterised and combined.

Separation of algorithms and interfaces

Standardized guidelines for graphical user interfaces are required to ensure the acceptance by end-users and to minimize required learning time. Although medical retrieval algorithms may carry unique semantics, most image retrieval applications use the query-by-example (QBE) paradigm: the submission of a sample image or image region is answered by the system displaying a list of archived images which are most similar to the sample. Thereafter, query refinement may follow. Decoupling the interface from the application-specific retrieval algorithm allows to compose all interfaces from standardized modules [6].

Development and deployment

Beside the decomposition of the algorithm, each of these steps should optionally be divisible at code level with the development environment hiding as many details of the build process as possible. The major goal is to minimize the amount of information that must be shared between all developers, as this allows them to focus on their primary field of expertise. Especially the implementation of GUIs completely differs from that of image processing algorithms. Thus, the deployment, i.e. the integration of new retrieval algorithms into existing user interfaces for testing and final release, must be as easy as possible.

Efficiency at run-time

The process of automatic image content extraction at various abstraction levels requires considerable amounts of computation time and storage space. The decomposition of algorithms as proposed above can also be exploited for distributed computation at query time. Distributed systems based on standard PC hardware provide a cost effective way to cope with these demands.

2. Materials and Methods

In our approach, a central relational database is used to store all entity definitions and administrative information, while source code and larger feature data are stored separately [7].

Entities of the algorithm model

The system can be extended by adding instances of the following entity types: *feature types*, *methods*, *networks*, i.e. algorithms, and *experiments*, i.e. partly parameterized networks, which are used to generate *queries*. Queries completely parameterize experiments, i.e. they define all input data for the invocation of an algorithm. All entities are defined via the eXtensible Markup Language (XML) and can be imported into the system using the according administration tool. The system provides features as a data container which is transparently stored.

Features carry type information, which defines the semantics of the data inside the container. Consequently, the developer must provide a feature type definition for each newly introduced content description. The system allows symbolic inheritance to express feature type compatibility. For example, a texture feature consisting of a tuple of floating

point values should be compatible with a general floating point vector as used by a lot of distance measures:

```
<featuretype name="texture_tamura">
  A 3D histogram of texture features proposed by TAMURA:
  Directionality, contrast and coarseness.
  <isa> vector </isa>
</featuretype>
```

Methods encapsulate a transformation of feature tuples and define the atomic processing steps inside algorithms. To cover all possible transformations, there are three method types: a method can either transform one input tuple into one output tuple (1:1), compress a set of feature tuples into one tuple (T:1, e.g. prototype computations or data set analysis like principal component analysis), or expand one feature tuple into a set of feature tuples (1:T). A method accesses the tuples via its inputs and outputs. Beside the method implementation, the developer must provide the method's feature interface, which defines the expected feature type for each input and the type of the generated feature for each output:

```
<method name="extract_tamura" type="1:1" project="texture_tamura">
  <input featuretype="image" keeps_ref="1"> Input image. </input>
  <output featuretype="texture_tamura"> Texture feature. </output>
  Extracts texture features proposed by TAMURA from an image.
</method>
```

Networks are used to build algorithms by defining the flow of features between methods and their dependencies inside the computation. Consequently, networks also have a feature interface, which is expressed using sources and sinks. Sources are nodes with exactly one output and no input and refer to one feature as part of the algorithm's input. Sinks provide the analogue for the algorithm's output. The network also ensures a consistent parameterisation, e.g. identical parameters for a feature extraction method which is applied to both reference images and a sample image.

Experiments are used to partly parameterise networks. This is done by assigning suitable features to some of the network's source nodes, e.g. empirically determined parameters for feature extraction. Experiments hide fixed parameters from query-relevant parameters, e.g. the selection of a query image. By assigning features to the remaining sources, the end-user provides the complete set of input features.

This triggers a query, which is executed within the backend (see below). Once the backend completes the computations, the user interface can access the features associated with the sink nodes.

Web-based interfaces

All end-user interfaces to the retrieval system are JAVA applications or HTML dialogues implemented in PHP. Therefore, the end-user requires a web browser and a JAVA virtual machine. For interactive applications, a PHP script can initiate a query and blocks until the backend completes the computations. The backend returns the resulting features, i.e. the features propagated at the sinks of the network. For example, this might be a list of images similar to the selected sample. More complex feature types such as hierarchical attributed region-adjacency graphs (HARAGs) [8] demand more sophisticated viewing applications. Therefore, the PHP script can also generate hyperlinks to a JAVA application responsible for this feature type, which can be integrated into web pages via JAVA WebStart. The JAVA application accesses the demanded feature from the web server via the hypertext transfer protocol (HTTP).

It should be noted that the experiment entity implicitly defines the interface of available algorithms to the end-user. Since each source and sink carries feature type information, it is possible to abstract an algorithm from its interface to the end-user. Thus, it is possible to obtain a functional end-user interface by implementing only a generic one which includes input and output modules for each occurring feature type in the experiment's interface. This is illustrated in Fig. 1.

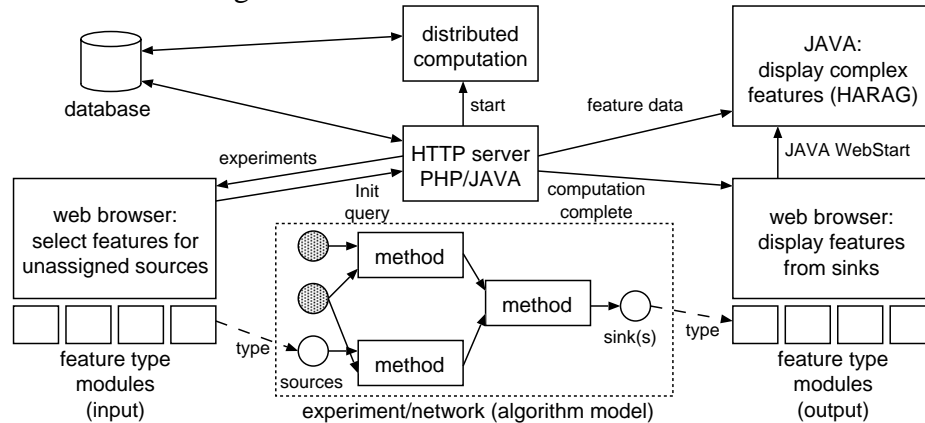


Figure 1 – Schematic view of the system components used for the retrieval process.

Development environment

The development environment organises all implementation parts as *projects*. Projects encapsulate internal functionality, exported functionality, which is visible by other projects, stand-alone programs, and methods. The supported programming languages are C and C++, but all methods must have a C++ interface. The developer provides direct dependencies from other projects and external libraries via the project's configuration file. Known external libraries can be accessed by a project under their symbolic names. The dependencies are automatically unrolled by the build system. Their configuration and integration into the build process is transparent to the developer. The build system is implemented on top of the GNU make utility.

Distributed computing and storage

All entity definitions are stored in a central relational database. For efficiency, small features are directly stored inside the database, while images and complex feature data are stored on file servers and the database only tracks each feature's location.

When a query is issued, a central scheduling service determines the sequence of all necessary method calls using the underlying network definition and the source nodes' features. For each method call, the scheduler picks a suitable host in the network cluster, determines the nearest locations of the involved input features, and allocates the resulting output features. Afterwards, the call is dispatched to a daemon service running on the selected host. The communication between scheduling service and the daemons uses HTTP. Before calling the method, the daemon fetches all input features and provides them as objects, making the features' locations transparent to the method. The database also stores each feature's generation history, so that already computed features can be identified during query processing and the respective computations are skipped.

3. Results

The system was implemented using open standards and free software and currently runs on Intel/Linux and Sparc/Solaris platforms. At this time, the systems holds 13,151 images from the Department of Diagnostic Radiology, Aachen University of Technology and 2,355

dental radiographs obtained from the Department of Oral Maxillofacial Radiology, Göteborg University, Sweden. The contents of these images were encoded by radiologists using a hierarchical coding scheme, which provides the ground truth for query experiments. The system also stores other medical and non-medical image data for evaluation purposes.

All important administrative operations like image import, export, and content encoding can be performed using web-based interfaces. The currently available retrieval algorithms use the QBE paradigm and can therefore be used from a single web-based query interface, which also offers a basic relevance feedback/query refinement functionality. An extensive object-oriented and template-based PHP library was developed to support the development of the graphical user interfaces.

The implementation and testing time of methods is significantly decreased, since a number of core projects contain often-required functionality, e.g. image file access and classification-related data structures, and the methods do not need to concern the access to features. Instead, all feature data and parameters are accessible locally via objects. Simple pre-processing steps like image filters can be implemented and deployed within minutes. The feature type concept and the method definition allows it to use a method in arbitrary context compatible with the method's interface. Existing algorithms can be modified without changing the method implementations. For example, additional pre-processing steps can be integrated by modifying the network entity. The parameterisation can be altered by editing the experiment entity. The deployment of new algorithms is done by defining an experiment entity. This mainly consists of setting up algorithm's parameters by assigning parameter features to the respective sources and choosing a set of reference images. If a compatible PHP interface for the algorithm's interaction scheme exists, it is directly available online. So far, several retrieval algorithms were implemented to carry out the experiments for several publications. The retrieval system is accessible from anywhere on the internet via a web browser and a JAVA virtual machine and does not require any maintenance or configuration by the end-user.

The run-time environment can effectively utilize the computation power of a computer cluster. This is also useful during the feature extraction process, which is performed offline. Most content descriptions are computed per image with no dependencies between single images during the extraction process, which is ideal for distributed processing. For this scenario, first experiments with ten identical stations yielded a speedup factor of 8, which is primarily limited by the bandwidth of the file server holding the feature data. Like feature access, the concurrent processing is transparent to the method implementation.

4. Discussion

The more general modelling approach for the retrieval algorithms causes a certain overhead. Thus, the efficiency of optimized monolithic systems cannot be reached. On the other hand, the proposed mechanisms provide several advantages regarding extensibility, maintenance and reusability. The primary benefits result from the strict separation of all system entities, which can be transparently accessed using well-defined interfaces. For each new entity, its creator must provide a proper documentation. Afterwards, the interface documentation is all the co-developers need to use the entity. This very basic concept is employed on all levels. During the implementation, other projects are easily accessible and the build process is configured automatically. At run-time, methods are automatically called and provided with the required feature data, so the developer only has to implement the feature transformation itself. The composition of new algorithms using existing methods can solely rely on the methods' feature interface. A new algorithm for an existing user interaction scheme (e.g. QBE) can be deployed by defining an experiment which fits the scheme. These two steps are manageable without knowing any details about method

internals or source code and significantly speed up development and testing cycles. The entity interfaces also help to establish a standardized documentation, which is valuable especially for groups of developers. Based on the concepts proposed in [9], a full integration of the system into the clinical workflow is currently being planned.

5. Conclusion

The proposed concept allows it to effectively support the development and deployment of content-based image retrieval in medical applications. The strict separation of entities at different levels (modelling, implementation, deployment and run-time) allows a maximum transparency for each person involved, both on the developer's and on the physician's side. When fully integrated into clinical routine, content-based access to image data promises a significant impact in the fields of evidence-based medicine, case-based reasoning, and computer-based training.

6. Acknowledgement

This work was performed within the IRMA project, which is funded by the German Research Community (DFG), grant Le 1108/4.

7. References

- [1] Müller H, Michoux N, Bandon D, Geissbuhler A: A review of content-based image retrieval systems in medical applications. Clinical benefits and future directions. *International Journal of Medical Informatics* 2004; 73: 1-23
- [2] Carson C, Belongie S, Greenspan H, Malik J: Blobworld – Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(8): 1026–1038, 2002.
- [3] Tagare HD, Jaffe CC, Duncan J: Medical image databases – a content-based retrieval approach. *Journal of the American Medical Informatics Association* 4:184–198, 1997.
- [4] Smeulders AWM, Worring M, Santini S, Gupta A, Jain R: Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2000; 22(12): 1349-80
- [5] Lehmann TM, Güld MO, Thies C, Fischer B, Spitzer K, Keysers D, Ney H, Kohlen M, Schubert H, Wein BB: Content-based Image Retrieval in Medical Applications. *Methods of Information in Medicine* 2004; 43(4): 354-361
- [6] Lehmann TM, Plodowski B, Spitzer K, Wein BB, Ney H, Seidl T: Extended query refinement for content-based access to large medical image databases. *Proceedings SPIE* 2004; 5371: 90-98
- [7] Güld MO, Thies C, Fischer B, Keysers D, Wein BB, Lehmann TM: A platform for distributed image processing and image retrieval. *Procs SPIE* 2003; 5150:1109-1120
- [8] Thies C, Metzler V, Lehmann TM, Aach T: Extraction of biomedical objects by sub-graph matching in attributed hierarchical region adjacency graphs. *Proceedings SPIE* 2004; 5370(3): 1498-1508
- [9] Lehmann TM, Wein BB, Greenspan H: Integration of Content-based Image Retrieval to Picture Archiving and Communication Systems. *Proceedings Medical Informatics Europe (MIE 2003)*, IOS Press, Amsterdam, ISBN 1 58603 347 6 (CD-ROM only).

8. Address for Correspondence

Mark Oliver Güld
Department of Medical Informatics
Aachen University of Technology (RWTH)
Pauwelsstr. 30
D - 52062 Aachen, Germany

Email: mgued@mi.rwth-aachen.de
Web: irma-project.org