

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 paper presents a technical framework to support the development and installation of system for content-based image retrieval in medical applications (IRMA). A strict separation of feature extraction, feature storage, feature comparison, and the user interfaces is suggested. This allows to reuse implemented components in different retrieval algorithms, which improves software quality, shortens the development cycle for applications, and allows to introduce standardized end-user interfaces. Based on the proposed framework, the IRMA engine has been established, which is currently used to evaluate content-based retrieval methods on a collection of 20,000 medical and 135,000 non-medical images.

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 textual information to access a patient's image data, which has been mainly entered manually. Content-based image retrieval (CBIR) depends on automatically extracted content descriptions (numerical features) for each image as well as their storage and comparison upon a query.

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 [1], and programming tools which support the development of image processing algorithms and the automatic distributed execution of them. 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 according to the specific requirements of medical image retrieval. Existing image processing tools like Khoros/Cantata/VisiQuest¹, the Insight Toolkit (ITK)², the Visualization Toolkit (VTK)³, or ImageJ⁴ provide a huge number of routines useful for feature extraction and comparison out of the box, but they lack support for organising feature storage as needed by a CBIR system. This also complicates the easy deployment of retrieval algorithms to the end-user, who must not be involved in technical details.

To close this gap and further motivated by the considerable data volume of medical image archives, recent works apply grid technology to medical CBIR [2]. The grid concept coordinates resource sharing in dynamic virtual organisations [3], which incorporates data sharing as well as access to computers and software. A suitable concept is required to run complex workflows like CBIR algorithms on a grid infrastructure [4][5]. For instance, *MediGRID* [6] provides the distributed storage of a large scale image database and utilises distributed computing for content-based retrieval on the image data. The main focus lies on the optimization of the queries themselves.

¹ <http://www.accusoft.com>

² <http://www.itk.org/>

³ <http://public.kitware.com/VTK/>

⁴ <http://rsb.info.nih.gov/ij/>

However, the user interface, the feature extraction, and a general model which supports the development of retrieval algorithms are not discussed.

HASTINGS et al. implemented *GridPACS* [7], which additionally supports the development and execution of image analysis algorithms on its grid infrastructure [8]. It provides a wrapper for ITK/VTK programs to build a feature extraction chain and stores the resulting image features as user-defined meta-data. However, the coupling between retrieval algorithms and their user-interfaces is not discussed, and the model for image content abstraction is strict, as images can only be treated individually during feature extraction. Also, the algorithmic model does not seem to include the feature comparison step at query time.

TAGARE et al. have pointed out the specific demands and challenges of CBIR in the medical domain [9], which significantly differ from demands of general-purpose CBIR [10]. More general, a framework for medical image retrieval must cope with four important aspects.

Extensibility and flexibility

To provide satisfying query results, multiple levels of content abstraction are necessary [11], with each abstraction level using completely different types 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 hierarchical features to express spatial relationships between identified objects. The context in medical queries can also vary, e.g. the physician might focus on details of bones, tissue structures, or the quantitative analysis of organs imaged. 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, the retrieval

algorithms must be divisible into reusable computation steps, e.g. pre-processing, feature extraction, and feature comparison, which can be easily parameterised and combined into a processing chain.

Separation of retrieval algorithms from their interfaces

Standardized guidelines for graphical user interfaces are required to ensure the acceptance by end-users and to minimize the required learning time. Although some retrieval algorithms may carry unique semantics and demand a specialized user interface, 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 with a list of reference images, sorted by their similarity to the sample. Hence, it is also possible to re-use graphical user interfaces (GUI) if the interface to the underlying retrieval algorithm is properly defined.

Development and deployment

Beside the decomposition of the processing chain, 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 automatic extraction of the image content at various abstraction levels requires considerable amounts of computation time and storage space. The decomposition of the

processing chain should also support an automatic distributed computation at query time. Distributed systems based on standard PC hardware provide a cost effective way to cope with these demands.

Due to these challenges, most medical CBIR implementations focus on a certain domain [12]. This paper describes the architecture of the IRMA system for content-based image retrieval in medical applications⁵ and illustrates how the demands listed above can be addressed. Some parts of the proposed system have been described in earlier publications. Details of the algorithm model and the distributed computation of retrieval tasks are described in [13], and the concept of user interfaces can be found in [14].

2. Materials and Methods

For the implementation of a medical CBIR system, an abstraction of the content analysis process is proposed to meet the above requirements. It incorporates concepts from development tools for image processing algorithms and distributed computing.

Entities of the algorithm model

The system can be extended by adding instances of the following entity types: *feature*, *method*, *network*, *experiment*, and *query*. All entities are defined via the eXtensible Markup Language (XML) and can be imported into the system using the according administration tool.

Features are used to model the image data itself as well as all results of the content analysis. They are provided as a data container which is stored and retrieved automatically by the system. They carry a type information, which defines the semantics

⁵ <http://rma-project.org>

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 (Fig 1).

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

Figure 1 – A feature type definition: The “is-a” relationship specifies type compatibility.

Methods encapsulate a transformation of input features and define atomic computation steps inside the processing chain. A method interfaces features via its inputs and outputs: a method can have an arbitrary (but fixed) number of inputs and an arbitrary (but fixed) number of outputs. Each input and output carries semantics and is consequently bound to a feature type. The assignment of features to all inputs results in an input tuple, and the method generates features at its outputs, the output tuple. 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, e.g. the cut operation on radiographs which contain multiple shots on one film). When a new method is inserted, the developer must provide the method's feature interface, i.e. the expected feature types for each input and each output. Additionally, the definition file contains information about the location of the method's source code within the

development environment (see below) and serves as a standardized documentation for the particular method (Fig 2).

```
<method name="extract_tamura" type="1:1" project="texture_tamura">
  <input featuretype="image" type="std_feature_keeps_ref"> Input image. </input>
  <output featuretype="texture_tamura"> Texture feature. </output>
</method>
```

Figure 2 – A method definition: inputs and outputs are bound to feature types.

Networks are used to build a feature processing chain by specifying a directed graph. The nodes define processing steps, i.e. method instances, and the edges define the flow of features between them. Edges connect one output of the predecessor node's method to one input of the successor node's method. As methods do not have optional inputs, each input of all used method instances must be target of an edge. Outputs of method instances can be connected to an arbitrary number of inputs of other methods. The 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 processing chain's input. Sinks provide the analogue for the algorithm's output. Beside composing the processing chain, the network is also used to ensure 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, e.g. thresholds and other arguments used by feature extraction methods, from query-relevant parameters, e.g. the selection of a query image.

Queries are completely parameterize experiments, i.e. they define all input data for the invocation of the respective processing chain. This is typically done by the end-user at query time. Consequently, the underlying processing chain is ready to be executed.

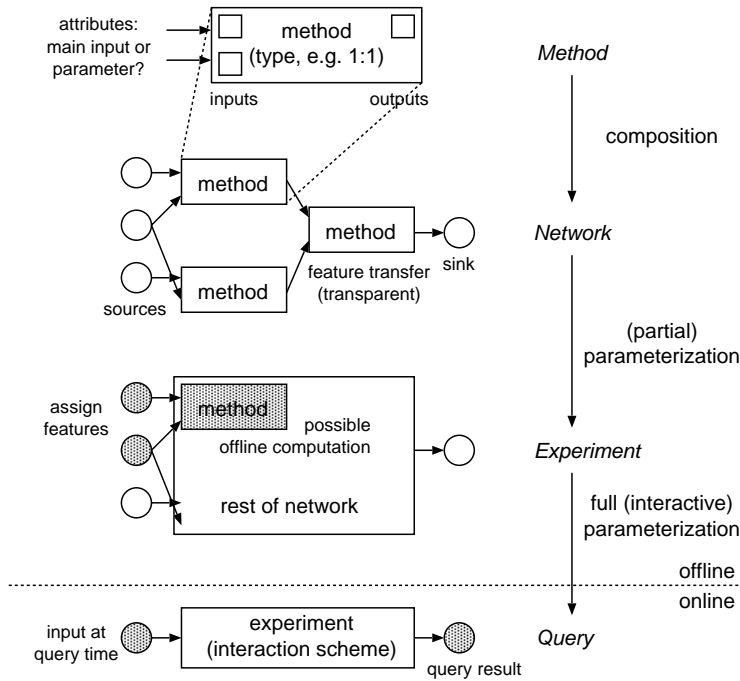


Figure 3 – Components used for the modelling processing chains. Hatched elements symbolize the assignment of features to entities and the induced dependencies.

Figure 3 displays the elements of the algorithm model, their relationships and the implications for running retrieval algorithms built on this model: Networks can be represented as directed graphs and the methods' dependencies from source nodes can be analysed. Thus, it is possible to determine parts of the processing chain which depend on user input at query time and parts which don't. The latter parts can be computed offline in advance.

Web-based interfaces

All end-user interfaces to the retrieval system are JAVA applications or HTML dialogues implemented in PHP⁶. Therefore, the end-user only 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 demand more sophisticated viewing applications, for example hierarchical attributed region-adjacency graphs (HARAGs), which are used for the representation of prominent image regions on different levels of detail. [14]. Therefore, the PHP script can also generate links to a JAVA application that is used for this feature type. They are integrated into the web pages via JAVA WebStart. The JAVA application can access 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 the retrieval algorithm to the end-user. Since each source and sink carries feature type information, it is possible to abstract the user interface from a particular processing chain. 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. 4.

⁶ <http://www.php.net>

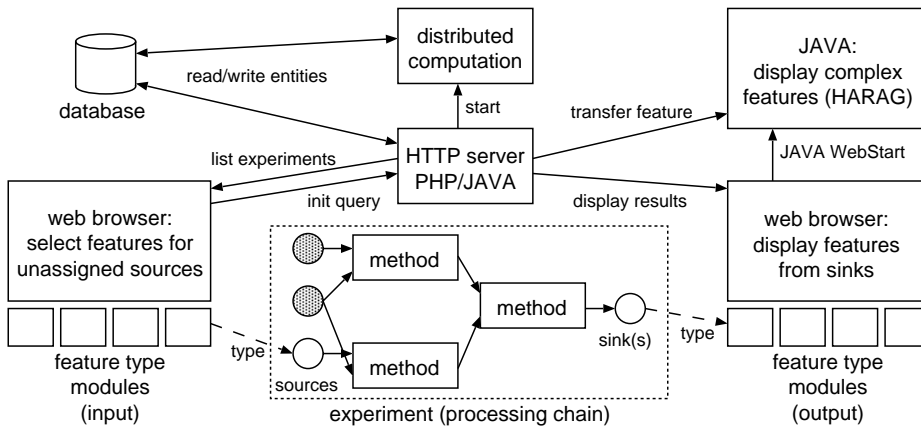


Figure 4 – Schematic view of the system components used for the retrieval process.

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. Direct dependencies from other projects and external libraries must be provided by the developer via the project's configuration file. Known external libraries can be accessed by a project by their symbolic names. Their configuration and integration into the build process is maintained in a central component in the build system and is transparent to the method developers. The dependencies are automatically unrolled by the build system at the time of the compilation. The locations of resulting binaries are automatically determined according to the current platform (processor architecture, operation system, compiler version) and the selected build version (debugging, profiling or release). The build system is implemented on top of the GNU make utility and uses the GNU compiler collection (GCC⁷).

⁷ <http://www.gnu.org/software/gcc>, <http://www.gnu.org/software/make>

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, complex feature data, and source code are stored on file servers and the database only tracks their location.

For query processing, a grid-like architecture of one central scheduling process and daemon processes on each participating host is used.

Once a query is started, the required method calls are automatically determined by the scheduling service based on the input features for the processing chain. 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 a protocol based on the transmission control protocol (TCP), which handles registering/unregistering of daemon processes, command transfer and the transfer of features between the database and the daemons. Feature data outside the database is accessed via the network file system (NFS) for locations inside the local area network (LAN) and via the file transfer protocol (FTP) for locations outside the LAN. Before calling the method, the daemon fetches the input feature tuple. At invocation, the method accesses the features through objects, which provides full location-transparency. The method functionality is contained inside a shared library which is dynamically loaded by the daemon on demand. The library name is determined by the method's name and the name of the transformation function to be called is determined by the method's type. The method function is executed inside a child process, which prevents the whole daemon process from crashing in case of faulty methods. Once the method's computations are complete,

the daemon stores the resulting feature data at the locations specified by the scheduler. Figure 5 displays the invocation of a 1:1 method from the processing chain during a query. Note that the execution of T:1 and 1:T methods differs slightly from this, because these methods need to receive multiple input tuples or produce multiple output tuples, respectively. This repeatedly requires feature transfer between the scheduler and the daemon.

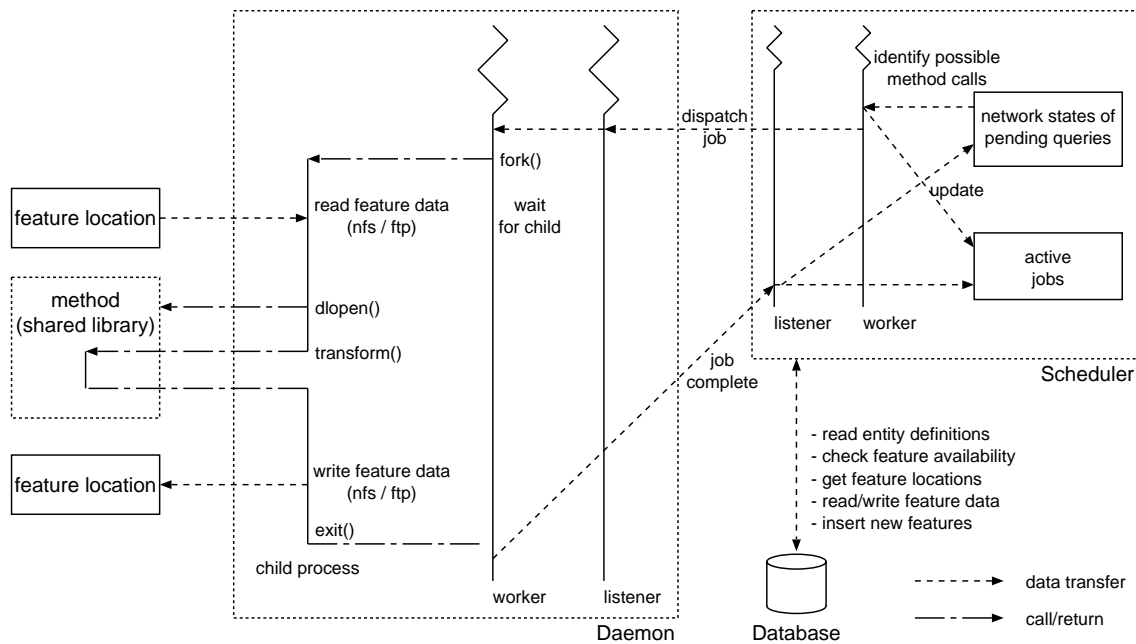


Figure 5 – Execution of a method during a query. Jobs encapsulate the method call command, input features, and specify locations for output features.

The database also stores each feature’s generation history, so that already computed features can be identified during query processing. Such computations are skipped by the scheduler, which then propagates the existing features to succeeding nodes in the network.

3. Results

The system was implemented using open standards and free software (GCC, PostgreSQL⁸ database management system, Apache web server⁹, PHP) and currently runs on Intel/Linux and Sparc/Solaris platforms. At this time, the systems holds 16,953 images from the Department of Diagnostic Radiology, Aachen University of Technology, Aachen, Germany, and 2,588 dental radiographs obtained from the Department of Oral Maxillofacial Radiology, Göteborg University, Göteborg, Sweden. The system was used for reference coding as well as to carry out experiments using global features for the categorization of medical images [15]. The contents of these images are continuously encoded by radiologists using a hierarchical coding scheme [16], which provides the ground truth for query experiments. At the moment, the history of reference coding logged 30,435 interactions of physicians with the system in this process. The system also stores more than 135,000 other medical and non-medical images for evaluation purposes. A QBE demonstration including extended query refinement [14] is available on the internet¹⁰. Beside the query interface, there are currently ten web-based user interfaces, which support all important administrative operations like image import, image export, and the content encoding. An extensive object-oriented and template-based PHP library was developed to support the development of the graphical user interfaces and the invocation of JAVA applications. The feature type concept and the method definition allow it to use a method in arbitrary context compatible with the method's interface. Existing processing chains can be modified without changing the method implementations. For example, additional pre-

⁸ <http://www.postgresql.org>

⁹ <http://httpd.apache.org>

¹⁰ <http://irma-project.org/onlinedemos.php>

processing steps can be integrated by modifying the network entity. A parameterisation can be altered by copying and modifying the experiment entity. The framework can be extended at run-time without re-compilation or other downtime.

The experiment entity is used to connect retrieval algorithms to a user interface. The remaining unassigned inputs of the underlying network are semantically mapped onto data provided by the user at query time through the query interface. All currently implemented retrieval algorithms use the QBE paradigm with optional relevance feedback and query refinement cycles. Furthermore, the ability to upload query images is provided. The possible combinations of these functionalities result in four interactions schemes. Thus, four variations of a QBE retrieval interface were implemented and can be used for accessing compatible experiments. The end-user is not involved in maintenance or configuration tasks.

The method implementation only concerns the feature transformation itself, but the feature access and concurrency at run-time are completely transparent. All features are accessed via objects at runtime. Simple pre-processing steps like image filters can be implemented and deployed within minutes. Currently available methods include pre-processing steps like de-noising, extraction of a variety of global features for the categorization step and the respective distance measures, the extraction of local features and the extraction of HARAGs using a region merging approach. In all, there are over 30 methods available regarding feature extraction and classification, plus 20 methods used for pre-processing and auxiliary functions. The build system makes platform-specific details and dependencies from other projects transparent to the programmer. The deployment of a processing chain is done by inserting an experiment entity. If a compatible PHP interface for an experiment's interaction scheme exists, the retrieval

algorithm behind the experiment is directly available within the web-based query interface.

The run-time environment can effectively utilize the computation power of a set of computers. For the feature extraction on a set of images, first experiments with ten identical stations yielded a speedup factor of 8.36. More precisely, 549 images were used in a two-step feature extraction process: a pre-processing step followed by the extraction of a global texture feature. This results in 1098 jobs, i.e. invocations of 1:1 methods. Fig. 6 shows the resulting speedup for one up to ten workstations (each running one daemon).

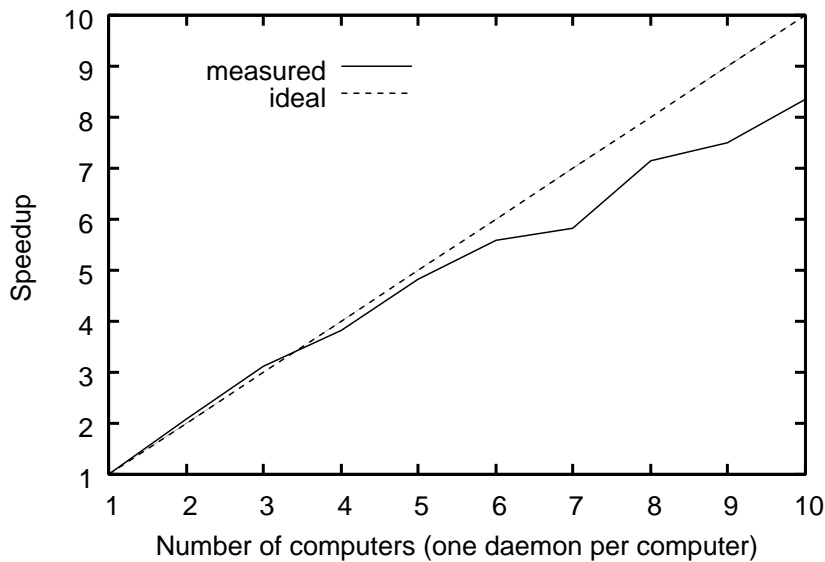


Figure 6 – Measured speedup for a two-step feature extraction processing chain

4. Discussion

The primary benefits from the proposed framework result from the strict separation of all system entities, which interact via well defined interfaces. For each new entity, its creator must provide a proper documentation. Afterwards, the documentation allows all

participants to use the entity. This simple, yet effective concept is employed on all levels. In particular, it supports the requirements of a medical CBIR framework.

Extensibility and flexibility

The method and network entities provide a flexibility for the design of retrieval algorithms which is comparable to component-based image processing development tools, e.g. the Cantata visual programming environment. Using the feature type and method concept, it is further possible to uniformly evaluate retrieval algorithms, and to post-process retrieval results by concepts of classifier combination [17]. The system could also be easily extended: For the analysis of dental radiographs, the coding scheme was extended via the web-based administration interface. Other interfaces access the coding scheme from the database and therefore did not require any modifications. To handle the dental images, a method for the automatic cutting of multi-field radiographs was added. A second installation of the system is also used for the content analysis of image sequences from sewer tunnels, which required additional methods, three new web-based interfaces to handle the sequence semantics, and a new encoding scheme for classifier results and ground truth. It currently contains 52,814 images from 325 sequences.

Separation of retrieval algorithms from their interfaces

The coupling between the retrieval interface and the processing chain is well defined through the experiment entity and the feature type entities. Thus, most retrieval algorithms can be presented uniformly to the end-user. The web-based interfaces to the system, especially for reference coding, enabled the cooperation of computer-scientists and physicians despite the geographical distance between Aachen, Germany and Göteborg, Sweden.

Development and deployment

The implementation of retrieval algorithms uses three layers of abstraction. First, the method entity is used to implement new feature transformations at code level. Second, method entities can be composed into a processing chain by defining a network entity. Thanks to the methods' inherent documentation of their feature interfaces, the composition requires no manipulation or knowledge of the method's source code. The third layer is the parameterization, which is done via the experiment entity and utilizes the network's feature interface and the feature type entities. The latter two levels open the development of retrieval algorithms to non-technical participants. The entities' interfaces also help to establish a standardized documentation, which makes the communications among the interdisciplinary group of developers more efficient. At code level, 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. A new processing chain using an existing user interaction scheme (e.g. QBE) can be deployed by defining an experiment entity which fits the scheme. This effectively separates the fields of interface programming and image processing, saves programmer resources, and speeds up testing cycles as the developer has instant access to the image data.

Efficiency at run-time

Compared to monolithic systems, the more general modelling approach for the retrieval algorithms causes a certain overhead. However, the offline distributed processing scales well for common feature extraction steps and is primarily limited by the bandwidth of the file server holding the feature data. At query time, the feature organization in the database identifies already existing output tuples for method calls, allowing to skip

unnecessary re-computations. Furthermore, the daemon's protection mechanisms against possibly unstable methods implementations permit the simultaneous use of a system installation for production and development.

Comparison to existing systems

Concerning the feature type concept, distributed processing and overall flexibility, several aspects of the system described here are also found in GridPACS, which stronger emphasizes the optimization of handling very large amounts of data, whereas our focus lies on the development aspects of content-based image retrieval and their deployment in general. Our framework is also more flexible thanks to the T:1 and 1:T method types and the unified handling of both offline feature extraction and online feature evaluation and comparison. The de-composition of processing chains into transformation steps (methods), their composition (networks), and their parameterisation (experiments and queries) effectively support the development aspects of CBIR.

5. Conclusion

The proposed framework 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 learning.

Acknowledgement

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

References

- [1] 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* 2002; 24(8): 1026–1038
- [2] Montagnat J, Breton V, Magnin IE: Using grid technologies to face medical image analysis challenges. *Proceedings of the Third IEEE ACM International Symposium on Cluster Computing and the Grid* 2003; 588-93
- [3] Foster I, Kesselman C, Tuecke S: The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications* 2001; 15(3): 200-22
- [4] Deelman E, Blythe J, Gil Y, Kesselman C, Mehta G, Vahi K, Blackburn K, Lazzarini A, Arbre A, Cavanaugh R, Koranda S: Mapping abstract complex workflows onto grid environments. *Journal of Grid Computing*, 2003; 1(1): 9-23
- [5] Frey J, Tannenbaum T, Livny M, Foster I, Tuecke S: Condor-G: a computation management agent for multi-institutional grids. *Procs 10th IEEE International Symposium on High Performance Distributed Computing* 2001; 55-63
- [6] Montagnat J, Breton V, Magnin IE: Partitioning medical image databases for content-based queries on a grid. *Methods of Information in Medicine* 2005; 44(2): 154-160
- [7] Hastings S, Oster S, Langella S, Kurc TM, Pan T, Catalyurek UV, Saltz JH: A grid-based image archival and analysis system. *Journal of the American Medical Informatics Association* 2005; 12: 286-95
- [8] Hastings S, Kurc TM, Langella S, Catalyurek UV, Pan T, Saltz JH: Image processing for the grid: a toolkit for building grid-enabled image processing applications. *Proceedings of the Third IEEE ACM International Symposium on Cluster Computing and the Grid* 2003; 36-43
- [9] Tagare HD, Jaffe CC, Duncan J: Medical image databases – a content-based retrieval approach. *Journal of the American Medical Informatics Association* 1997; 4: 184–98
- [10] 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.
- [11] Lehmann TM, Güld MO, Thies C, Fischer B, Spitzer K, Keyzers D, Ney H, Kohlen M, Schubert H, Wein BB: Content-based Image Retrieval in Medical Applications. *Methods of Information in Medicine* 2004; 43(4): 354-61
- [12] Müller H, Michoux N, Bandon D, Geissbühler 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
- [13] Güld MO, Thies C, Fischer B, Keyzers D, Wein BB, Lehmann TM: A platform for distributed image processing and image retrieval. *Procs SPIE* 2003; 5150:1109-20
- [14] 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
- [15] Lehmann TM, Güld MO, Deselaers T, Keyzers D, Schubert H, Spitzer K, Ney H, Wein BB: Automatic categorization of medical images for content-based retrieval and data mining.

Computerized Medical Imaging and Graphics 2005; 29(2): 143-155

[16]Lehmann TM, Schubert H, Keysers D, Kohnen M, Wein BB: The IRMA code for unique classification of medical images. Proceedings SPIE 2003; 5033: 109-117

[17]Jain AK, Duin RPW, Mao J: Statistical pattern recognition: a review. IEEE Transactions on Pattern Analysis and Machine Intelligence 2000; 22(1): 4-37

8. Address for Correspondence

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

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