Synergetic impact obtained by a distributed development platform for Image Retrieval in Medical Applications (IRMA)

Jörg Bredno[a], Michael Kohnen[b], Jörg Dahmen[c],
Frank Vogelsang[b], Berthold Wein[b], and Thomas Lehmann[a]

[a]Institute of Medical Informatics, Medical Faculty
[b]Department for Diagnostic Radiology, Medical Faculty
[c]Institute for Computer Sciences VI, Faculty of Computer Sciences

Aachen University of Technology (RWTH), D–52057 Aachen, Germany

ABSTRACT

Image retrieval in medical applications (IRMA) requires the cooperation of experts in the field of medicine, image analysis, feature analysis and systems engineering. A distributed developing platform was implemented to support the progress of the IRMA–system. As the concept for this system strictly separates the steps for medical image retrieval, its components can be developed separately by work groups in different departments. The development platform provides location and access transparency for its resources. These resources are images and extracted features as well as methods which all are distributed automatically between the work groups. Replications are created to avoid repeated network transfers. All resources are administered in one central database. Computationally expensive feature extraction tasks are distributed also automatically to be processed on concurring workstations of different work groups. The developing platform intensifies and simplifies the cooperation of the interdisciplinary IRMA–development–team by providing fast and automated deliveries of components from software developers to physicians for evaluation.

Keywords: Distributed system, interdisciplinary development, automated method transfer, access transparency, concurrency transparency, replication transparency

1. INTRODUCTION

Many different content–based image retrieval (CBIR) systems are available to handle queries in large image archives.[1] These CBIR–systems use different features which are extracted to describe images in a reduced dimension and different distance or similarity measures which are used to perform queries on the archives. Examples for features are color,[2,3] texture[2,4] and shape.[5,6] Only few CBIR–systems were conceived to work with medical archives.[7,8] For medical image archives, the extracted features must reflect medico–diagnostic image contents.[9,10] A new approach to image retrieval in medical applications (IRMA) strictly separates the steps of a CBIR–system according to the medical knowledge introduced during image processing.[10] These steps are (1) automated categorization of images using global features, (2) computation of registration parameters compared to a prototype of the image category, (3) extraction of local feature images, (4) selection of local features according to a query and the image's category, (5) indexing of the images using a hierarchical blob–tree as data structure, (6) identification of morphological structures, and (7) retrieval based on the blob–trees. The categorization describes the imaging modality, the region within the body, and its orientation.

The development of IRMA requires the intense cooperation of co–workers with different competences. On the one hand, medical image interpretation is a complex process[9] which is not reproducable by software developers. On the other hand, physicians mainly do not have the knowledge to design and implement computer algorithms. Therefore, the development of the system needs synergetic cooperation of experts in the field of medical image interpretation, image analysis, automated classification, and systems engineering.

Further author information: Send correspondence to
Jörg Bredno,Institute of Medical Informatics, Aachen University of Technology, D-52057 Aachen, Germany
E-mail: jbredno@mi.rwth-aachen.de

The IRMA–developers belong to local groups in different departments, the IRMA–team consists of *co–located groups*.[11] The groups use different hardware, operating, and file systems connected by internet services. Distributed development under these circumstances requires that each group has a complete development site consisting of a file server and a number of workstations.

Responsibilities for different components of the system are assigned to specific sites. All sites have access to replications of the latest released components of other groups. To minimize the risk of conflicts, strictly defined interfaces are required for these components.[11]

The components of the IRMA–system are methods to extract global and local features from images, methods to compute distance or similarity measures for global features, and methods to classify images by their feature vectors. The interdisciplinary knowledge is introduced into the system with the variety of different methods. The development platform as well as the IRMA–system itself must be highly generic and flexible as the number and kind of features in use are subject to continuous evolution.[10] Therefore, an automated transfer of the methods is needed to distribute this knowledge between the co–located groups.

From the viewpoint of distributed systems,[12] the development platform must provide the following levels of transparency:

- Location and access transparency is needed for images, features and methods. These important resources of the IRMA–system must be accessed from all co–located groups with identical operations regardless of their physical location.
- As the continuous transfer of files over the internet is not possible, a replication transparency must be provided for these resources. The system should automatically create the local replications and keep track of updates.
- Image analysis involves algorithms with high computational costs. As lots of images from the archive have to be processed, a concurrency transparency is needed to share image analysis between different workstations and groups.

2. THE IRMA–DATABASE

The IRMA–development–platform is based on the transparent distribution of its resources to the independent development sites. These resources include images, extracted features and methods which are administered in the IRMA–database (DB). The development sites access one central database server using SQL statements transferred over the internet. The access to this IRMA–DB is allowed for acknowledged users who try to connect from computers belonging to one of the developing groups. The user names are login names accepted by the user authentification of the trusted development sites.

The IRMA–DB contains the tabular data with entries organized by unique identifiers (IDs) for images, image sources, image formats, extraction methods with their parameter sets, and computer systems involved in the development platform. For images, the filenames, their properties, and the link to image sources is stored. Manually assigned categories provided by radiologic experts are linked to these entries. Image properties e.g. are the size and dimension of images. These properties and the image's value representation are read from the image file header and stored in the IRMA–DB to avoid frequent opening of files. The development platform describes the value representation by the number of values per image pixel and the number of bytes stored for each value as these characteristics are needed to determine whether an extraction method is applicable on a given image. Images from varying medical imaging modalities are used in the IRMA–system. Therefore, no standard value representation was defined.

Methods are administered by their descriptive name, the name of the executable, and a parameter set to control its behavior. For each executable, a list of source files is provided. Images and methods are connected either by global feature vector components and local feature images which are specific for each image and extraction method, or by a worklist that contains all the entries for images that have to be processed by these methods.

Workstations are defined by their hostname and belong to one development site. Each development site provides a list of available image sources which are full pathnames, and method sources which are located in defined subdirectories.

The basic tables of the IRMA–DB are shown in Fig. 1. The relations between the tables indicate one–to–many (1–n) and many–to–many (n–m) relations. E.g., each image may be found in different image sources, whereas every image source is assigned to exactly one development site. One–to–one relations are organized as columns in the same table.

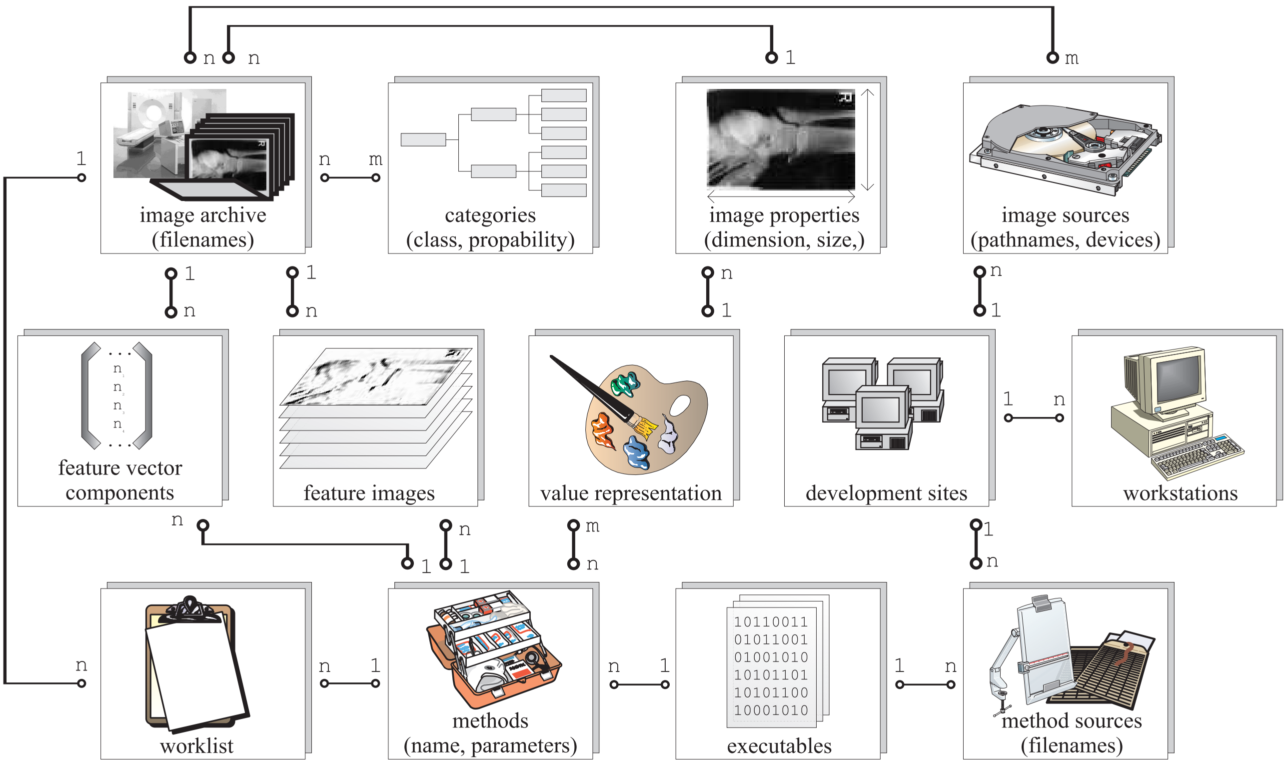

Figure 1. Schema of the main tables of the IRMA–DB.

3. DISTRIBUTABLE RESOURCES

The cooperation of the IRMA–developers is intensified by the automatic transfer of different resources. A resource is distributable for IRMA if access and location transparency have been implemented. If these resources are created or inserted in one development site, they automatically become accessible from all other sites as well. The IRMA–DB itself provides access transparency to all entries in DB–tables, but images, feature images and methods are stored as files. In the case of methods, it is not even sufficient to transfer these files as the different development sites need not to be binary compatible.

3.1. Images

The IRMA–system has to manage and file a large amount of various image data. The images themselves are taken from heterogenous imaging sources. Beside DICOM data, secondary digitized images from film-based and other modalities have to be processed. The heterogenous image material is stored in different locations which are administered in the IRMA–DB. The file format, size, image type, value representation, and the category are used to decide which images and feature extraction methods can be combined. These properties are stored in database tables and can be read without accessing the image file. Algorithms that extract features applicable only for specific image categories can be combined with appropriate images.

Image sources are referenced by the development site where they are stored and the corresponding path name. An image can have several sources because replications may exist on different development sites. These entries ensure the availability of images and enable location transparency. To access an image, first its source entries are read from the IRMA–DB. If a replication of the image exists on the demanding development site, the local file is opened. Otherwise, an automated FTP-process is initialized. The internet address of according file servers can be read from the IRMA–DB. The user is prompted for the password and the image is transferred onto the local development site. Here, the file is stored temporarily and is discarded or copied to a defined location. Then, the existence of this replication is documented in the IRMA–DB.

3.2. Features

Global features describe the contents of images in a dimension which usually is reduced compared to original image size, local features contain one feature value for each pixel of the image. The IRMA–system uses global feature vectors, local feature images and blob–trees as content–based abstractions.

- Global features are extracted once for every image by different methods. Because this process might be very time consuming, the result is stored in the IRMA–DB. Each feature vector is composed of float values. Its components are stored with a reference to the originating image and the extracting method. The global feature vector of one image consists of one or more components generated by each extraction method. When feature vectors are inserted into the IRMA–DB, they are available from all development sites. The SQL–database itself provides the access transparency. Standard distance measures for vectors (e.g. Cartesian or chessboard distance) are provided by the system. If these distance measures are not sufficient for classification, own methods to sort or classify images according to their feature vectors can be newly introduced by the developers.
- The IRMA–concept uses local features which are extracted for each pixel, e.g. to describe texture. Hence, the extraction of local features results in feature images. These images are handled as files and are administered in the IRMA–DB. The files are stored in predefined image source directories and accessed using the ID of the originating image and the ID of the extraction method. The implementations for access and replication transparency of original images are also used for local feature images.
- Blob–trees are internally handled like feature vectors. They are stored in the IRMA–DB in the same tables which contain the global feature vector. The two data structures are distinguished by the ID of the method which created these entries. Methods for the extraction of blobs and for the computation of distance measures between blob–trees use a predefined coding of tree structures in feature vectors.

3.3. Methods

The major property of an CBIR–system is the possibility to quantify visual and content based similarities. In medical applications, several different content–descriptions for images coexist. Examples for content descriptions are the image's category, the appearance of different morphological structures, or a finding. In IRMA, the hierarchical blob representation of images is the most important content description that has to be extracted from images.

Different methods extract features to reflect the medical contents of the image. A binary distribution of methods was not applicable because the project partners use different computer systems. Therefore, these methods are implemented using a high level of abstraction allowing source compatibility for all systems in use. As soon as a feature extraction method's implementation has reached a β-level, it has to be evaluated by a physician regarding its ability to extract medically relevant features. This step corresponds to the release and delivery of components in the sense of software configuration management.[11] The evaluation requires the immediate installation of new methods onto target systems for medical users. Therefore, the installation is performed by the system without user interaction.

A method is the combination of an executable with a set of parameters. Hence, one executable can provide different methods. For every executable, a subdirectory contains all source files and the binary file itself. Access to execute a method was implemented using access and replication transparency. Whenever a user selects a method from the IRMA–DB and the executable has not been installed on the target system so far, an automated transfer is started. The development platform automatically establishes a FTP-connection to the development site where the sources are available. These sources are copied onto the target system at the appropriate location. Further, the platform automatically generates a Makefile for the transferred sources and initiates the compilation and linking process. The user of the algorithm does not need any knowledge about those operation system dependent procedures. This knowledge is stored in one system-wide Makefile and another simple initialization file. A local replication of the executable then exists on the development site.

One executable might contain different methods which are distinguished by different sets of parameters stored in the IRMA–DB. The user chooses a method by its descriptive name. The executable and its parameters are read from the IRMA–DB. All distributable methods refer to the same main routine. This main routine initiates the connection to the IRMA–DB and reads its the parameter set. According to these parameters, different functions are called:

- To extract global features from images, the routine reads a list of images to be processed, opens these images, allocates storage for the components of the global feature vector, and then calls the function which computes the components. Afterwards, the results are stored in the global feature vector of the image in the IRMA–DB.
- The extraction of local feature images is called with the original image and an empty feature image where the local features are to be set. The resulting image is saved in a defined position with a filename that codes the original image and the extraction method. The image is inserted into the IRMA–DB and transparently accessible.
- The extraction of the blob–tree is handled like a function to extract global feature vector components. This extraction method requests local feature images which may vary due to different queries. Therefore, the method might initiate the automated extraction of local features, if the resulting feature image is not available so far.
- To categorize images or to handle a query, distance measures for the blob–tree or for components of the global feature vector are needed. The calculation of different distance measures is stored in methods. The main routine reads the appropriate components of the global feature vector for the image. These are either compared to the global feature vector of category prototypes for categorization or to blob-trees of images in the IRMA–DB to find query results. The function itself computes the distance between the relevant components of the feature vector. These distances are sorted by the main routine to find the results of the query or the image's category.

The functions described above are read from libraries which are individually linked to the main routine to create the executable for one or more methods. The different methods are distinguished by predefined function names in the library that is created for every executable.

4. FEATURE EXTRACTION DAEMON

Feature extraction methods determine global feature vectors or local feature images for a large amount of image data. This process can take a considerable amount of time. During development of such an algorithm, the evaluation is an important step and requires application on many images. Whenever new images are introduced into the IRMA–DB, all components of the global feature vector are needed for categorization. The computational time is reduced by distributing the extraction of features for many images to different computers and development sites. Feature extraction daemons can be started on different workstations in all development sites. These daemons run with low priority, so that user processes on the workstations are not disturbed. Each daemon polls the worklist in the IRMA–DB. If any entries exist there, the daemon reads one entry from the list and starts the according feature extraction method with a predefined command line parameter. The existence of this parameter leads to a new behavior of the main routine for feature extraction methods:

- The main routine reads only one image from the worklist each time and deletes its entry.
- Thereafter, the main routine tries to open this image. If the image has to be transferred via ftp, its ID is re–inserted into the worklist because the daemon does not have required password legitimation. The IDs of such images are inserted into a local list of images which cannot be accessed by the daemon.
- Otherwise, the feature extraction method is started, and computed features are stored.
- The feature extraction method returns to the calling daemon if all remaining images in the worklist are not accessible without FTP passwords.
- A method will not be restarted by the daemon unless its worklist was once empty.

If the worklist contains no entries, the daemon sleeps for a fixed amount of seconds before the next database poll. The feature extraction daemons offer a concurrency transparency to share not only the images but also the distribution of tasks on different computer systems without user interaction.

The start of the feature extraction method by the daemon is documented in a database table. If a database query is initiated which requires new feature extractions, the locally started feature extraction may return, even though instances of the same method may still be running on other workstations. The start of the computation of distance measures is delayed until the daemon table of the IRMA–DB contains no more instances of the feature extraction method.

5. DEVELOPMENT SITE

A development site is located in one cluster of workstations or PCs. Its installation adds new executable tools as well as libraries which can be used by developers. The tools and libraries have been written in C++. New methods for global or local feature extraction and the computation of distance measures are also developed in C++.

5.1. Tools to Interact with the System

The development platform is controlled by command line executables. These tools handle the interaction with the database server and the execution of methods.

- Whenever new images are inserted into the development platform, *InsertImages* has to be started. This tool gets a pathname and file filter as parameters and then inserts the name, location and properties of new images into the IRMA–DB.
- Methods are inserted into the IRMA–DB when their implementation has reached β-level. *InsertMethod* is used by the developer who has to specify the method's purpose (extract global features, feature images or compute distance measures) and a parameter set that controls the algorithm.
- As the parameters to control methods are stored in the IRMA–DB, *StartMethod* has to be used to start these methods by reading the parameters from the IRMA–DB and -if necessary- install a new method on the development site.
- A choice of images and features can be combined to perform a *primitive query*[10] in the IRMA–DB. *QBE* performs a query to evaluate methods for feature extraction and computation of distance measures. Concepts for the IRMA–system also include *semantic queries* and *browsing* to retrieve images from the DB.
- The image archive is permanently enlarged and changed. New methods are introduced or replace older versions. *Consistency* checks the consistency of all tables in the IRMA–DB and deletes or corrects illegal entries. This can also initiate the removal of image files or extraction sources and executables together with their replications if entries are deleted from the IRMA–DB.
- The feature extraction daemon is also an executable tool of the development site.

5.2. Evaluation Interface

The tools that are used to work with the IRMA–system are usually started from the command line. This is not acceptable for physicians. Therefore, a prototype of a graphical user interface has been developed as a front–end to the IRMA–system. This evaluation interface supports the user in browsing the images in the database, the selection of methods, and their combination to perform a *primitive query*. Afterwards, it presents the results. If necessary, the automatic distribution of resources is also initiated by this application. The evaluation interface is running on workstations which are part of a development site. It does not fulfill the requirements of a radiologists workstation to view radiographs and perform findings.

5.3. Installation

The number of co–located groups and development sites is not limited by the development platform. A development site can be installed on workstation clusters of a co–located group if the cluster fulfills the following specifications:

- An IRMA–user with FTP privileges must have an account on this cluster.
- All workstations must have access to at least one central fileserver using the same pathnames.
- The workstations in each cluster must be binary compatible.
- The client libraries for SQL–database access must be installed.
- For the evaluation interface, which is optional, the appropriate API for the graphical user interface is required.
- Because of the automated installation of methods, the Make–utility, a C++–Compiler and a linker must be available for all IRMA–users.
- Different image format filters are included from standard or optional libraries. The platform also is able to start executables for image format conversion automatically if a library to open the current image file is not available.

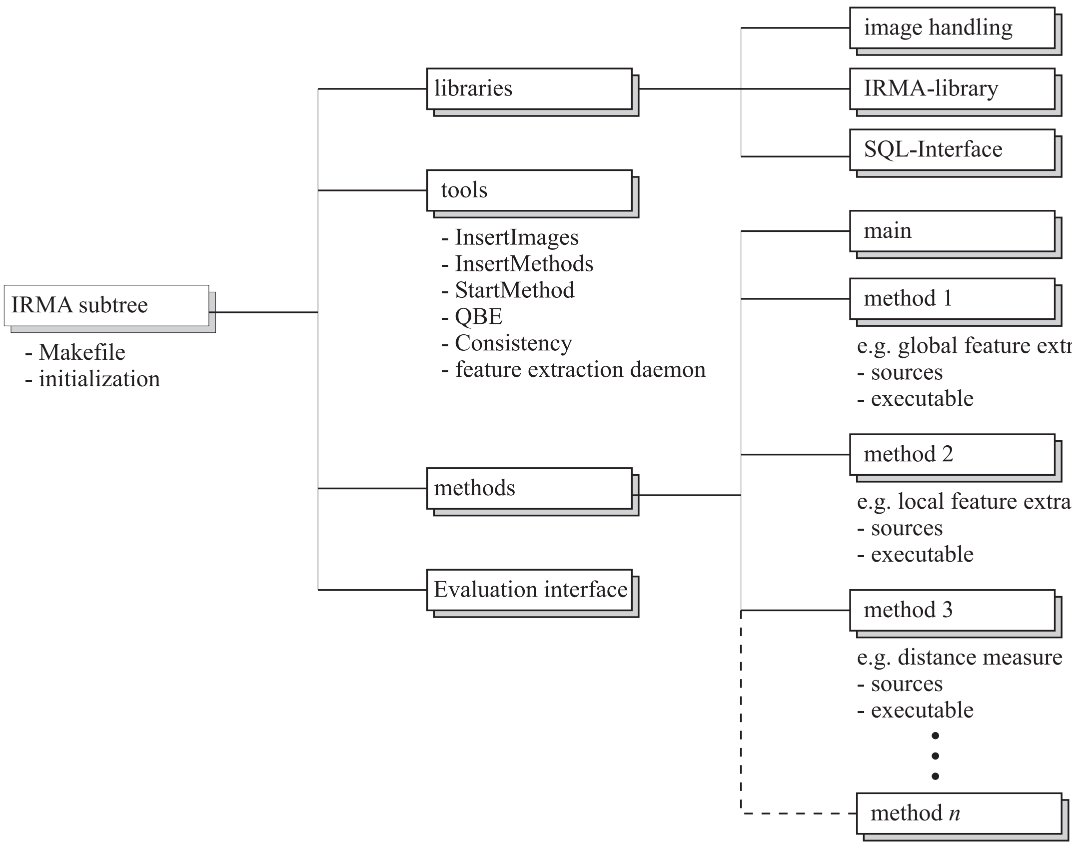

Figure 2. The installation subtree of a development site.

A development site is installed in a defined subtree of a file system and exported to all workstations belonging to this cluster. Subdirectories exist for the tools, the executables of methods with their sources, and libraries. The libraries are subbranched into basic functions to access images, the SQL-interface to the database and IRMA–specific libraries. The optional evaluation interface is placed in an own branch. Each executable for methods is placed in a subdirectory in the method branch (Fig. 2). One of these subdirectories contains the main routine of all methods which acts as an interface between the IRMA–system and the individually developed feature extraction and distance measure methods. Images can be stored in arbitrary locations if these are accessible from all workstations in the cluster.

Two basic files describe the individual development site:

- A text file in the root directory of the subtree contains a simple key–value database with few entries for initialization. Here the connection to the IRMA–DB server, the location of executables for methods and temporal image file space is defined. The file also contains the user name of an IRMA–user whose login is used when automated FTP–connections between different development sites are established to distribute resources.
- A Makefile in the same directory contains all information needed to create executables that are used by the IRMA–system.

These two files have to be edited whenever a development site is newly installed on a cluster of workstations. The expert knowledge needed to create the libraries and executables of the IRMA–system is preserved here and later used to create new methods automatically.

6. RESULTS

6.1. Establishment of Development Sites

The development site has been installed in three different departments of the Aachen University of Technology. Several PCs using Linux and NetBSD, Sun Sparc and Ultra computers with Solaris, SGI computers with Irix, and DEC alphas are connected for the IRMA–project. Creating a new development site takes less than two hours if it is performed by an expert who is familiar with both the target cluster and the IRMA–system. The documentation, which is written in German so far, allows a user to become familiar with the system in about an hour. After the installation of the platform, all knowledge needed to create and start new methods is existent in the system itself.

The IRMA–archive for development is planned to consist of 10.000 images with a distribution of categories that reflects the original distribution of image categories of the Department of Diagnostic Radiology. So far, 2071 in vivo images where arbitrarily chosen and inserted into the system. The most frequent images are film based radiographs, images from computed tomography, magnetic resonance imaging, ultrasound and nuclear medical imaging are also available for developers (Tab. 1). These images were acquired from all regions of the human body (Tab. 2) with varying orientations. Images are continously digitized and anonymized to extend the archive.

Table 1. Images available for development, sorted by imaging modality.

modality	# of images
X–ray	1776
Computed tomography	181
Magnetic resonance imaging	88
Ultrasound	24
nuclear medical imaging	2
sum	2071

Table 2. Images available for development, sorted by primary visible body region.

region	# of images
upper limb	387
lower limb	354
pelvis	70
spine	268
abdomen	266
chest	454
scull	167
breast	103
total body	2
sum	2071

6.2. Development of Methods

A short reference which describes access to image data as well as to to feature vectors is provided, so that C++–programmers can directly start to implement new methods. Programmers do not need to open image files or interact with the SQL–database themselves. A standardized class for images of all file types is passed to all extraction methods. The feature vector has its own class with functions to add or access entries. On return of the global feature extraction function provided by the developer, the features are inserted into the IRMA–DB automatically and used for queries or classification tasks. Local feature extraction methods set image values into an initialized feature image. For the categorization of images or retrieval using blob–trees, a method has to assign distance measures to components of feature vectors. The system itself sorts these measures by size and selects the query results or the category. Even for new methods, the standard steps for the above tasks need no re–implementation. Only the characteristic properties of the methods are implemented with minimal overhead.

The following methods are implemented already:

- Extraction of textural features using the similarity of co–occurrence matrices by a synergetic classifier.[13,14]
- Detection of the contour of body parts on radiographs using a deformable model.[15]
- Extraction of Fourier coefficients, invariant moments[5,16] and semi–local invariant signatures[6] of contours.
- Detection of the scribor or other text parts superimposed in radiographs.
- Extraction of dominant regions from images using region–growing to create blob data structures that describe their location, size, shape, and texture.
- Extraction of color histograms using the histogram intersection as a similarity measure.
- Calculation of transformation distance measures[17] after scaling images to a fixed square size.[18]
- Computation of vertical and horizontal projections of the image and the extraction of Fourier descriptors for these projections.

6.3. Automated Transfer of Methods

Whenever a new method is implemented, its existence is documented in the IRMA–DB. The source files, which are identified by file filters, are inserted into the IRMA–DB as well. This entry is all information that is needed to install the method on other development sites. This installation is not started by a user. The development site initiates the transfer of sources when the method is selected for execution on another site for the first time.

This aspect is very important if a new method has to be evaluated by a radiologist, who is able to judge the relevance of a method for medical tasks. The FTP-transfer is accomplished in less than half a minute, the creation of the executable takes less than 10 seconds for 1000 lines of method source on a 333MHz PC. Hence, the new method is available in less than one minute. Because all features are stored in the IRMA–DB, the medical user can evaluate a method without significant waiting time. We use conventional communication medias like phone or e-mail for fast feedbacks to the developers. The fast and automated distribution of methods massively improves the synergetic transfer of knowledge between the interdisciplinary IRMA–partners.

The automated method transfer as well as the transfer of images and features is used regularly. The implementation of the developing platform shows the capability of the system to handle the transparent administration and distribution of resources.

6.4. Feature Extraction Daemon

The ability of the platform to speed up feature extraction methods by using feature extraction daemons has been tested. A particular extraction method was suitable for only 50 images from the archive. The method was started on a 333MHz PC with Linux. It has a 10MBit internet connection to the IRMA–DB server. The executable started to search the list of all images to find those where features had to be extracted. This task involves lots of SQL queries and took 4 minutes. When the chosen images were inserted into the worklist, the extraction method itself was started which took 8 minutes and 40 seconds. Altogether, the features were available 12 minutes and 40 seconds after the start of the test.

Then, four feature extraction daemons were started on four Sun Ultra I workstations belonging to an image analysis cluster. The IRMA–DB was reset to its state before the first run, then the method was started again on the PC. The first images that had to be processed were found 1:50 minutes after the start of the test. These images are directly inserted into the worklist. The first feature extraction daemon found these entries and started the according method 2 minutes after the start of the test, the other machines followed 20 to 30 seconds later. The feature extraction method then ran on four machines even while the creation of the worklist was still in progress. At this state, the IRMA–server had to handle SQL–requests from the daemons as well, so that the completion of the worklist took 20 seconds longer than before. Then, the extraction method was started on the first PC as well. Here only two images were left for feature extraction, all features were available 4:30 minutes after the start of the test. Even though the PC itself was the fastest machine to extract the features, the summed computational time for extraction on all five machines was 8:05 minutes only. This acceleration was possible because the workstations in the image processing pool have 100MBit connections and therefore faster access to the images on a disk server in that pool.

7. DISCUSSION

Software developers are often located at several physical sites where each site develops one or more subcomponents of a complete system.[11] Often, a manual source transfer is used to update sources on different sites.[19] This source transfer usually is needed to synchronize parallel development and is carried out either by experienced developers or by software configuration management tools.

For the development of IRMA, physicians, computer scientists, radiologists and engineers are merged within the development team. The synchronization of methods therefore had to be as easy as possible to enable a synergy between software developers and medical experts in the co–located groups. The IRMA–development–platform fulfills the requirements for parallel development[11]:

- The IRMA–system consists of independent components, the methods, which have their own source code and executables.
- The dependency between different departments is minimized. No component of the IRMA–system depends on the existence or correctness of all other components. In particular, not all methods must be available at each site.
- Each of the co–located groups is assigned the responsibility for methods which are developed there. The location of source code stored in the IRMA–DB is always the location of the responsible group.
- The methods are available to other groups by replication transparency. The awareness of changes in the methods is initiated by the responsible developer using conventional communication medias. Here improvements of the development platform will include automated version updates of distributed methods.
- The methods have a strictly defined interface to the IRMA–system.

These qualities of the IRMA–development–platform minimize the risk of conflicts and accelerate the evaluation of methods by medical experts. The platform improves the synergetic cooperation of software developers and physicians. It provides an automated and fast transfer of technical methods into the clinical routine. The software developers without medical knowledge get immediate feedback from physicians judging the relevance of a method for classification or diagnostic issues. Further, these physicians need no knowledge of the internal installation processes. So, it becomes possible to apply methods reflecting the similarity of radiographs according to diagnostic issues. The development platform assists and facilitates the interdisciplinary knowledge transfer that is required for the development of systems for image retrieval in medical applications. Even though the development platform was designed and implemented to reflect the concepts of the IRMA–system, the automated transfer of resources could assist the cooperation of locally separated interdisciplinary development teams in other software–projects as well.

REFERENCES

1. M. de Marsicoi, K. Cinque, and S. Levialdi, "Indexing pictorial documents by their content: A survey of current techniques," *Image and Vision Computing* **15**, pp. 119–141, 1997.
2. W. Niblack, R. Barber, W. Equitz, M. Flickner, P. Yanker, and J. Ashley, "The QBIC-project: Querying images by content using color, texture and shape," *Proc.SPIE* **1908**, pp. 173–187, 1993.
3. J. R. Smith and S. F. Chang, "Tools and techniques for color image retrieval," *Proc. SPIE* **2670**, pp. 426–437, 1996.
4. C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik, "Blobworld: A system for region–based image indexing and retrieval," *Procs. 3rd Int. Conf. On Visual Information Systems* , pp. 509–516, 1999.
5. D. Huang and C. L. Huang, "A content-based image retrieval system," *Image and Vision Computing* **16**, pp. 149–163, 1998.
6. M. Kliot and E. Rivlin, "Invariant–based shape retrieval in pictorial databases," *Computer Vision and Image Understanding* **71**(2), pp. 182–197, 1998.
7. C. Shyu, C. Brodley, A. Kak, A. Kosaka, A. Aisen, and L. Broderick, "ASSERT: A phyisician–in–the–loop content–based retrieval system for HRCT image databases," *Computer Vision and Image Understanding* **75**(1/2), pp. 111–132, 1999.
8. E. Petrakis and C. Faloutsos, "Similarity searching in medical image databases," *IEEE Transactions on Knowledge and Data Engineering* **9**(3), pp. 435–447, 1997.
9. H. D. Tagare, C. C. Jaffe, and J. Duncan, "Medical image databases: A content-based retrieval approach," *Journal of the American Medical Informatics Association* **4**(3), pp. 184–198, 1997.
10. T. Lehmann, B. Wein, J. Dahmen, J. Bredno, F. Vogelsang, and M. Kohnen, "Content-based image retrieval in medical applications: A novel multi–step approach," *Proc. SPIE* **3972**. (this issue).
11. U. Asklund, B. Magnusson, and A. Persson, "Experiences: Distributed development and software configuration management," *Lecture Notes in Computer Science* **1675**, pp. 17–33, 1999.
12. G. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems - Concepts and Design*, Addison–Wesley, Reading, 1994.
13. F. Weiler, F. Vogelsang, M. Kilbinger, B. Wein, and R. W. Günther, "Automatic recognition of image contents using textural information and a synergetic classifier," *Proc. SPIE* **3034**, pp. 985–989, 1997.
14. H. Haken, *Synergetic Computers and Cognition*, Springer–Verlag, Berlin, 1991.
15. V. Metzler, J. Bredno, T. Lehmann, and K. Spitzer, "A deformable membrane for the segmentation of cytological samples," *Proc. SPIE* **3338**, pp. 1246–1257, 1998.
16. R. Gonzales and P. Wintz, *Digital Image Processing*, Addison–Wesley, Reading, 1987.
17. P. Simard, Y. LeChun, and J. Denker, "Efficient pattern recognition using a new transformation distance," *Neural Information Processing Systems* **5**, pp. 50–58, 1993.
18. J. Dahmen, R. Schlueter, and H. Ney, "Discriminative training of gaussian mixtures for image object recognition," in *21 DAGM Symposium Mustererkennung*, pp. 205–212, Springer–Verlag, (Bonn, Germany), 1999.
19. L. Allen, G. Fernandez, K. Kane, D. Leblang, D. Minard, and J. Posner, "ClearCase MultiSite: Supporting geographically–distributed software development," *Lecture Notes in Computer Science* **1005**, pp. 194–214, 1995.