I^2C: A System for the Indexing, Storage, and Retrieval of Medical Images by Content

S. C. Orphanoudakis†‡, C. Chronaki†, and S. Kostomanolakis†‡

†Institute of Computer Science, Foundation for Research and Technology-Hellas

and

‡Department of Computer Science, University of Crete,

Heraklion, Crete, Greece

January 1994

Abstract

Image indexing, storage, and retrieval based on pictorial content is a feature of image database systems which is becoming of increasing importance in many application domains. Medical image database systems, which support the retrieval of images generated by different modalities based on their pictorial content, will provide added value to future generation Picture Archiving and Communication Systems (PACS), and can be used as a diagnostic decision support tool and as a tool for medical research and training.

This paper presents the architecture and features of I^2C, a system for the indexing, storage, and retrieval of medical images by content. A unique design feature of this architecture is that it also serves as a platform for the implementation and performance evaluation of image description methods and retrieval strategies. I^2C is a modular and extensible system, which has been developed based on object-oriented principles. It consists of a set of cooperating modules which facilitate the addition of new graphical tools, image description and matching algorithms. These can be incorporated into the system at the application level. The core concept of I^2C is an image class hierarchy. Image classes encapsulate different segmentation and image content description algorithms. Medical images are assigned to image classes based on a set of user-defined attributes such as imaging modality, type of study, anatomical characteristics, etc. This class-based treatment of images in the I^2C system, achieves increased accuracy and efficiency of content-based retrievals, by limiting the search space and allowing specific algorithms to be fine-tuned for images acquired by different modalities or representing different parts of the anatomy.

KEYWORDS: image management, medical image database systems, retrieval by content, PACS.

1 Introduction

The large number of medical images currently generated by various diagnostic modalities have made not only the interpretation of such images by humans very difficult, but also their management. The use of computer methods in the analysis and interpretation of medical images, as well as in the proper integration of multimodality imaging information is one of the major trends shaping the future of medical imaging. Similarly, computer methods are currently used for the efficient transmission, storage, and retrieval of medical images and image related data [1].

Recent developments in digital medical imaging by various modalities have paved the way for a filmless digital department of diagnostic imaging in the near future. Today, many diagnostic imaging departments make extensive use of computer networks, mass storage devices, and sophisticated workstations at which humans and machines interact, assisted by techniques of computer vision, graphics and artificial intelligence to achieve integration of multimodality imaging information and expert medical knowledge [2].

Computer networks provide physical links to other hospital departments and patient wards in order to improve interdepartmental communications and patient monitoring procedures. Furthermore, medical image databases become easily accessible by physicians and other hospital personnel and all components of a hospital information system (i.e. patient records, clinical support, organizational support, and communications) are functionally integrated.

Image database systems and image management in general are extremely important in achieving both technical and functional integration of Hospital Information Systems (HIS), Radiological Information Systems (RIS), PACS, and Telemedicine Systems due to the technical constraints imposed by the volume and information density of image data, as well as various clinical and legal requirements such as availability of image data where and when it is needed, length of image storage, possible information loss resulting from image transmission, etc. In the emerging new clinical environment of integrated HIS, RIS, PACS, and telemedicine systems, it is possible to extend the capabilities of diagnostic medical imaging techniques, as well as image management and communication systems, by developing medical image database systems, which support the efficient transmission, storage, retrieval, presentation, and interpretation of medical images. A particularly interesting possibility in this context is to provide physicians and other medical personnel with the capability of content-based access to medical images for the purpose of assisting diagnostic decision making and providing support for medical research and training.

There is no doubt that robust content-based retrieval will prove valuable to the medical imaging field by providing considerable clinical decision support, increased effectiveness in diagnostic image interpretation and valuable support in medical training and research. However, one must first provide appropriate descriptions of image content and corresponding measures of similarity, such that medical images can be compared based on their morphological appearance, their clinical content, or a combination of the two. This is a difficult problem in general, but satisfactory solutions can be found based on image feature extraction techniques optimized for different image classes and tools for facilitating the interaction of potential users with the system for the purpose of selecting the set of features and similarity criteria to be used in specific content-based retrievals.

This paper presents I^{2}C (Image Indexing by Content), a prototype image database system developed in our laboratory, which supports the indexing, storage, and retrieval of medical images by content and serves as a platform for the implementation and evaluation of image content description methods, image analysis algorithms, and content-based retrieval strategies. The emphasis of this paper is on the architectural features of I^{2}C, its database engine, the class-based organization of the information stored in its database, and the role of such a system as an added value PACS service (see figure 1)). Therefore, specific image analysis and image content description algorithms are not considered in any detail in this paper. However, such algorithms are currently being developed and are also class-based in order to increase the accuracy of content descriptions and retrieval responses, and to improve the efficiency of content-based retrievals, since queries are directed to relevant classes, thus reducing the search space.

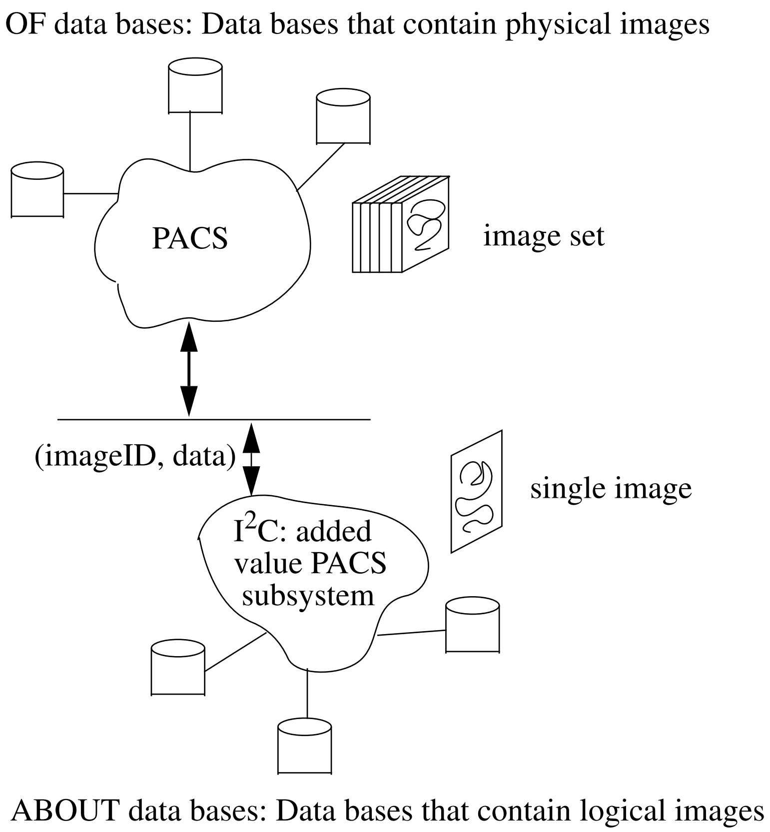

Figure 1: The role of I^{2}C as an added value PACS subsystem.

2 Background

In conventional medical image databases, image retrieval is accomplished through individual fields in the medical record of a patient, or using information contained in the image header. Image retrieval based on pictorial content is an emerging feature of modern image database systems and is becoming increasingly important in many application domains. This additional capability is particularly important in diagnostic medical imaging, both as a diagnostic decision support tool and as a valuable tool for medical research and training. In particular, content-based retrieval would not only yield cases of patients with similar diagnostic imaging examinations and the same confirmed diagnosis, but also --and perhaps more interesting-- cases of patients with similar diagnostic imaging examinations and different confirmed diagnoses. Content-based retrievals based on textual similarity of diagnostic reports would not necessarily yield the same information. The most elaborate methods of image content description in existence today incorporate mostly information from textual descriptions of image data and, to a lesser extent, information extracted from the raw image data. As a result, the development of methods for the description of image content is an active area of research [3][4][5][6]. It should also be pointed out that descriptions of image content used in image database systems may be entirely different, and possibly much simpler, than image descriptions used in image understanding.

The term *logical image* has been established as a standard term for the symbolic information that is extracted from the raw image data and auxiliary textual information. Consequently, the term *logical* database refers to a database that manages logical images. Various approaches have been exploited in deriving logical images and organizing them within the logical database [7] [8][9][10]. Chang [11] has proposed the use of 2D-strings for the indexing of images based on spatial relationships and attributes of the objects in the image. His methods have been extended by other researchers [12][13]. In all of these methods, first the image is segmented to derive the contours of objects in the image. The user may intervene to select and modify the contours of dominant objects. Appropriate attributes of the objects, such as shape, orientation, edge type, area, location, texture, etc. can then be computed. These attributes, together with the spatial relations of objects, are stored and indexed in the logical database.We use the term *Intermediate Image Description (IID)*, to denote the structure that contains the selected objects of an image, together with their attributes weighted according to their significance. *Description type* objects encapsulate information relevant to a content description algorithm and a corresponding retrieval strategy based on content similarity (matching algorithm).

The design of I^2C is not dependent on specific algorithms for image content description. Since the system is intended as a platform for the implementation and performance evaluation of image content description methods, no specific approach to content description is taken. The system provides the framework for the implementation of different content description methods in *description type* objects. We consider I^2C as an ever expanding toolbox for the development and comparative study of a wide range of algorithms for image content description and retrieval based on morphological and clinical similarity. Nevertheless, image classes are associated with default methods which are activated when users do not specify their own preference.

3 Image Classes

All actions and processes supported and used by I^2C are based on the organization of images and image related information into classes representing different imaging modalities and anatomical features. Thus, algorithms used to obtain descriptions of image content can be tuned for the images in a specific class. Algorithms tuned for a specific class are likely to yield far better results for images in this class than general-purpose algorithms, since they can exploit knowledge regarding the characteristics of images in the class. Furthermore, the efficiency of retrievals can be increased by directing content-based queries to specific logical databases.

For each image class, the system maintains a class object that encapsulates attributes relevant to all images in the class. Class attributes are segmentation algorithms, description types, and general information on the class members (images). Upon assigning an image to a class, its logical image is generated and stored by each of the description types supported by the class. As already mentioned, a description type object encapsulates information relevant to an image content description algorithm and a matching algorithm. The logical images generated by each content description algorithm are stored and indexed into a logical database associated with images in that class and the corresponding description type. The association of segmentation algorithms and description types with image classes is dynamic. The user may bind (or unbind) a segmentation algorithm or description type to a class. Default values of algorithm parameters can also be changed in order to tune the performance of a segmentation algorithm or to modify the similarity criterion and retrieval accuracy of a description type (see section 6).

The image class hierarchy is initialized at system installation via a configuration file. However, during routine system use, the user may arbitrarily modify the class hierarchy using the *class editor* of the I^2C user interface (see section 4.2). The user is allowed to insert, delete, and modify empty classes. Images can also be moved from one class to another, while new images can be inserted into a class and images belonging to a class can be deleted. However, in such interactions with the system, caution should be exercised, since some of these operations can be extremely time-intensive. For example, when many images are moved from one class to another, the logical databases of all description types supported by the source and target class are affected.

Images are assigned to classes using *primary* criteria, such as imaging modality, part of the anatomy, orientation, plain of cut, etc. and *secondary* ones. Secondary criteria of class membership may be derived from the clinical interpretation of image content or determined by a machine learning algorithm based on quantitative image features. An effort is currently in progress to use machine learning to automatically or semiautomatically assign images to classes. Automatic classification will result in an image being assigned to an existing class or the "unknown" class, while semiautomatic classification would give users an opportunity to approve or disapprove of the class selected by a machine learning algorithm for each image. The usefulness of machine learning in the I^{2}C environment is twofold. First, a large number of images can be automatically inserted in the system in batch mode. Second, during the processing of content-based queries, instead of the user manually selecting the target classes to which the search will be directed, appropriate target classes will be indicated by a machine learning algorithm.

4 System Architecture

4.1 Overview

The architecture of I^{2}C [14] is object-oriented and adheres to the principles o*f generality*, *extensibility*, and *modularity*. By generality, we mean that the system can be easily modified to be used in other application domains, in which the storage and retrieval of images by content serves a purpose. Although description types developed for medical applications do not necessarily apply as they are to other areas, most of the tools provided by I^{2}C can also be used in other application domains.

I^{2}C is an evolving system and part of this evolution is the incorporation of new methods and the extension of existing ones to better suit not only the medical expert who uses I^{2}C for content-based queries, but also the developer of new description types. To achieve this goal, the system architecture has to be extensible. New image processing algorithms and graphical tools should be easily incorporated and maintained within the I^{2}C framework. Given the extensibility of I^{2}C, it has been possible to integrate many of the graphical tools and image analysis algorithms which have been previously developed in our laboratory. In addition, new tools have been developed and integrated into I^{2}C.

The system consists of a small set of communicating modules shown in figure 2. These are the *I^{2}C core*, the *image processing module*, the *segmentation module*, the *image description module,* the *database engine*, and the *user interface*. All modules exchange information through the *I^{2}C core*. It is also through the *I^{2}C core* that the system communicates with the PACS world.

The integration of new graphical tools and image processing algorithms currently involves minimal changes to the I^{2}C core and recompilation. On the other hand, state-of-the-art automatic image segmentation algorithms and image content description methods can be introduced to I^{2}C at the application level, part of a process that requires no code recompilation. I^{2}C has been implemented on Sun workstations using XWindows Version 11 [15]. The user interface of I^{2}C has been developed in XView [16].

4.2 The main modules of I^{2}C

Each of the I^{2}C modules depicted in figure 2 is responsible for a specific type of operations

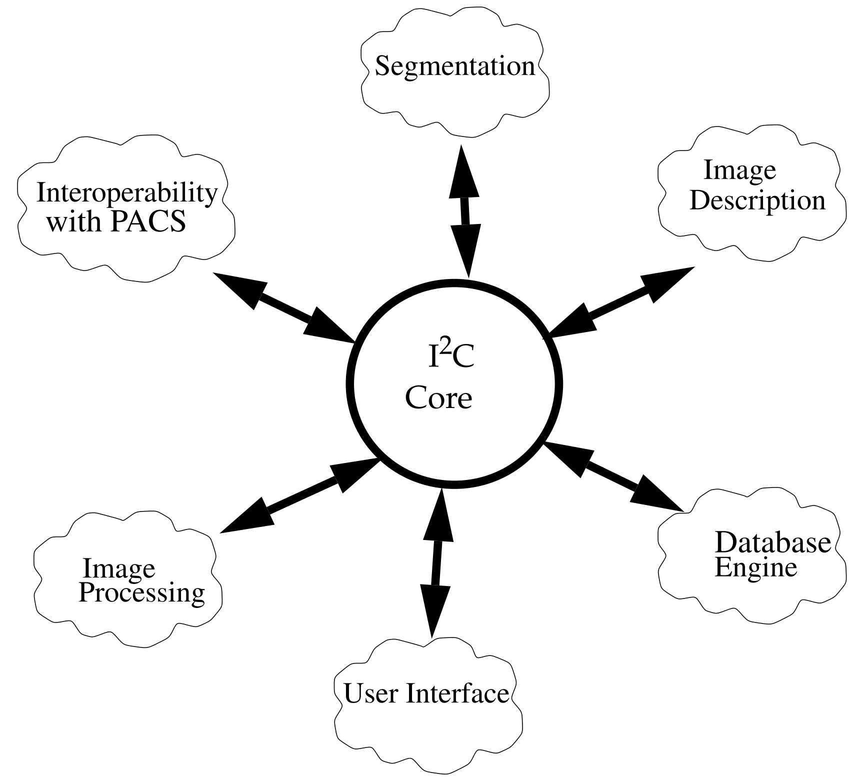

Figure 2: The role of I^{2}C as an added value PACS subsystem. The architecture of I^{2}C is modular. Different modules communicate by exchanging messages through the I^{2}C core.

supported by the system. All modules exchange information through the I^{2}C core, which handles messages and links I^{2}C to the PACS image archives. In the following paragraphs, we discuss each of the system modules:

Image Processing Module.The I^{2}C image processing module hosts various image processing and analysis algorithms, tools for the interactive segmentation of images and the computation of image features used in descriptions of image content.

Segmentation module. The I^{2}C segmentation module hosts segmentation algorithms. A segmentation algorithm should be *registered* before it can be used within the I^{2}C framework. A specification file, which contains the name of the algorithm, its parameters, and information on how to execute the algorithm is required for that purpose. The registration program reads the specification file and creates an appropriate persistent I^{2}C object. At that point, the segmentation algorithm is not *bound* to any class. The user may use the segmentation algorithm on any image, but the algorithm is not tuned to the characteristics of any particular class. Thus, it is yet another general purpose segmentation algorithm, which can be executed by I^{2}C. The user must explicitly *bind* an algorithm to an image class before it can be tuned for this class. Upon binding, a segmentation algorithm object for the specific class is created. This object keeps the default parameter values for that class and a *log* of recent execution results. Thus, if the segmentation of an image is requested two times in a row with the same parameters, the second time, system response is much faster.

Image Description Module. The image description module hosts image content description types. The components of a description type are the description manager, description generator, and description matcher (see section 6). The description manager inserts, deletes, and modifies logical images in the logical database of the description type. The description generator creates logical images from the raw image data and the intermediate image description. The description matcher processes content-based retrieval queries. Description types are registered with the system in a way similar to that of segmentation algorithms. The user supplies a specification file which contains the name of the algorithms, information on their parameters and direction on how to execute them, as well as the specification of the logical database schema. The latter is optional since it is possible that the description type does not use any of the I^{2}C persistence primitives, but creates and maintains its own logical databases. When a description type is registered with the system, the user may preview the content-based description of an image in the specified description type, but the logical image cannot be saved. Description types are bound to classes in the same way segmentation algorithms are. Upon binding, a description type to a class, the logical images of all images in the class need to be generated and stored by that description type.

Database Engine. The I^{2}C database engine manages the persistent I^{2}C objects, which keep information on classes, algorithms, and description types. It also provides primitives that allow the creation and maintenance of logical databases by description types. In the current version of the system, the I^{2}C database engine has been implemented on top of the EXODUS storage manager [17]. We arrived at this decision after extensive experimentation with POSTGRES [18] and careful consideration of other well-known object-oriented DBMS like O2 [19]. EXODUS is a storage manager which provides low level primitives for atomicity, indexing, and object management. It is quite fast when compared to other known DBMS, largely due to its low level interface. An important feature of the I^{2}C database design is that it yields a certain degree of portability and independence from the current implementation platform. Another advantage of the I^{2}C database engine is that the various logical databases may have dedicated servers, a fact which calls for the exploitation of the inherent parallelism in queries that involve multiple classes. The database engine is further discussed in section 5.

User Interface. The modular design of I^{2}C allows the easy integration of graphical tools [20] within the system, in order to facilitate the efficient and reliable extraction of image features. The basic philosophy of the I^{2}C user interface is that all activity should be directed from a single console-window, whose appearance changes as the user activities change. This approach is implemented with multiple overlapping panels, which disappear when they are not used, and reappear when needed. In this way, a smooth interaction of the system with the user is accomplished. We use the term *UI frames* to denote the multiple levels of the user interface in which various graphical tools reside. Currently, the I^{2}C user interface has two UI frames, the *basic* and the *secondary*. However, we expect that additional frames will be needed in the future, as new graphical tools are developed and integrated into the system. Each graphical tool occupies part of a UI frame. When it is integrated in the system, its call-dismiss scenarios should be specified, in addition to its exchange of information with the other I^{2}C graphical tools. This approach upholds the extensibility and modularity principles of the I^{2}C design. Figures 3 and 4show parts of the basic and secondary frames of the I^{2}C user interface:

- **Display Window**: The display window occupies the lower right portion of the basic UI frame. Up to four image sequences, of potentially different modalities can be concurrently displayed in the display window. At any one time, one of these sequences is *current* and the image of the current sequence on display is the *active* image. Image processing and enhancement

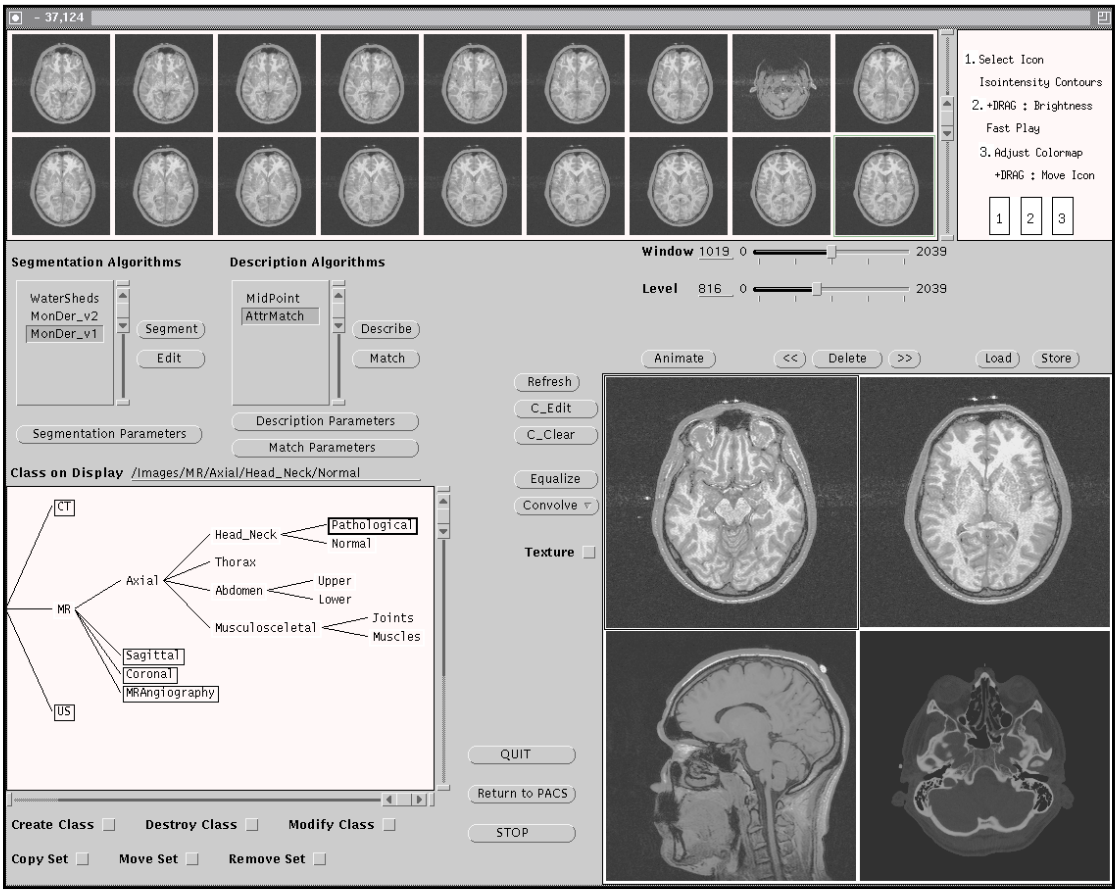

Figure 3: The basic frame of the I^{2}C user interface

algorithms, as well as interactive segmentation, may be applied to the active image. I^{2}C is able to display 8 and 16 bit images. The range of displayed pixel intensities and contrast of 16 bit images may be adjusted by level and window sliders.

- **Class Browser/Editor**: The class browser occupies the lower left part of the basic UI frame. The user may browse through different parts of the class hierarchy and observe the attributes of individual classes. Miniature versions of all images in a class may also be previewed by loading them to the image browser, which is located at the top of the basic UI frame. Using the class editor buttons, below the class browser window the user is able to modify the class hierarchy. Specifically it is possible to:
    - ♦ Create, delete, merge, or modify image classes.
    - ♦ Bind or unbind segmentation and description algorithms to image classes.
    - ♦ Move images from one class to another.
    - ♦ Insert (delete) images to (from) classes.
- **Image and Contour Browser**: The image browser occupies the top left portion of the basic UI frame. It may display miniature versions of the images or intermediate image representations. The user may select a set of miniatures for further processing in the display window.
- **Help Window**: The help window occupies the top right portion of the basic

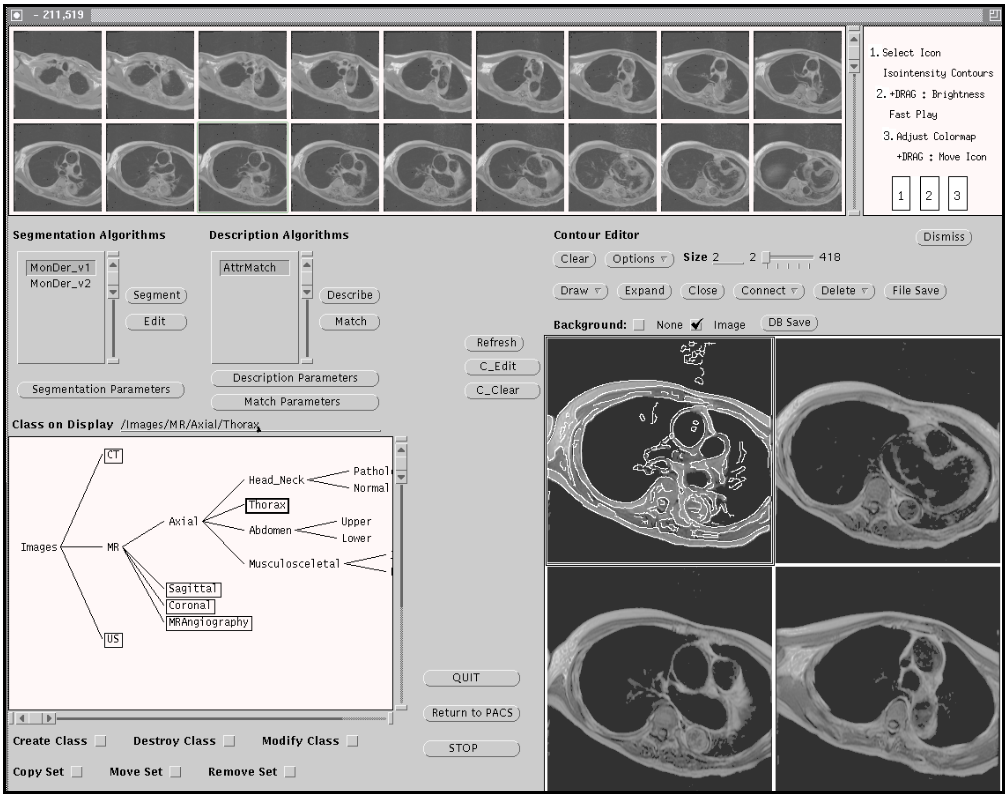

Figure 4: The secondary frame of the I^{2}C user interface

UI frame. It displays the functions of the mouse in various operational modes, for different areas of the UI frame.

- **Contour Editor**: The contour editor occupies the lower right portion of the secondary UI frame. Its main purpose is the creation of an intermediate image description. The user may import the result of the segmentation to the contour editor for further processing. In the case of automatic segmentation, image contours are represented as polygonal approximations and imported into the contour editor. In the case of interactive segmentation, the user may draw arbitrary shapes to outline interesting regions. The user is allowed to preview various attributes of the regions contained within existing contours, compute texture descriptors, and associate significance factors with specific contours. Another use of the contour editor is to draw sketches that will form part of a *sketch* query (see section 4.3).
- **Note Editor**: The note editor occupies the lower left portion of the secondary UI frame. Using the note editor the user may annotate specific images in I^{2}C. It is also possible to annotate individual contours in the intermediate image description. These annotations can later be modified or deleted.
- **Algorithm Workbench**: The algorithm workbench occupies the middle left portion of the basic UI frame. It is used to change the parameter settings of segmentation algorithms and description types, to activate such algorithms, and also to enter content-based queries targeted at the class of the active image. The user may select a segmentation algorithm and apply it to an active image using the default parameter settings for the current class or parameter settings of the user's choice. If the segmentation results obtained

with the new parameters are better than those obtained with the default settings, the current parameter values may become the default for the current image class. It is also possible to export the segmentation results to the contour editor, by simply pressing the *Edit* button. In the case of a description algorithm, the user may create, preview, modify, or delete the resulting logical image. The user may change the default parameters of the description generator for the current class, or just preview the logical image produced with given parameters.

4.3 The flow of Information in I^2C

Although, I^2C has been designed and developed as an added value PACS subsystem, it retains a certain degree of independence. It communicates with other PACS subsystems via *imageIDs.* An *imageID* is a byte string which identifies individual medical images, the imaging station that produced them, and the archive they can be retrieved from. Images may be imported from the RIS or PACS archive, to be used in content-based retrievals or to be inserted in the I^2C database. The use of imageIDs in this process is transparent, since users do not handle imageIDs directly. It is at the network level that each image is identified by its imageID. I^2C requests images from PACS archives based on their imageIDs. Images may be transferred over the local network in any of the standard formats.

Thus, I^2C does not manage large image archives. Assuming that only images recalled in the recent past are likely to be requested in the near future, only recently recalled images are stored in the local archive. All reference to images in the I^2C logical databases involves imageIDs. When an image is referenced or retrieved, I^2C looks up its imageID and if the image is not stored locally, the system requests the image from the appropriate PACS archive transparently. Figure 5 gives an overview of the I^2C architecture and the flow of information through its modules. The network shown on the figure refers to the communication medium that links I^2C to the local PACS. In the following paragraphs, we describe scenarios for inserting an image in the system, and initiating a query for content-based retrieval.

Image Insertion. In a typical image insertion session, I^2C is accessed from the RIS, or PACS after a set of images has been selected. The imported images appear on the image browser. The user selects the image to be inserted in I^2C and the image class in which the image will be inserted. The image is loaded in the display window and each description type supported by the target class generates and stores its logical image. Logical image generation may proceed automatically or interactively. In the automatic mode, the image is inserted in the description generation module, the intermediate image description is automatically generated, and each of the description types associated with the image class creates and stores an appropriate logical image to its logical database, making suitable modifications to the indices involved.

Alternatively, the user may *interact* with the system for the purpose of generating the intermediate image description. After some preprocessing, the image is segmented automatically or interactively using one of the graphical tools for interactive segmentation. The user may preview the result of the segmentation, and repeat the whole process with different parameter settings, if necessary. Next, the segmented image is imported to the contour editor, so that the user can make the necessary modifications on object contours. Individual contours may be assigned significance factors, and attributes of segmented regions, such as area, location, texture, etc., are computed. All this information is stored in the *intermediate image description* object. The intermediate image description and the image data

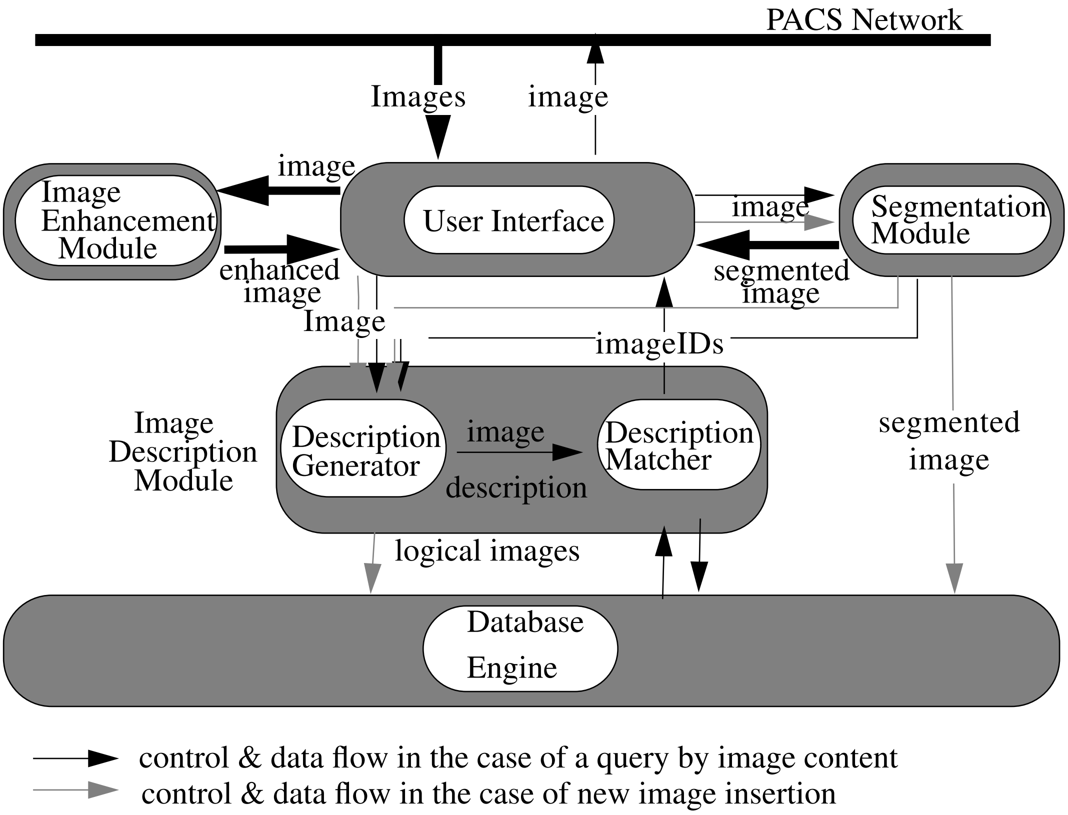

Figure 5: Architecture and main data paths of the I^{2}C system.

form the input to description types. This information may or may not be used by individual description types.

Upon inserting in the I^{2}C database the user may add annotations to the image. Annotations may be also added to the regions enclosed by contours in the intermediate image description. Intermediate image descriptions are stored in the I^{2}C database and the user may preview and modify them, independently of description types. The flow of data in case of image insertion appears with dotted lines in figure 5.

Content-Based Queries. I^{2}C supports two basic forms of content-based queries: *image* queries and *sketch* queries. As in the case of image insertion, the user accesses I^{2}C from the RIS or PACS system and imports to it a set of images, which will form the basis for a content-based retrieval query. In the case of a simple content-based image query, the query image is loaded on the display window, the target class and description type are selected, and the query is entered. The response to the query appears on the image browser, in the form of miniature images. An image query involves the interactive or automatic generation of an intermediate image description for the query image and the setting of relevant retrieval parameters.

In the case of sketch queries, the user enters the contour editor environment and sketches relevant anatomical objects, which will form the basis for the retrieval of similar images by content. Then, the target class and description type are selected and the query is entered, as before. The flow of data in content-based queries appears with solid lines in figure 5.

5 Database Engine

In recent years and in many database applications, the relational paradigm is gradually being replaced by the object-oriented paradigm. The advantages and limitations of both paradigms were studied, both from a theoretical point of view and based on extensive experimentation with selected commercial and public domain databases. Although the object-oriented paradigm has better abstraction capabilities, most of its current implementations are still slow and complex. The I^{2}C database engine has been designed with a view toward achieving the speed of the relational model, while taking advantage of the abstraction and modularity of object-oriented models. Currently the I^{2}C database engine has been implemented on top of EXODUS, a storage manager known for its simplicity and speed. The engine uses a small set of EXODUS primitives for atomicity, indexing, and object management to build and manage its persistent objects. This makes it possible for the I^{2}C database engine to be ported with minor changes to any storage manager or DBMS that provides similar primitives.

The I^{2}C database engine provides two types of persistent objects: I^{2}C object types and persistence primitives which may be used in the creation and organization of logical databases. Figure 6 shows the I^{2}C object types of the database engine:

- The *classEntry* object contains information related to an image class. This information consists of pointers to imageEntry objects which keep general information regarding images in the class, user annotations relevant to the class, and pointers to the segmentation algorithms and description types supported by the class.
- The *imageEntry* object includes image annotations and information on how the raw image data can be accessed. If the image is stored locally, this information is a filename in the local disk; otherwise, it is the address of the PACS archive, where the image can be found using its imageID.
- The *intermediate image description* object contains the IID of an image, consisting of contours of important image regions, a significance factor for each, and various geometrical features.
- The *segAlgoEntry* object contains information related to a segmentation algorithm, which has been tuned for a particular image class. It includes the default parameter setting of its parameters, information on how to execute the algorithm, and a log of recent execution results.
- The *i2cAlgoEntry* object contains information about a description type associated with a specific class. This information includes default parameter settings for the description generator and description matcher, information on how to execute the relevant algorithms in order to describe an image or initiate a content-based retrieval, and a pointer to the corresponding logical database.

I^{2}C provides the application programmer with two primitives which can be used to create and organize persistent objects: *clusters* and *indices*. The I^{2}C programmer may use clusters to store selected persistent objects, while indices may be used to achieve fast access to these objects (see figure 7). As shown on figure 8 objects which consist of various attributes, are stored in clusters as byte strings starting with an *objectID.* The objectID uniquely identifies the object within the logical database. Individual object types must provide methods to pack an object into a byte string and to read out its contents by unpacking a byte string. Clusters provide sequential access to their objects, as well as random access

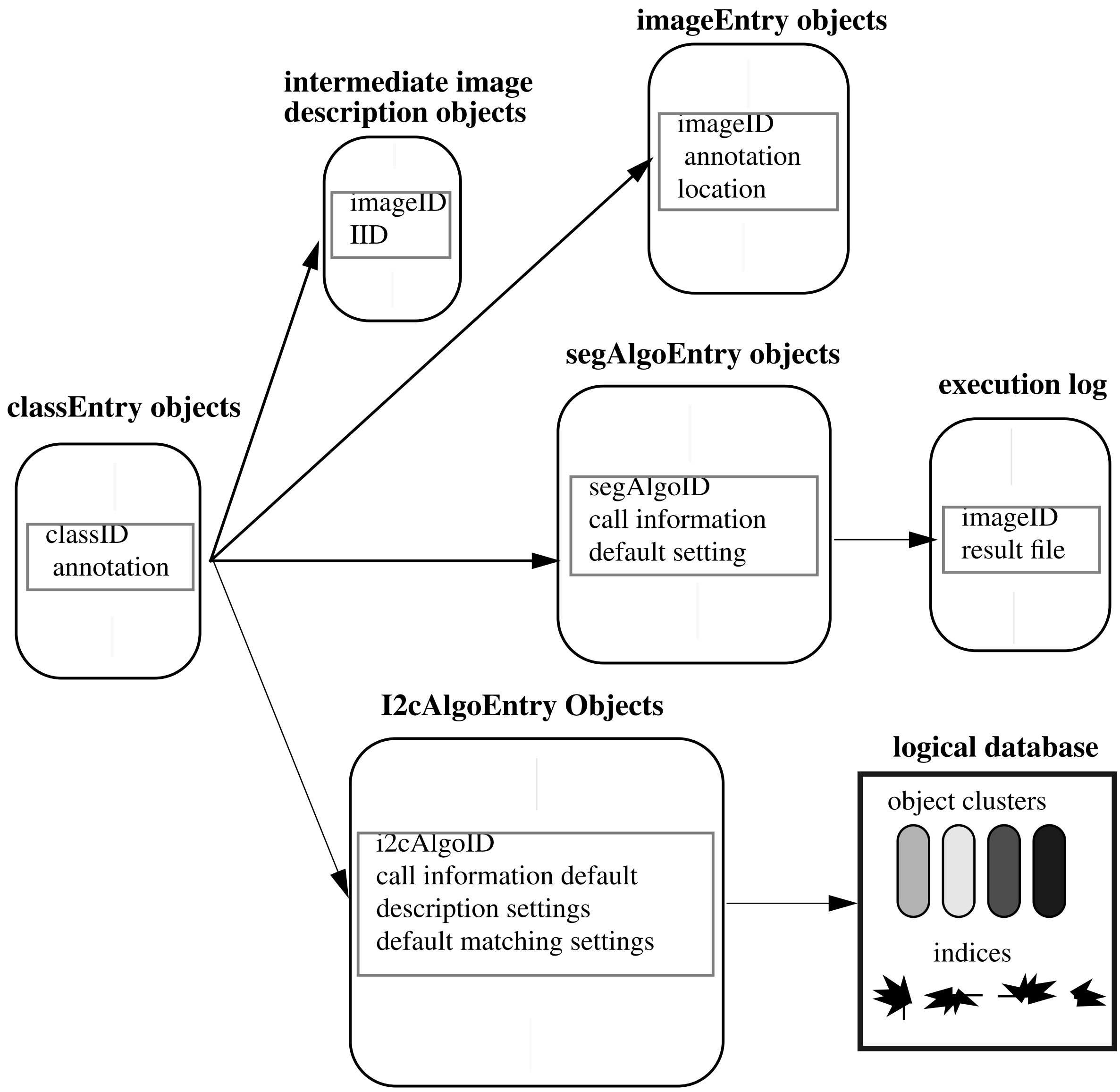

Figure 6: The I^{2}C persistent object types.

based on ObjectIDs.

Indices provide fast access to objects stored in the logical database. The objectIDs of the objects are indexed via user provided keys. The type of the keys may be integer, float, double, or string. Multiple level indices may be implemented by combining indices and clusters. This is also illustrated symbolically in figure 7. Although the system currently supports *hash* and *B-tree* indices, the programmer may build structures of greater complexity combining these two types of indices. Nevertheless, other spatial indexing methods, e.g. R-trees, will also be incorporated in the future to assist the programmer in the task of creating sophisticated logical databases.

It should be mentioned, however, that the I^{2}C application programmer is not limited to using the primitives provided by the I^{2}C database engine. Description types that may use their own specific means of persistence may also be integrated with the system.

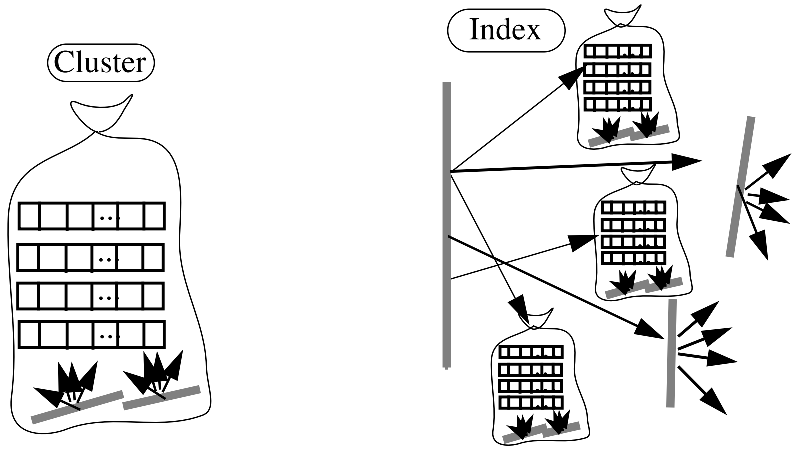

Figure 7: The I^{2}C application programmer may use clusters to store selected persistent objects of his choice and indices to provide fast access to those objects.

6 Description Types

Description types are used in I^{2}C to encapsulate information relevant to an image content description method and a content-based retrieval strategy. The structural components of a description type are: the *description generator*, the *description manager*, and the *description matcher*. The description generator produces the logical image, which consists of a set of persistent objects. The description manager manages logical images in the logical database of the description type. It is responsible for the insertion, deletion, and modification of such logical images, with concurrent update of the indices involved. Finally, the description matcher processes content-based queries addressed to the description type and identifies images similar to the query image. The imageIDs of the images that constitute the response to the query are reported back to the system, the raw image data are retrieved, and their miniatures are displayed on the image browser. The flow of data between the different components of a description type is shown in figure 9.

The object-oriented approach which has been adopted, imposes minimal constraints on

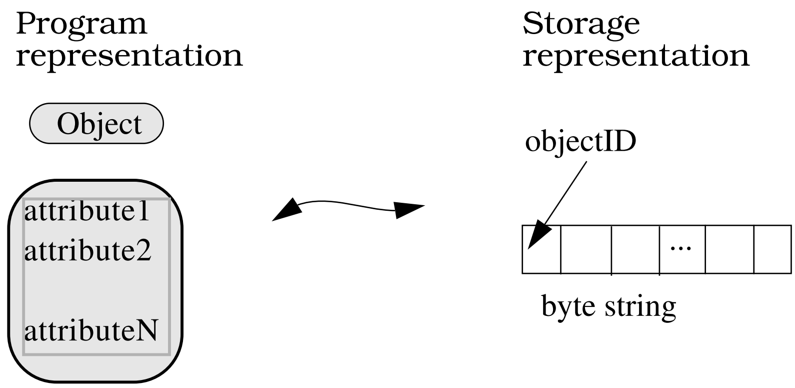

Figure 8: The storage representation of an object within a cluster is comprised of a byte string and an objectID. Individual object types provide their own methods

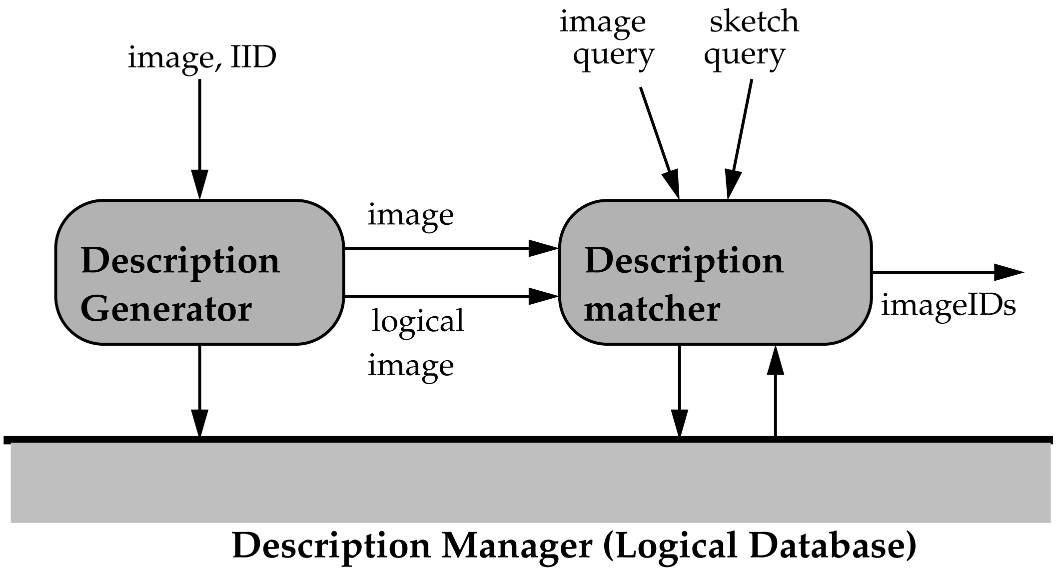

Figure 9: The components of a description type and relevant flow of data.

the description methods that can be supported by the system. Each description type is viewed as an object capable of managing logical images and responding to content-based queries. The input to the description generator and the description matcher is the image and its intermediate image description. The input to the description manager is an imageID and commands such as preview, insert, delete, and update. It should be noted, however, that the description manager of a given description type may only support a subset of these commands, i.e. some description types may not allow logical image deletion. In some cases, the similarity criterion of a description type can be modified by changing the input parameters of the description matcher. The user may influence the precision of a content-based retrieval by modifying these parameters, thus trading precision for speed, and vise versa. In the I^{2}C system, knowledge about the schema of the logical database and the search and retrieval strategies associated with a specific description type, is encapsulated in the particular description type. What the system provides is a set of primitive operations that enable indexing by content methods to create persistent clusters of objects, maintain indices on them, insert objects into a cluster, search and retrieve such objects, etc.

To demonstrate the validity of the approach and to provide I^{2}C with a default description type, we have implemented *AttributeMatch*, a description type which uses a flexible similarity criterion based on a combination of weighted image features. Weighting factors are used to order image features in terms of significance in determining image content similarity. Adjusting these factor, the users may combine features in such a way as to obtain a purely geometrical similarity criterion or a similarity criterion which conforms with clinical similarity as this may be perceived by a physician.

The *AttributeMatch* description generator uses the intermediate image description to create a hierarchical structure comprised of contours, segments, and points. At each level, a set of attributes are defined. Contour attributes relate to statistical and geometric properties of the objects depicted in the image. Some attributes are computed and stored by the description generator, while the rest are computed on demand by the description matcher. In addition to stored attributes, each contour, segment, or point is associated with a significance factor. This factor weighs the contribution of each contour, segment, or point to image content descriptions at higher levels of abstraction. Significance factors can be either set by the user, who interactively participates in the generation of the intermediate image description, or determined automatically using predefined criteria.

Currently, the performance of this description type is being evaluated and its parameter settings are being fine-tuned for different image classes. However, a detailed presentation of *AttributeMatch* and alternative description types is beyond the scope of this paper and thus has been omitted. Figure 10 displays the I^{2}C user interface upon getting the result of

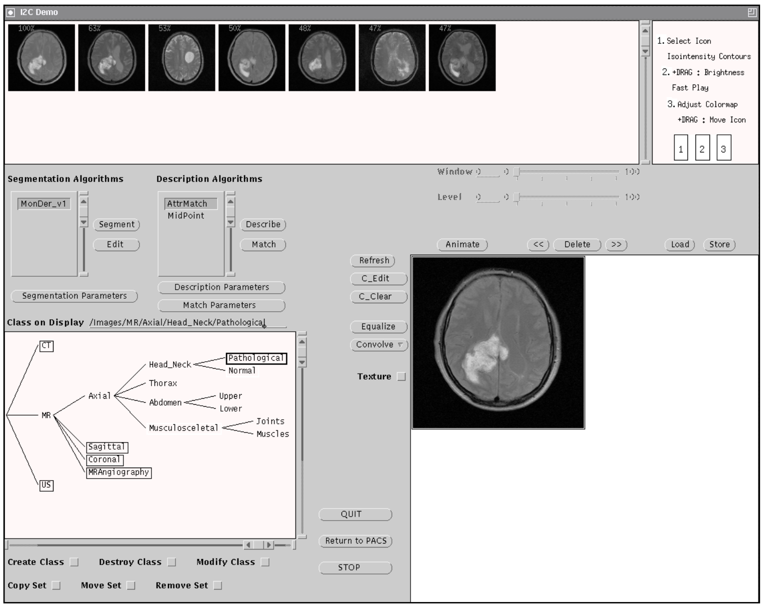

Figure 10: The appearance of the I^{2}C user interface upon reporting the result of a query using *AttributeMatch*.

a retrieval by content query using *AttributeMatch.*

7 Discussion

Version 1.0 of I^{2}C has been up and running in the Computer Vision and Robotics Laboratory of our institute for about 3 months. Various image processing and segmentation algorithms have been successfully integrated in I^{2}C. Alternative description types based on various image content description algorithms and similarity criteria have also been implemented at the I^{2}C programming level.

The performance of I^{2}C is currently being evaluated in terms of the accuracy of content-based retrievals achieved by different description types, the response time, and the ease of use. Generally speaking, a retrieval strategy is evaluated on the basis of efficiency, recall accuracy, and precision [21]. Efficiency can be easily quantified based on the response time to different content-based queries. Recall accuracy reflects the percentage of relevant images which are actually retrieved, while precision reflects the percentage of irrelevant images which are retrieved. There is an inherent difficulty in developing strict methodologies toward obtaining quantitative measures of recall accuracy and precision. Furthermore, in many content-based retrieval methods, efficiency is not independent of recall accuracy and precision. Specifically, there may be a trade-off between efficiency and recall accuracy. In the future, quantitative and qualitative measures of system performance will be developed and used to evaluate in a systematic way the performance of I^{2}C.

In this paper, the architectural and functional features of I^2C, a system for the indexing, storage, and retrieval of medical images by content, have been considered. I^2C is expected to provide added value to future generation integrated PACS, through its role as a diagnostic decision support tool, as well as a tool for medical research and training. Furthermore, I^2C has been designed and currently functions as a platform for the development and evaluation of image content description and matching algorithms in the form of description types. Future extensions to I^2C include the development of tools for the evaluation of algorithms and methodologies for the description and retrieval of images based on their content.

The clinical evaluation of I^2C in a PACS environment will soon provide the necessary feedback, so that system can be further developed and a proper evaluation of its performance and true potential be carried out.

Acknowledgments

This work has been funded in part under project *EurIPACS* of the CEC AIM Programme. The authors wish to thank Manolis Lourakis, Despina Vamvaka, Kostas Georgiadis, Afroditi Galata, Maria Glarou, Dora Magoulioti,and Fenia Zioga, the entire *EurIPACS* team at ICS-FORTH and Yiannis Kavaklis from KNOSSOS Technologies, S.A. an associate partner of ICS-FORTH in *EurIPACS*, for valuable contributions to the work presented in this paper.

References

[1] S. C. Orphanoudakis. Supercomputing in Medical Imaging. *IEEE Engineering in Medicine and Biology*, 7 (1988), pp. 16-20.

[2] G. D. Rennels and E. H. Shortliffe. Advanced Computing for Medicine. *Scientific American*, 257 (1987), pp. 154-161.

[3] S.-K. Chang and S. Y. Lee. Retrieval Of Similar Pictures On Pictorial Databases. *Pattern Recognition*, 24 (1991), pp. 675-680.

[4] S.-K. Chang and S.-H. Liu. Picture Indexing and Abstraction Techniques for Pictorial Databases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6 (1984), pp. 475-484.

[5] S.-K. Chang. Image Information Systems: Where Do We Go From Here? *IEEE Transactions on Knowledge and Data Engineering*, 4 (1993), pp. 431-442.

[6] P. Kofakis and S. C. Orphanoudakis. In *A Second Generation PACS Concept*, M. Osteaux et.al. (Eds.) Springer-Verlag, 1992, ch. 7.3: Image Indexing by Content, pp. 250-291.

[7] L. Gurta and M. D. Srinath. Contour Sequence Moments For The Classification of Closed Planar Shapes. *Pattern Recognition*, 20 (1987), pp. 267- 272.

[8] T. Joseph and A. F. Gardenas. PIQUERY A High Level Query Language for Pictorial Database Management. *IEEE Transactions on Software Engineering*, 14 (1988), pp. 630-638.

[9] I. Kapouleas. Segmentation and feature extraction for magnetic resonance brain image analysis. In Proceedings of the *10th International Conference on Pattern Recognition*, Atlantic City, New Jersey, June 1990, pp. 583-590.

[10] S.-Y. Lee, M.-K. Shan, and W.-P. Yang. Similarity Retrieval of Iconic Image Data-

bases. *Pattern Recognition*, 22 (1989), pp. 675-682.

[11] S.-K. Chang, Q.-Y. Shi, and C.-W. Yan. Iconic Indexing by 2D Strings, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 9 (1987), pp. 413-428.

[12] E. Petrakis and S. Orphanoudakis. Methodology for the representation, indexing and retrieval of images by content. *Image and Vision Computing*, 11 (1993), pp. 504-521.

[13] S.-Y. Lee and F.-J. Hsu. Spatial Reasoning and Similarity Retrieval of Images using 2D C-Sstring Knowledge Representation. *Pattern Recognition,* 25 (1992), pp. 305-318.

[14] S. Kostomanolakis, M. Lourakis, C. Chronaki, Y. Kavaklis, and S. Orphanoudakis. The Architecture of a System for the Indexing of Images by Content. In Proceedings of the *International Symposium CAR'93,* Springer-Verlag, 1993, pp. 278-282.

[15] V. Quercia and T. O'Reilly. *X Window System User's Guide*. O'Reilly and Associates, Inc., 1990.

[16] D. Heller. *XView Programming Manual*. O'Reilly and Associates, Inc., 1990.

[17] Using the EXODUS Storage Manager V3.1. *Technical Report*, Computer Science Department, University of Wisconsin Madison, Nov. 1993.

[18] M. Stonebraker and L. Rowe. The Design of POSTGRES. In Proceedings of the *1986 ACM-SIGMOD Conference*, Washington DC., June 1986.

[19] O. Deux et al. The Story of O2. *IEEE Transactions on Knowledge and Data Engineering*. (1990).

[20] P. Kofakis and S. C. Orphanoudakis. Graphical Tools and Retrieval Strategies for Medical Image Databases. In Proceedings of the *International Symposium CAR'91*, Springer-Verlag, 1991, pp. 519-524.

[21] E. Petrakis. Image Representation, Indexing and Retrieval Based on Spatial Relationships and Properties of Objects. *PhD thesis*, Department of Computer Science, University of Crete, 1992.