Support to Content-Based Image Query in Object-Oriented Databases[1]

Caetano Traina Jr.[§,*]; Agma J. M. Traina [§,*]; Rildo R. dos Santos[&,*]; Edna Y. Senzako [§];
[§] Computer Science Department - Mathematics Institute at Sao Carlos of Sao Paulo University - USP - Brazil
Cx. P. 668 - Sao Carlos, SP
13560-970 - Brazil
Tel: +55 162 274 9136, FAX: +55 162 274 9150

[&] Computer Science Dept. - State of Sao Paulo University at S.J. Rio Preto - UNESP - Brazil
[&] Informatics and Physics Dept. - Physics Institute at Sao Carlos of Sao Paulo University - USP - Brazil
[*] Development and Research Institute on Database and Information Management - Sao Carlos
E-mail: [Caetano|Agma|Senzako]@icmsc.sc.usp.br , Rildo@dcce.ibilce.unesp.br

Keywords: Image Database, Digital Library, Medical Databases, Image Information Systems, Data Modelling, Object Oriented Database.

Abstract

This paper presents an Object Manager that stores medical images as a data type of attributes that can be associated to objects, and a support system developed to offer this kind of tool to medical application developers. The purpose of this work is to support the retrieval of images through queries based on the graphical contents of stored images. The usual approach uses icons and textual attributes stored with the images to specify queries. This work uses a novel modeling technique to define the "image data type" , by means of which it is possible to decide, before the query itself, the key data of each image that must be extracted from the image when it is stored in the database, allowing faster search when queries are issued. This approach enables building of expansible systems, where new image processing algorithms can be added easily, using its syntactic representation stored through an Image Meta-schema into the application database schema. This work shows how such a system has been implemented, and also a query language used to refer and execute these algorithms from inside the database management system.

1. Introduction

Medicine is using ever more equipment that generates large amounts of data in digital format, much of which must be stored for long periods of time [1]. This paper assumes that these images are being stored into databases, although the highly variable nature of this kind of data is usually inadequate for traditional Database Management Systems (DBMS) [2]. One type of equipment that has raised a lot of interest is the Magnetic Resonance Tomography (MRT) device [3], which has been used to obtain images from the internal structures of the human body. The high quality images obtained can be cross-compared with other images, looking for predefined biological structures, in the search for appropriate diagnoses. This and other modern medical equipment can generate large amounts of data that must be stored and indexed, to be quickly retrieved later.

A Database System, suitable to support applications that must handle this kind of data efficiently must be able to effectively store, manipulate, retrieve and distribute images and non-image data. This kind of underlying system is known as the Picture Archiving and Communication Systems (PACS). Applied in medical image storage, PACS provides graphical information of the many aspects of each patient's health, diseases and treatment for the physician and technicians of a health care center [1]. Therefore, the database must store all the images, the parameters used to gather those images, and the criteria and results of the diagnoses in the same place.

[1] This paper was partially supported by: CNPq, FAPESP and CAPES-PICD.

Existing systems usually store descriptive information associated to the images, such as the comments and the diagnoses already made by a physician, or shortened icons of each image. To find images through textual data, a detailed description is obtained with the help of a specialist, but the description depends on his/her knowledge of each specific topic, or on his/her purpose at the time of analysis. Therefore, textual data cannot substitute the actual information contained in an image. An iconic image is also not appropriate for the domain of a medical image database, because it is difficult to select an icon that could, in a significant way, represent the complex information presented in this image [4] [5]. So, meta-information cannot substitute the actual information contained in an image.

The search for stored images based on their full contents, although highly desirable, imposes a processing load on the search data engine that the current machines cannot accomplish within an acceptable period. This work describes an alternative way to deal with this excessive workload by pre-processing each image as it is stored into an Object-oriented Database Manager (Object Manager) [6]. The images are treated as a new data type, so index structures can be constructed over them. The index structures are built using a set of parameters of each image extracted by an Image Matching System [7]. The use of index structures can quickly reduce the set of target images to be retrieved when queries are issued.

This work is part of a joint project involving activities in the Informatics and Physics Department, of the Physics Institute at Sao Carlos (IFSC), the Computer Science Department at the Sao Carlos Institute of Mathematics (ICMSC), both of the University of Sao Paulo (USP), and the Santa Casa Hospital in Sao Carlos. This project consists of the development of a complete Magnetic Resonance (MR) Tomography System [3] [12]. This article deals with the image storage and visualization system.

2. Language elements and Correlated Concepts

Images are defined in the Object Manager as part of its data meta-model, as a specialization of the concept of Attributes associated to Objects. The meta-model of this system assumes every attribute has a **characteristic**, which defines the set of operations where it is involved. For example, a number is an attribute characteristic, so numbers are attributes that can be added, multiplied, and so on; strings are another attribute characteristic, and strings can be searched for substring matching, can be concatenated, and so on. It must be noted that the set of operations allowable over an attribute characteristic is independent of the exact data type used to represent that attribute, so it does not matter if a number is represented as an integer, a real or a double precision float. Characteristics are specializations of the concept of object attributes that are further specialized in the Data Type concept, so each data type can have a specific set of properties. For example, complex numbers can have a method to calculate their real part. Images and procedures (methods) are two other attribute characteristics of this system, enabling various image representation formats (image data types) and image processing algorithms to be associated to objects, like any other numerical or textual attributes.

To support the definition of images in this meta-model, the following concepts are needed [8]: **Image Constants**, **Image Summarizer**, and **Image Parameters**. *Image Constants* are predefined images, meaningful in the application domain. For instance, a physician could have obtained a tomography where a given organ pathology appears in a predefined manner, that could be used as a reference to search for or to identify other images with similar visual information. This image is, thus, defined as an Image Constant, so that it will be used by the system as a comparison pattern.

An *Image summarizer* is a procedure that operates over two images, extracting useful data from the image, called *Image Parameters*. Image Parameters can be numerical values, image coordinates, sets of values, etc. Each Image Summarizer can return one or more Image Parameters, each of them representing some useful information for the process of image selection and retrieval.

Another construction was included to yield the many variations in which a target object can

appear in an image. That construction is called a **modifier**, and its purpose is to allow alterations on the default action of each image summarizer. The acknowledged variations in the images do not affect the basic behaviour of the summarizer, yielding image variations like geometric transformations, contrast, etc. The application of any of these modifiers is directed towards two categories of constant image information that must be retrieved: i) *Partial Constant Image*, means the comparison of one target image with the image constant can be found in sub-images of the image, and ii) *Full Constant Image*, in which each image constant must be entirely compared with the target image.

Using these concepts, images can be stored in association with objects in the database, and are treated as a characteristic of object attributes of a given data type. However, to speed up the queries involving image content data, the operational environment must be configured for the specific application domain. This is achieved by defining the set of image constants and image summarizers that are meaningful for each application. The idea is: when an image is to be stored in the database, it is submitted to a predefined set of <image summarizer, image constants> pairs, and the resulting image parameters are used to create index structures, representing some form of "distance" of each stored image from this Image Constant.

When an application is built, the set of Image Constants related to each summarizer is defined, and the object attribute type summarizer needs to be evaluated. Thus, when each image is stored in the database as an attribute value, only the pairs <summarizer, constant> related to this particular association of the image are evaluated. Each of the resulting parameters is used to create the index structures, one for each image summarizer, applied over each image constant meaningful to this object attribute. Each index structure is a modified R-tree [9], representing the image in a N-ary space, where N is the number of dimensions of the particular parameter. In the retrieval environment, each parameter is attached to a user-defined range, so each query retrieves the images that each parameter has within the corresponding range. Each image can have indexes in many trees, one for each parameter of each summarizer meaningful for the object attribute type of which the image is a value.

In medical applications it is very important to work with adjustable matching methods, because images do not need to match an image constant exactly to be included in the result set of a specific query. For instance, images that match 85% or more of the image constant can be considered a good answer, and they must be included in the index structure. Methods have been implemented that permit retrieval of images that match a stored image exactly, as well as methods where we can discard images with less than 85% of similarity, which are not accepted as a match. The matching methods implemented are based on the Correlation Coefficient, Correspondent Points Summation, Sequential Similarity Detection Algorithm-SSDA [10] and Moment-Preserving Pattern Matching-MPPM [11].

3. The Image Content Schema

The image support system has a meta-schema that represents the information contained in an object attribute of the image characteristic. The application domain is defined using objects of the

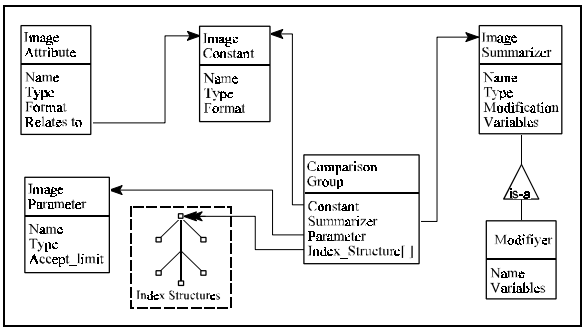

Figure 1: Meta-schema of an image-supporting application**.**

type Image Constant, each having the attributes: *name*; *type* (partial or full Image Constant); and others describing the image. The characteristics that can be extracted from each stored image through a comparison with one Image Constant are represented as an object of the summarizer type, whose attributes are: *name*; *type* - meaning the image processing algorithm (histogram evaluation, matching searches, texture identification, etc.); and *modification* - an indication that the summarizer can have another definition for its default action.

Each parameter returned by an Image Summarizer is represented as an object of the Image Parameter type. Its attributes are: *name*; *type* of information returned by the Summarizer; and an *Acceptance_Limit* defining the threshold from which an image will have one entry into the index structure. The Summarizer Modifiers are specializations (subtypes) of Summarizers, and thus are, in fact, alternative image processing algorithms used in particular situations, depending either on the query or on the object to which the image is attributed.

Figure 1 shows an overall diagram of the Meta-schema of this structure. The whole information extracted from one image when it is stored in the data base is represented by objects of type "Comparison Group", relating objects of the Image Constant, Summarizer and Image Parameter types. This object type provides a semantic meaning for each value stored in the index structures. The index structures are accessed from their instances (as represented in figure 1 by the small "tree" associated to this object type). The image itself is represented as an Attribute of the characteristic image, whose own set of attributes consists of: *Attribute_type_Name*, the name given to this type of attribute in the application schema; *Instance_Name*, an identifier of each image instance in the data base. Other attributes are specific for each of their instances, such as *length*, the *image* itself (the pixel matrix), *dimX*, *dimY*, and its *format*.

When an application is developed, its schema is added to the Meta-schema through the definition of some attributes of the data type "image". Each attribute of the data type image, assignable to object types must have at least one "image constant", thus creating the link between the application schema and the meta-schema. For example, if objects of the user-defined type Patient have an attribute "lung tomography" defined in the application schema, then each object of the system-controlled type "Image Constant" whose values are interesting images of lung tomographies will provide the index structures to all images that are values of the attribute "lung tomography" of all objects of type Patient stored in the database.

4. An image definition language.

This section presents the extensions made to an object-oriented dialect of SQL, through commands aimed to define and retrieve images using the concepts presented, that make up the Image Query Language (IQL). The description uses the language keywords in capital letters, although the language is in fact case insensitive. These commands are divided into the image definition and the image manipulation commands. An image is treated as a set of values of attributes whose characteristic is "image", so applications do the real storage and manipulations of images using the "regular" commands of the SQL language.

4.1 - Defining image constants

The first step to define image indexes is the creation of image constants. This is done with the following command syntax:

```
CREATE [SUB] IMAGECONST
    <const_name> AS <locator>;
```

where <const_name> represents the identification of the constant to be used in subsequent statements. The optional clause SUB defines a constant image that can be part of others. The clause <locator> is a general way to indicate where the target of the command is, in this case, the image itself. It can point either to an external file, or to an already stored image. In the former case the syntax is:

```
<file type> <path>
```

where <file type> indicates a file outside the database through one of the following keywords: FILE to indicate a data file; TEXT for text files; or BINARY for executable files. The following syntax points to images already stored in the database:

```
<attr.charact> <obj.type> <obj.name>
    <attribute> <attrib.structure>
```

where <attr.charact> indicates the characteristics of the referenced attribute <attribute>, through the keywords: NUMBER, PROCEDURE, or IMAGE. Here the keyword IMAGE is used because in this context <attribute> always belongs to the image characteristic. For example, the following commands create the Image Constant Heart from a disk file and the Image Constant Lung from an image already stored as the second

image of the array of RM_Thorax_Exams of a Patient named John:

```
CREATE IMAGECONST Heart AS FILE
    ~image/heart.tiff;
CREATE IMAGECONST Lung AS IMAGE
    Patient John RM_Thorax_Exams[2];
```

If an image of a Liver can be found as part of another image, it is possible to indicate the image as a sub-image constant by the command:

```
CREATE SUB IMAGECONST Liver as IMAGE
    Patient Bob RM_Abdomen[5];
```

4.2 - Defining image parameters

The specification of an image summarizer must represent each image parameter returned when it is executed over an image and an image constant. The association of each image summarizer with the image parameter that it extracts is described later in this paper. The definition of an image parameter has the following syntax:

```
CREATE IMAGEPAR (<data_type> <var_list>
    [RANGE <value> | FROM <value> TO
        <value>]);
```

The RANGE or FROM ... TO clauses give the tolerance threshold needed to put an entry of each image in the access structure of that parameter of that summarizer. For example, the following commands define some image definition variables to be later associated to the Match and Histo summarizers:

```
CREATE IMAGEPAR INT CorrelationRate
    FROM 0 TO 100;
CREATE IMAGEPAR INT CoordinateX,
    INT CoordinateY;
CREATE IMAGEPAR DECIMAL Scale
    FROM -1 to 1;
CREATE IMAGEPAR INT MediumLowGrey
    RANGE 256;
```

4.3 - Defining the image features to be extracted

a) Defining image processing methods to act as Image Summarizers.

The image summarizers must be previously developed in a programming language as an Image Processing Method (IPM). The syntax to assign an IPM as a summarizer is:

```
CREATE SUMMARIZER <summarizer_name>
    AS <locator> [<restriction>]';'
```

where, <summarizer_name> is the name to be used as a reference in subsequent query statements and expressions, and <locator> indicates where the IPM is. The following predefined IPM was implemented for use as image summarizers:

a) HISTOGRAM: maps the density distribution of the image to a defined number of levels; and
b) MATCHING: Searches for a specific pattern into another image.

These IPMs were stored as image-summarizer attributes of objects of type Image_operator called Histogram and Matching, respectively. For example, it is possible to define the image summarizer Match through the following command:

```
CREATE SUMMARIZER Match AS
    PROCEDURE Image_operator Matching
        Image-summarizer;
```

b) Defining method execution

The optional clause <restriction> permits the user to control the execution of the IPM, assigning the IPM to a set of parameters called Internal Variables. Each internal variable can receive a set of values and returns a value. If no value is received, a single default is assumed. Internal variables must be attached to an Image Parameter, so the user of the IQL can control the values returned from the internal variable. The values passed to an IPM are defined in the <restriction> clause, whose syntax is as follows:

```
(<ImagePar Name> <Internal Variable Name>
    [value | RANGE <value> | FROM <value>
        TO <value>]
    , ... ) ;
```

The Matching IPM was implemented with the following Internal Variable:

INT ROTATE: allows the constant image to which the modifier is associated to appear rotated at some angle (in degrees) in the compared image;

INT TRANSLATION: permits the constant image to appear translated by a number of pixels in the compared image;

DECIMAL SCALE: allows the constant image to appear in the compared image in a different scale, represented by an absolute value;

If not specified, each of these variables assumes the default zero.

The Matching IPM produces the following

results:

DECIMAL CORRELATION: the correlation between the two images;
INT COORDINATEX: the x coordinate in pixels of the best correlation obtained;
INT COORDINATEY: the y coordinate in pixels of the best correlation obtained;
DECIMAL SCALE: the Scale between the images if they are fully compared.

The Histogram IPM has only the Internal Variable INT LEVEL implemented. It is used to establish the number of levels used to calculate the histogram - if not specified, it assumes the number 256. This IPM gives the following results:

DECIMAL MINIMUN: The minimum grey level obtained;
DECIMAL MAXIMUN: The maximum grey level obtained;
DECIMAL AVERAGE: The average level obtained;
DECIMAL DEVIATION: The standard deviation level obtained;
INT HLEVEL[ ]: an array of the grey levels obtained.

The previous example does not specify restrictions, so the Summarizer Match defined there does not allow rotation, translation or scale: the default for each is zero. To define the image summarizer Histo7 which generates histograms with seven levels, the following command can be issued:

```
CREATE SUMMARIZER Histo7 AS
    PROCEDURE Image_operator Histogram
        Image-summarizer
        (Level_Number LEVEL 7);
```

4.4 - Defining image summarizer modifiers

Summarizers can be specialized to complement the description of the summarizer behavior, through *modifiers*. A modifier is either another IPM or an already existent one with an added set of restrictions. The syntax to define a modifier is:

```
CREATE MODIFIER <modifier_name> ON
    <Summarizer Name>
    AS [<locator>] [<restriction>]
```

If <locator> is omitted, then the same IPM is used. The following is an example of this syntax:

```
CREATE MODIFIER Partial_Match ON Match
    AS (
        Rotate_30     ROTATE RANGE 30,
        Trans_10      TRANSLATION FROM -
                10 to 10
        Scale_5       SCALE RANGE 5);
```

4.5 - Setting the relationships

It is not necessary for each stored image to be compared with the full set of image constants. The definition of what pair <summarizer, image constant> needs to be applied over each incoming image is reached in two phases. The first corresponds to the specification of objects of the type "Comparison Group" shown in the meta-schema of figure 1, using the following syntax:

```
SET COMPARISON <comparison_group>
    CONSTIMAGE <const_name>
    CHARACTERIZED BY
        <summarizer_name>
    DESCRIBED BY <var_list>';'
```

This command states that an object of type Image_constant identified by <const_name> must be compared with each image to be stored using the image summarizer identified by <summarizer_name>, and these comparison results are gathered in the parameters listed in the <var_list>. It also gives the name <comparison_group> to this comparison. For example, to specify that the Image Constant Heart needs to be processed by the Histo7 and Match summarizers, the following commands must be issued:

```
SET COMPARISON Heart_Histogram
    CONSTIMAGE Heart
    CHARACTERIZED BY    Histo7
    DESCRIBED BY Average, Deviation,
        Hlevel[3] MediumLowGrey;
SET COMPARISON Heart_Position
    CONSTIMAGE Heart
    CHARACTERIZED BY    Matching
    DESCRIBED BY  Correlation
        CorrelationRate,
        CoordinateX, CoordinateY;
```

It is possible to define different comparison groups for the same summarizer and obtain different results. This provides flexibility to the queries, in the sense that only the comparison aspects required in a query need to be described. However, to be useful, a meaningful project of the intended queries as part of the database project is needed. For example, the same summarizer Histo7 can describe another image constant using a different characteristic set, as shown in the

following statement:

```
SET COMPARISON Lung_Histogram
    CONSTIMAGE Lung
    CHARACTERIZED BY Hsto7
    DESCRIBED BY Hlevel[3]
        MediumLowGrey;
```

This example considers that when the Histo7 image summarizer is associated with an image constant named Lung, only the intensity of the MediumLowGrey level (which corresponds to the third position of the histogram array) is important to describe the Lung_Histogram relationship.

4.6 - Linking Object Attributes with Image Characteristics

The second phase to define what pair <summarizer, image constant> is to be applied over each incoming image is made when an object type is created. To each attribute of the characteristic image, a link is made to all the comparison groups of interest to this attribute. These links, which correspond to the "Relates to" attribute of the "Image Attributes" objects in figure 1, are established by the following clauses: in the Create Object_type command (the Create Object_type is the equivalent of the Create Table in standard SQL):

```
<atrib.name> IMAGE <Attribute_Type>
    DESCRIBED BY <Comparison_group_list>
        [',']
```

For example, if the images stored as values of attributes ThoraxExam of objects of type Patient are compared with the Heart_Scale, Heart_Occurence and Lung_Histo comparison groups, the object type patient can be created by this command:

```
CREATE OBJECT_TYPE Patient AS (
    Name            VARCHAR(50),
    ToraxExam  IMAGE
    DESCRIBED BY Heart_Scale,
        Heart_Histo, Lung_Histo);
```

To change the application schema, there are also equivalent commands to the Alter and Drop ones from SQL. Once the image index structure of an application is defined, the database can be initialized and the images stored using "traditional" SQL commands, such as the Insert, Delete and Update commands. Whenever an image is stored as a value of an attribute of an object of a given type, all Image Summarizers of the comparison groups associated to that attribute must process it.

4.7 - Image query commands

Queries are formulated through a modified SELECT-FROM-WHERE command, and the condition in the WHERE clause describes the images intended to be retrieved. For example, to retrieve a set of thorax exams of some patients where a heart appears, accepting only those images that have a correlation rate between 80% and 100%, can be expressed as:

```
SELECT Name, ThoraxExam
FROM Patient
WHERE Patient.ThoraxExam.Heart.
        Match.CorrelationRate
    IN From 80 to 100;
```

This query establishes the acceptable values for the parameter CorrelationRate returned when the image Summarizer Match is used to compare the image constant Heart with the Thorax Exams of the objects Patient. This query results in all patient names whose Thorax Exam images have a Heart like the one depicted in the Heart image constant. The processing to recognize this image constant in all stored images of the Thorax Exams was actually done when each image was stored. However, when the query is issued, no further image processing was executed: a fast search through an index structure selects the target images.

5. Conclusion

This work describes the use of the concept of images as a data type, Image Constants and Image Summarizers as a way to pre-process images and speed up the content-based retrieval of images stored in an Image Database. The concepts presented herein can be used in cooperation with more traditional searches based on descriptive text strings and icons attached to the images, although the described method can almost always replace the others. The proposed approach maintains the accuracy of the search by storing the entire image. Image retrieval processing is accelerated by distributing the workload of the image processing methods at the storing time. In addition to taking advantage of the time when the workload is slower, the results of the processing are stored so each computation is done only once.

This approach can use a set of indexes to quickly discard a lot of images that undoubtedly are not part of the response set of a query, enabling presentation of only a reduced set of images to the more time-consuming filters needed to complete the answer. A new set of commands to enable the definition of such

structures was added to an object-oriented version of the Structured Query Language (SQL).

The fact that modifications in an image by a given image processing algorithm correspond to equivalent changes in the image parameters returned by an image summarizer, permits manipulation of the index structures accordingly. Thus, the image modifiers establish operations over the images homomorphic to operations over the index structures. However, the operations over the index structures are much less time-consuming, so when queries require operations over image constants to be used in the image comparisons, those operations over images can be mapped to operations over the index structures. This leads to a new range of optimizations that can be explored in the image content-based queries, to be studied in more detail in future.

6. References

[1] G.F. Dutton, "The Current State of PACS", *Applied Radiology*, Aug. 1990, pp 15-19.

[2] S. Chang and A. Hsu, "Image Information Systems: Where do we go from Here?", *IEEE Trans. on Knowledge and Data Engineering*, Vol. 4, No. 5, Oct 1992, pp. 431-442.

[3] E.L. Vidoto, H. Panepucci, A. Tannús, M.J. Martins, "A New Type of Head Coil for MRI in Ultra Low Magnetic Field", in Socity of Magnetic Resonance, Second Meeting, June 20 - July 06, 1986.

[4] A. F. Cardenas at all, "The knowledge-based Object Oriented PICQUERY+ Language", *IEEE Trans. on Knowledge and Data Engineering*, Vol. 5, No. 4, pp. 644-656, Aug. 1993.

[5] V.N. Gudivada, V.V. Raghavan, "Content-Based Image Retrieval Systems", *IEEE Computer*, Vol. 28 , No. 9, Sep. 1995, pp 18-22.

[6] C. Traina Jr. and J.F.W. Slaets, "The Object Representation Model ", (in portuguese) *Technical Report of ICMSC* - University of São Paulo Nº 104, Feb. 1992.

[7] E. Y. Senzako, "SisMatch - Matching for Magnetic Resonance Imaging", (in portuguese) MS Theses, University of São Paulo Oct 1996.

[8] R. R. Santos, A.J.M.Traina and C. Traina Jr., "An Image Definition and Manipulation Language for Object Oriented Data Base" (in portuguese) *Proceedings of XI Brazilian Symposium on Database*, São Carlos - SP, Oct. 1996, pp 127-142.

[9] A. Guttman, "New Features for Relational Database Systems to Support CAD Applications", *PhD Theses*, University of California , Berkeley, Jun. 1984.

[10] L.G. Brown, "A Survey of Image Registration Techniques", *ACM Computing Surveys*, Vol. 24, No. 4, pp. 325-376, Dec 1992.

[11] C.H. Chow and X.C. Chen, "Moment-Preserving Pattern Matching", *Pattern Recognition*, Vol. 23, No. 5, Oct 1990, pp. 461474.

[12] A. Tannús, A. Torre Neto, M. J. Martins, T. J. Bonagamba, N. Beckman, J.F.W. Slaets, H. Panepucci, "Architecture of 2.0 Tesla NMR Tomograph", in 9 º International Socity of Magnetic Resonance, Aug 6-12, 1994, S. Francisco, USA, vol III, pp. 1101-1109.