B-SPID: An Object-Relational Database Architecture to Store, Retrieve, and Manipulate Neuroimaging Data

Barrou Diallo, Florent Dolidon, Jean-Marcel Traverre,
and Bernard Mazoyer*

*Groupe d'Imagerie Neurofonctionnelle UPRES EA-2127,
Université de Caen & CEA LRC no 13, France*

Abstract: We propose a hardware and software architecture to respond to crucial problems in the neuroimaging field: storage, retrieval, and processing of large datasets. The B-SPID project, here discussed, concerns the processing of neuroimages and attached components stored in an object-relational multimedia database management system (DBMS). Advanced bioinformation concepts are exploited in this project such as large scale data storage, high level graphical user interfaces and 3D graphical processing and display of data. Our database implementation is based on standard programming components, runs on several UNIX platforms and is written to be evolutive. Queries on this database are designed to obtain and display from neuroimaging data several types of results (pictures, text, or 3D graphical shapes) on heterogeneous systems. *Hum. Brain Mapping* 7:136–150, 1999. © 1999 Wiley-Liss, Inc.

Key words: database; neuroimage; PET; MRI; metaanalysis

INTRODUCTION

Over the last decade, the brain mapping community has undergone considerable technological mutations such as three-dimensional anatomical and functional imaging. This revolution has allowed researchers to improve the global knowledge in the human cognition field. New imaging methods such as magnetic resonance imaging (MRI), functional magnetic resonance imaging (fMRI), or positron emission tomography

(PET) are nowadays powerful tools with which the study of the human brain can be done in vivo. Datasets acquired with these high-performance imaging devices have the global characteristic to improve resolutions in both time and space. These technological advances have operated in two distinct ways: acquisition dedicated devices and image analysis programs [e.g., Benton, 1996]. A correlate of this obvious fact is the growing amount of collected data. But, as previously pointed out by Roland and Zilles [1996]: “science will not automatically progress if just given sufficient amounts of data.” A major difficulty in implementing a database which contains files of neuroimages lies in the organization of a large amount of data and in the minimization of the user’s final search time. Organizing such data will only be efficient with the use of adequate software like database management systems (DBMS). DBMS have the ability to define, construct

This work was presented in part at the Third International Conference on Functional Mapping of the Human Brain, May 1997, Copenhagen.

*Correspondence to: Professor Bernard Mazoyer, GIP CYCERON, Bd Henri Becquerel, BP 5229, F-14074 CAEN cedex, France. E-mail: mazoyer@cyceron.fr

Received 5 December 1997; accepted 31 July 1998.

and organize relationships of new data structures in a specific area.

Database technologies have different levels of abstraction depending on the desired type of result. The first level consists in storing data and offering tools to query the desired information and obtain textual or graphical results. The second type is the data organization in a special manner so that content-based queries can be efficiently performed (e.g. statistical calculation on an image). The third type introduces knowledge-based information (from medical specialists) to process coupled studies and incorporate database ability to reason about its underlying data.

The brain mapping field has undergone several database experiences. Jamzad et al. [1996] have developed a reporting tool to store PET studies and retrieve picture by the way of a text describing the patient information and the visual interpretation of the PET examination. In the Human Brain Project, Peterson et al. [1996] have designed an environment to run and store computational models and their results into a database. Databases have been also used to create powerful meta-analysis brain tools [e.g., Fox et al., 1994a], symbolic atlases of the brain [e.g., Natarajan et al., 1996; Heisenberg et al., 1995], and to offer the ability to perform complex queries and reasoning over abstract data. Large projects have been initiated around databases [e.g., Mazziotta et al., 1995; Roland et al., 1996]. However, to our knowledge, the present paper

reports one of the first attempt to integrate complex neuroimaging data into an object-relational database model.

Here, we present the architecture's main principles of our multi-media neuroimages database named B-SPID (for Brain Structural and Parametric Image Database). This project was designed to allow the study of human brain variability both from the anatomical and the functional points of view. B-SPID was conceived in order to allow storage, query and statistical operations on multiple data types: MRI images (either functional, anatomical, or spectroscopic), PET scans, 3D curves manually drawn over sulci, 3D meshed volumes of interest, and program outputs. Our goal was to build an integrated system from which several operations would be possible: running batch programs over multiple protocols, performing meta-analysis, deforming and registering images, computing statistics over volumes of interest (VOIs), segmenting MRI images using data fusion algorithms between different types of acquisitions, and visualizing results using 3D rendering algorithms.

This paper is composed of three main parts: (1) The B-SPID hardware data storage strategy; (2) the architecture and software implementation of the B-SPID database under Illustra[™]; (3) finally a discussion about the main problems in implementing a neuroimage database and about the global development strategy of B-SPID vs. other medical imaging systems.

OUTLINES OF THE B-SPID PROJECT

The B-SPID database server is one of a powerful object-relational servers whose main characteristics are an efficient storage and retrieval of several kinds of information (images, text, 3D polygons, 3D regions of interest, subject information).

B-SPID architecture and software implementation

The DBMS

B-SPID is built with the help of the Illustra[™] DBMS which manages the internal data representation (see Fig. 1). On the top of this input-output layer, we have built software components to specifically respond to the management of neuroimaging data. The Illustra[™] DBMS [e.g., Kim, 1995] is classified in the object-relational category because of its ability to handle in the same manner textual fields or geometric features like rectangles or 3D meshes. Its engine allows the system user to program and plug new components in the kernel with the help of dynamically linked libraries.

Abbreviations

ACR-NEMA	American College of Radiology and National Electrical Manufacturers Association
API	application programming interface
B-SPID	brain structural & parametric image database
DBMS	database management system
DAT	digital audio tape
DICOM	digital imaging and communications in medicine
DLT	digital linear tape
fMRI	functional magnetic resonance imaging
GUI	graphical user interface
HIS	hospital information systems
ICBM	international consortium on brain mapping
JSD	Jackson system development
LUT	lookup table
MEG	magnetoencephalography
MRI	magnetic resonance imaging
OSF	open software foundation
PACS	picture archiving and communication systems
PET	positron emission tomography
RIS	radiology information systems
ROI	region of interest
SADT	structured analysis and design technique
SQL	structured query language
VOI	volume of interest

Figure 1.

This diagram illustrates the link between a database management system (DBMS) and a database program build over this first system. The DBMS functions looks like a extended powerful file system which has its own language, its complex access rules and a sophisticated storage device management system. First, a program accessing this special file system (like B-SPID) has to build its data

architecture into the storage space and define the rule to access them via the DBMS. Second, after the retrieving of a specific data set, the host program can use the DBMS high-performance to perform complex operations like image segmentation or scientific visualization.

ies. These components, represented as simple functions or procedures, consist in SQL statements as well as C functions. The DBMS kernel can be so enhanced as to provide facilities in handling specific data types. The basic rules and functions of the initial kernel are fully SQL-92 compliant [e.g., Cattel, 1996], but all object-oriented features are automatically included as new specific instructions or reserved words. A second important aspect of the Illustra's application programming interface (API) remains the easiness in managing a multi-part project. For instance, to plug our specialized libraries (digital operations on 3D volumes, I/O processing) in the kernel and new functions to the SQL parser, we only had to define some interface routines with our C libraries but no change had to be done in our initial code. These functions and datatypes were dynamically available without recompilation of the database kernel.

Development language and programming environment

Portability of our applications was an essential point that has constrained the choice of the programming language. We had to port our code on multiple platforms and operating systems such as SGITM, SunOsTM, and PC running LinuxTM. Because of its high portability, we have chosen the C ANSI language to build our applications and, in general, only ANSI certified or standard software components. All our routines are written in an object-oriented fashion and stored into specialized libraries in order to improve the adaptability capacity of our code. The graphical components were developed under the X-WindowTM system X11-R6 (X Consortium Inc, Cambridge, MA) because of the stability of its protocol and its portability on many platforms. The final user interface was designed with the OSF MotifTM 1.2 widget toolkit (Open Software

Foundation) [e.g., Heller, 1991]. To improve the portability of our graphical routines, we have developed many graphics libraries at different levels of interaction: (1) user interaction aspects to manipulate in the same way input-output commands to the database; (2) real-time 3D display of the queries. These routines are easily usable because of a comfortable graphical user interface (GUI). This critical point has required the use of an interface builder named Builder Xcessory Professional 4.0 (ICS Cambridge, MA). This tool allows us to fast prototype and develop our graphical interface.

Input/output operations

One of the most fundamental aspects in building a database is to provide tools to import new data. These tools have the difficult task to convert existing image formats into the database format. An automatic insertion of a new image implies the recovery of information from the original image header and a filtering of the data in order to transform them into a standard format. All image file formats do not provide the same type of information such as pixel size or byte-ordering. We have created a meta-class based on a simple concept: the database owns essential information like the image size, the number of bits per pixel or the image type to perform most operations. A special child table is added to store data headers in their native format so all reverse transcoding remains possible. The pivot format in B-SPID is the Analyze[™] file format (version 7). B-SPID can import most of the classical image formats like ACR-NEMA or DICOM. However, using the concept of the specific library, B-SPID allows the user to add its own filter to import his data.

Implementation of object-classes

Since Illustra[™] is an object-oriented database, information is stored in a hierarchical manner into classes. All object management facilities, such as inheritance, are available and neuroimaging remains a field where different objects may share the same father (MRI acquisition may be a subset of raw data for example). Figure 2 shows an example of such a relationship between database objects. In our project, we currently use several types of relations: a super-class named "DataFlow" (see Fig. 3) manages the internal relations between different types of data. The "Voxel Type" class transforms image volumes and 3D vectors into a single space (either Talairach or subject space). Once all the data needed to perform a transformation are aligned,

converted and normalized, the database kernel allows child processes to produce the result. When the transformations are completed, B-SPID computes statistical parametric maps over the result and stores them in specialized classes.

Image Objects

At a higher level, B-SPID distinguishes several types of image objects during the processes. They may be categorized as Computed, Parametric, Resliced, or Classified.

- Computed images are results of digital image processing programs. They may have different characteristics in size and information coding but are semantically attached to an original picture.
- Parametric images concern all computer-generated images which represent maps of relations between parameters. They may be temporarily-stored images used in successive processing stages or final result images.
- Resliced images are geometrically transformed images (translation, rotation, zoom) to match a template image. These transformations often include interpolation procedures and this is why these images may be considered as special case of computed images.
- Classified images are parametric maps generated by simple or complex segmentation tools that decompose the initial picture into labels and connected regions. This type defines a set of regions which have a unique colour (or label). Classified images are an extension of binary images.

Internal image formats

The main characteristic of our database is the multiplicity of the data it handles and the correct handling of value ranges. MRI data are, for example, generally acquired on unsigned 16 bit words for each voxel. But, in fact, only about 10 bits are used. Similarly, PET images are formed by signed 16 bits voxels with only eight to nine significant bits. Working on real-valued data (regional cerebral blood flow for instance) automatically implies that over/under-flows have to be minimized and that all classical ranges have to be coded. Because of the wide variety of data types represented by images, we focused our work on the most representative: one, eight, 16, or 32 bits for integer values and single/double precision for floats. All these types are handled automatically along all the

```

create type image_t (name varchar(30), width int, height int, slices int);
create type PETData_t (count int, tracer text, duration time);
create type MRIData_t (TR float, TI float, TE float);
create type MergedPicture_t (colormap int[256], min int, max int);

```

```

create table IMAGE of type image_t;
create table PET of type PETData_t under IMAGE;
create table MRI of type MRIData_t under IMAGE;
create table Picture of type MergedPicture_t under PET, IRM;

```

Figure 2.

Illustration of using a table hierarchy to solve the dependences between two types of images acquisitions (PET and MRI) with the Illustra[®] DBMS. First, we create new object types named `image_t`, `PETData_t`, and `MergedPicture_t`. These new types can then be used as normal type in all SQL statements. Second, we create

relational tables in a hierarchical manner: the supertable `IMAGE` has two childs named `PET` and `MRI` which contain specific data. These two childs have one common child named `Picture` which is a display table.

processes, and type-conversion tools have been implemented to transform (with several rules) one datatype to another. We were especially careful to keep at each level of the processing and queries, the entire information stored in the image volumes. In general, no automatic processing will damage the content of the original image or loose information in storing the intermediate computed results. For example, when a function sums two eight-bits images, it will store the results into a 16-bit picture.

Geometrical objects

Stored images are often accompanied by geometrical objects. These objects, generally an important set of points, may compose anatomical volumes of interest (VOIs) or computed features like segmented zones. All these objects have the particularity of having the possibility of interchanging themselves into other datatypes in order to perform specific operations. For example, a 3D volume can be cut to produce a set of

Figure 3.

B-SPID is built into classes (object entities) and this simplified diagram illustrates the relationship between the main components of the B-SPID kernel. The class Dataflow kernel receives information of different types (images, vectors, protocols). Its goal consists in formatting the data into a common space organizing them into the database form. The Voxel Type class permits the logical link

between raw data formatted by the last class and the rest of the program. This class receives information from the instrumental part as well as from statistic programs designed to compute morphological parameters. Finally, the Voxel Type class plays a fundamental role in displaying the database content via a 2D indexed bitmap and an appropriate colour map.

slices and inversely a set of slices can be aligned and stored as a single 3D volume. Some transformations are possible in a single way such transforming 3D objects into 2D objects.

Geometrical objects definition

All object coordinates are expressed as local coordinates stored between 0 and 1. To process a geometrical form, programs have to find the associated matrix (2D or 3D) which permits the computing of affine transformations (translation, rotation, and zoom) and then transforms the set of points which compose the object into the global space. 2D and 3D matrices are stored into arrays of double precision real values. Points are stored as couples (x, y) or triplets (x, y, z) of real coordinates depending on the dimension of the space in which they are described. Bitmaps are special slices

designed to be displayed on the computer screen. They do not contain the real data values but indexes referring to a colour lookup table (LUT). Regions are sets of connected points either in 2D or 3D space.

Object transformation rules

To transform 2D objects into 3D objects, programs (or users) have to build a transformation matrix and then build database objects named Matrix2d or Matrix3d. 2D points and segments cannot directly give bitmaps, but have to be drawn into a slice that can be subsequently transformed to a bitmap. A bitmap cannot be geometrically transformed, it does not have the ability to have an associated matrix. To recover its ability to be processed, a bitmap must be back-transformed into a slice. Geometrical forms (lines, points) can give their discretized equivalent in the

Figure 4.

Illustration of the relational dependencies between four main tables. These SQL table samples are linked together with relational links. For example, the table *mridata* contains a primary key named *idima* which is logically linked to all other tables. These classical relational links remain possible in a object-relational database.

same dimension, for instance a 2D segment can generate a 2D slice, a 3D point can give a 3D volume. 3D volumes can give 3D regions, which themselves can produce 3D meshes. But volumes cannot directly produce 3D segments: they have to be transformed into points before; in the same manner, slices cannot produce 2D segments before being transformed into 2D points. 3D segments cannot produce 3D regions because they have to be transformed into 3D points, the same rules holding valid for 2D segments. 3D meshes can give 3D points or 3D lines, but not directly 3D regions. They need first to be transformed into 3D points before. 3D regions can, however, directly give 3D meshes.

Main B-SPID tables

Four main relational tables are used to access the most important data information (see Fig. 4). The

imageo table describes the internal geometrical features of a volume (dimensions, voxel sizes). The *mridata* table stores the MRI specific information whereas the *imadesc* table contains global features like the byte alignment style or the 3D acquisition orientation. The table named *dataaccess* is a logical link between the database and the external world. This table contains the access type (inline, offline, or archive) and eventually the original file name. A unique database query can then transform an internal data to a flat file reusable with an external software.

Database queries and results

The basic idea of a client/server application is to perform all of the database processing on the server. The client sends a request to the server and the server responds with the requested data. The client and the

server are usually distinct, running on different machines connected via a network. The client-server model is a particular and simplified case of distributed systems in general. It only means that all queries are emitted by only one site and processing is performed by another distinct site. The client-server model may be put at fault by a database overload and, in this case, a distributed model is a requirement. In general, this last model is high-performance to improve the interactivity aspect but complex to implement. Servers also have to be well synchronized for queries that need updates on records in order to preserve the unity and coherence of data. However, one can note that the server's response speed is not only directly correlated with the amount of stored data, but to the manner data is organized and indexed. If many users run several concurrent access, the rate of the network layer and the performance of the input-output devices will determine the overall response time to the end-user. The SQL-92 RDA (Remote Data Access) standard allows the implementation and the management of multi-server transactions. We have tested it, but not currently fully implemented. One client has the ability to switch from one server to another during a same transaction so part of this transaction is performed by a server and part is processed by another. This system is efficient if automatic connection/disconnection rules are established in the database kernel. These rules, which depend on the processing context and the application complexity, need some experience to be efficiently defined. This modern architecture was designed to free the client (usually a desktop computer) and to reduce the global network load. In a X-Window environment, the B-SPID user has the possibility to build his query on the Motif-based interface (see Fig. 5). In this case the network load may be sometimes important. X-Window contains a complete and verbose protocol which needs, for some applications, a large network bandwidth, typically when allocating colours on the client's X-server or redrawing a complex graphical interface. However, these X queries are mainly performed once, at program launch. In the case of a simple database query, the conversation between the X clients and server summarizes in a mouse-event management and textual data exchanges. In the more sophisticated case of an image extraction and 3-D visualization query, the resulting object manipulation may be saccadic because of the TCP/IP network protocol which do not allow a constant flow of data packets. Our current hardware configuration is composed of a 100 Mbits/sec network and stations with 10 to 100 Mbits/sec ethernet cards. Stations are connected to two network HUBS, each corresponding to their transfert rate capabilities. With

this classical environment, we have not encountered any problem specifically due to the X protocol used as a graphics layer. On another hand, our choice was greatly motivated and guided by the high-portability of the X-Window environment. For example, a Sun user will find the same interface on an SGI workstation, as well as on a PC running Linux or a Windows-based desktop. This is unfortunately not true for other (sometimes lightweight) graphical libraries. The use of small machines as database client (little memory and CPU power) was successfully tested, B-SPID running under an X-Server installed on a Pentium-based PC (75 MHz/16 Mo, video 2 Mo, 3Com Etherlink II PCI).

From this graphical input, a stand-alone program (*generator*) constructs a SQL statement which is analysed and interpreted by the Illustra[®] kernel. This kernel, on the server part returns the result sets of the user's query. Note that this dataflow must be formatted to be read on the the user graphical interface. This work is performed by another stand-alone program (*formatter*), which also determines which type of data is returned and calls the adapted displayer. All stand-alone programs are written in the same client-server principle and communicate via Unix sockets. Thus, these programs may run either on a unique machine, or be launched on another machine declared over the network.

For textual data, the results are put on the screen into classical textfields, but for pictures or 3D meshes, special displayers have to be written. ROI, stored as 3D surfaces are displayed using a surface-based rendering method, but 3D voxel images are displayed with a volume rendering method that we have specially developed to be interactive and data independent.

The database interactivity may be deteriorated by several distinct factors: (1) the overall load of the computing environment (network load, client machine load and database server load), (2) the query complexity, and (3) the storage mode of requested data. The network load is a parameter which is difficult to control. However, databases queries can be optimized in order to minimize the number of transactions between the different software parts and thus to contribute to share the network ressources more fairly. The client machine CPU load is under the user responsibility: he has the ability to send queries while locally performing other tasks, but when the server sends the results, the CPU power will be shared between all background jobs (database programs included). Thus, the user has to note that not only the remote machine designed as the server does the work, but also his local

Figure 5.

Flow chart representing the global pathways of the B-SPID components. A query generated by the graphical user interface navigates between the desktop client part and the Server Machine part using stand-alone programs dedicated to first prepare the query for the database kernel and second format the resulting data for the graphical user interface.

station and perhaps intermediate machines. Finally, given that images are large, their storage mode varies in time (old data on DLT tapes and new/refreshed data on harddisks). Thus, for a same query, the database reaction-time can be noticeably different. The user interactivity is not so much affected by this constraint, because the user is always informed by the database processing progression. The remaining computation time can be estimated and the user has the ability to decide switching to a batch mode without graphical interface for long run queries. A preliminary sort of queries by their complexity can also be performed. This method allows a client to obtain information about the requested jobs in real-time, and without being freezed by monopolizing complex transactions. A such waiting queue management permits a best interactivity and fluidity of database commands, but depends on a consistent estimation of complexity by the database optimization kernel.

Two main types of image queries are resulting from our architecture: (1) queries for selecting one or more images that correspond to a certain criterion and (2) queries to choose an image, apply image processing and save the resulting computed image into the database. The first kind of query may be of the following: select ImageId from image where name like upper ('Da%x') and width <256; This simple query will extract the content of one field containing the unique identifier 'ImageId' of all stored pictures where the subject name begins with 'Da' and ends with 'x', and where the image width is smaller than 256 pixels. This query is not case-sensitive thanks to the upper keyword. The resulting identifiers can thus be used as arguments to another queries which will display image slices as a coloured bitmaps for example. The second kind of query is a bit more complex, dynamically calling loaded functions written in C in a separate library:

GetSlice (image, integer):image;

This function extracts and returns a slice from an image.

AddImage(setof(image)): image;

This function computes the sum of all images given as parameters. The results consists in another image, probably stored in a higher datatype as presented above.

Display(image): int;

This function displays the picture on the screen using the adequate displayer (2D or 3D) and a computed palette that maximizes the displayed contrast and intensity. All these functions show the global linkage of our project around the database itself. The object-oriented programming style makes our functions full reusable.

Hardware strategy for storing data

B-SPID data management consists in a specialized storage devices named “jukebox” (or “library”) accessed by a robot. This heterogeneous library can be implemented with different types of media (DAT or DLT tapes, optical disks or hard disks). The B-SPID database is stored in two media: classical harddisks and DLT tapes. The most widely used data remains stored in the fastest technology (hard disk). An automatic transfer program, which uses date and size criteria on database records, periodically moves data from harddisks to DLT tapes to free space. The automatic handling of online and offline data is performed by a high-performance storage machine.

An archive management system

Our storage machine based on the AMASS technology (Emass Inc., Englewood, NJ) permits the database kernel to query semi-off-line data in a transparent manner. This powerful system allows system administrators to manage libraries, disk drives and volumes (tape or optical disk). Amass is implemented at the Unix operating system level and supports applications by providing access to the files using standard file system semantics. This very important feature allows programs that currently run on standard UNIX file systems to use AMASS without any modification. This system making transparent access to the files stored in the AMASS filesystem allows disk drives and media (normally considered offline storage) to appear as a single online logical device, with a single mounted filesystem. Therefore, the extensive storage provided by a library appears as one large file system. This transparency allows our database server to request

information about physical offline data in the same manner as if the data was stored locally. The volume mounting and moving are totally independent from both the host and the client side. Consequently, the AMASS VFS (Virtual File System), considered as another UNIX filesystem like NFS (Network File System), does not necessitate any major modification on the server's operating system. This 64-bits file system allows a quasi-infinite number of stored files. We have inserted, into a preliminary sample database, more than 80 MRI volumes and about 30 PET scans. Each volume were stored in both the original acquisition format and a converted smaller format to speed up the data access. As an example, to retrieve an online set of ten images and their attributes using a single extraction query, the average query duration were about six seconds. Note that, if data are stored on DLT tapes, the query necessitate the storage machine which performs an internal search and thus, about 10 minutes are needed to browse a 50 Gigabytes DLT library.

Security and input-output performance

Recovery features are implemented in this hardware architecture at several physical and logical levels: (1) file level; (2) library level; and (3) volume verification and backup. Database back-up, full or partial, can be performed on dedicated volume sets in the library because a unique numerical identifier is assigned to each tape. This identifier is physically stored as a bar code. To improve the entire system performance, the Amass system has two levels of storage: a cache-disk and a jukebox/library. A two-step process is automatically performed to access a specified file: first it looks at the cache-disk and if not found the dedicated process searches in the DLT-library. For the migration, an independent daemon generates a list of eligible files on the client file system, along with their migration attributes. These last parameters are used to compute a score for each file and determine which one has to be moved to a DLT. The migration policy ensuring that the client file system never remains full, depends on several criteria like: file access frequency, price of the volumes and speed of their seek access.

A data analysis scenario

As a first database application, we have built a PET/MRI database to perform anatomical-functional correlations between twenty healthy subjects. For each, an MRI scan and a PET scans under several conditions were performed. The online stored MRI data were composed of more than 960 Mbytes which included

Figure 6.

Data Analysis scenario example. MRI acquisitions and PET scans under all experimental conditions are included in the B-SPID Database. Subject records (with information like handedness and gender) are also requested from the user. After this collection stage, a first data analysis consists in requesting the database for a subset of data suitable to the protocol (stage #2). Additional data like ROI description files (here the Planum Temporale region) can

be used to complement the protocol. Images are processed to be analyzed in a single pool: MRI data are reoriented to the PET scans, which also have been normalized. These new data are all inserted into the database to allow a statistical computation over the entire pool. Results which are images and statistical values, can thus be formatted and displayed. Here, 3D volumes and structural features (ROI) are displayed using a dedicated 3D real-time display engine.

the original, the edited and finally the PET-registered volume. The same amount of data were needed to store the PET images (raw, normalized, and processed data) for about ten experimental conditions. All these data have to be processed at the same time using additional informations like anatomical 3D structures (region of interest, ROI) as well as stored individual subject characteristics (like handedness). With the use of B-SPID, we have computed individual correlations between the size of the Planum Temporale (known for its high anatomical variability) and the PET activations during several tasks. These correlations can also be computed with the subject handedness as a parameter. This example shows the implication of such a database in the data analysis process. From the data

collection stage to the statistical analysis, B-SPID allows a direct link between structural data ROI, image data, and subject records (see Fig. 6).

DISCUSSION

In this paper, we proposed a general database architecture for neuroimaging purposes based on the Informix-Illustr[®] DBMS. We have described the beginning of our database implementation and our development strategy based on portable components. B-SPID is among the first attempts to implement a general database in neuroscience that allows heterogeneous objects to be stored and processed at the same place.

Organization models for large data volume

First introduced at the beginning of the 80's [e.g., Chang et al., 1980a], image databases have undergone a fantastic development during the last years [see Carey et al., 1996] for a complete review of database models and technologies). This evolution has been followed by new concepts and models in knowledge engineering like the SADT type design [e.g., Marca et al., 1987] or the JSD design method [e.g., Sutcliffe, 1988]. However, with the widespread use of the object-oriented approach [e.g., Booch, 1993; Kim, 1995], the data management began a new era really more adapted to the imaging fields [e.g., Aubry et al., 1996]. The capacities for storage are rapidly growing today and different hardware components are now available on the market to store, in an efficient manner, large amounts of data [e.g., Lehmann et al., 1996]. We have chosen to disconnect the storage part (Amass system) from the organization part of our project (Illustra system). The Amass system deals with all files on our local network and its logical link with the database consists in a transparent virtual filesystem. Because of the growth of network connectivity and storage devices [e.g., Chong et al., 1996; Ikeda et al., 1996], some experiences in performing both image processing and database queries on large heterogeneous image collections have been realized [e.g., Carson et al., 1996; Arya et al., 1996]. Another way of organizing picture data is the content-based retrieval approach using image reference databases (IRDBs) [e.g., Tagare et al., 1997]. This approach is especially interesting in picture sorting and automatic classification and has many applications in computer-assisted diagnosis [e.g., Korn et al., 1996] and educational purpose [e.g., Bucci et al., 1996; Macura et al., 1995]. Architectural issues and design methods of this approach were well exposed by Wong et al. [1996]. A similar approach in linking both text and image data have been used by Dionisio et al. [1996] to organize clinical data.

Picture archiving systems versus research databases

Nowadays, several systems exist in the field of applied databases: picture archiving and communication systems (PACS) [e.g., Huang et al., 1992; Wong et al., 1992], hospital information systems (HIS) [e.g., Lowe et al., 1995a], radiology information systems (RIS) [e.g., Wimmer et al., 1991], and several research databases (genome mapping, pathology, brain map-

ping, etc.). These information systems are commonly based on alphanumeric data like pathology reports (free text fields) or images (computed tomography, magnetic resonance). Traditional query keys are composed of patient name, date of birth or sex. In research databases, the problem is more sophisticated and concerns entities about the data content, calculation on morphological picture aspects and complex retrieval paradigms. Several parameters can be extracted from raw data and content-based queries can be available. Spatial and temporal relationships between stored objects or events can be calculated using models like TEDM [e.g., Dionisio et al., 1996; Chu et al., 1992]. Our system dissociates medical information (text fields for the subject identity) which are processed by classical relational tables and binary data (images and attributes) stored in an object manner.

In the medical imaging community more and more experiences proof the need of a structured, high-speed and multimodality database. Joubert et al. [1994] worked on knowledge representation, natural languages, and understanding in medical information. Ikeda et al. [1995] designed a distributed database to perform a computer-aided diagnostic system. This last Informix database project combines data from both the Hospital Information System (HIS) and laboratory test results, medication, or diseases entities. Mun et al. [1995] explored the image database question in the teleimaging field to provide teleradiology and telepathology. ARCADIA, a system for the integration of angiocardigraphic data and images is another example of the use of a object-oriented database system to store medical records [e.g., Pincioli et al., 1995]. Lowe et al. [1995b] have designed a multimedia database system to store non textual clinical information and retrieval pictures over a network using an indexed Metathesaurus.

Database and knowledge modeling

The KMeD system [e.g., Chu et al., 1995], based on the GemStone[™] (Servio Corporation, Alameda, CA) database software, stores high-level semantic spatial and temporal information in the database. For instance, it enables complex queries about the spatial organization or relative position of graphical objects. To perform these queries, original images and precalculated contours are stored in stacks structures [e.g., Chock et al., 1985]. Spatial relationship can be calculated at different level: point level [e.g., Chang et al., 1980b], image level [e.g., Roussopoulos et al., 1988] or feature extracted level like with KMeD.

Picquery [e.g., Joseph et al., 1988] is a front-end language capable of expressing many new constructs. Its enhancement (Picquery+) [e.g., Cardenas et al., 1993] has been provided to fulfill Kmeds querying: multimedia database federation, support, visualization of data models, and knowledge-based hierarchy, query processing of temporal and object evolutionary events. These last development more focuses on the manner to intelligently query a database system and show the range of problems in implementing a usable general database. In the scope of B-SPID, our goal was first to offer a set of complex operations dedicated to the neuroimaging field and not to model the user knowledge.

QBISM (Query By Interactive Spatial Multimedia) [e.g., Arya et al., 1996], a logical and physical prototype for querying and visualizing 3D spatial data. This Starburst DBMS architecture [e.g., Schwarz et al., 1986] has been enhanced to handle spatial data types. QBISM is a compilation of tools: a DBMS prototype, a visualization front-end (Data Explorer/6000, IBM), and anatomic models in human brain imagery. This results in a time-efficient storage and query prototype which is not portable. The physical layer concerning the data organization is based on the Hilbert curve [e.g., Bially, 1969] which allows an efficient spatial data clustering but give an machine-dependent solution. B-SPID is based on another strategy: to open a picture in our program, the user has the choice in loading it from the network (using a path and filename) or from the database (using a graphical or textual query). The database access remains only a more intelligent and sophisticated manner to access the data.

Databases and brain mapping

The interest for probabilistic atlases is growing in the neuroimaging community [e.g., Roland et al., 1994], the ICBM project [e.g., Mazziotta et al., 1995], and the European Computerized Human Brain Database [e.g., Roland et al., 1996; Huerta et al., 1993] doing examples of attempts to pack neuroscientific research electronically. Databases have also been used for interpretation of magnetoencephalography (MEG) acquisitions and for aiding the planning of neurosurgical operations [e.g., Anogianakis et al., 1994]. Previous databases or atlases were dedicated either to cover the PET activation literature in providing a tool for metaanalysis [e.g., Fox et al., 1994] or to the efficient storage of patient MRI acquisitions and VOIs [e.g., Evans et al., 1991]. Such traditional database implementations often manage data interactions on a few data types. On the contrary, B-SPID allows interactions between ob-

jects and performs internally complex operations on a large palette of data. It is an integrated, fully portable and automatic kernel for storing, computing, visualizing and extracting complex experimental results.

A neuroimaging database has to provide the means to display its contents (i.e., pictures) as well as a management software to provide the ability to sort and display text. A major problem in the neuroimaging field is the representation of both the internal and external data. The ability to query, process, and display 3D pictures (using volume rendering techniques) or structures implies more parameters and computations than performing arithmetic operations on tables. Our database, as a complete product, manages data from the acquisition stage to the processing and displaying stages. All steps are represented by independent libraries, used in other satellite projects. For example, the user can ask the database (in a single query) to extract a picture, threshold its contents and display the contrast-enhanced result. The first part of this query is contained in the library which manages the kernel of the database and all input-output operations. The second part calls a function of our image processing library and then a library dedicated to the display is used to convert the data into a displayable form. All these steps remain transparent for the final user. When a user employs a another satellite program to display an offline picture, the same display library is used. The development of reusable softwares is a real challenge in a world where environments are daily evolving. We have oriented the design strategy of our work in an object-oriented plan to allow different software components to exchange their data and methods.

ACKNOWLEDGMENTS

All Trademarks are the properties of their respective companies.

The authors thank Nathalie Tzourio for her expert assistance.

REFERENCES

- Anogianakis G, Harding GF, Peters M, Apostolakis M, Foroglou G, Vieth J, Ioannides A. 1994. Biomagnetic methodologies for the noninvasive investigations of the human brain (MAGNO-BRAIN). *Comput Methods Programs Biomed* 45:111-114.
- Arya M, Cody W, Faloutsos C, Richardson J, Toga A. 1996. A 3D medical image database management system. *Comput Med Imaging Graph* 20:269-284.

- Aubry F, Chameroy V, Di Paola R. 1996. Implementation of a Medical Image Database Supporting Set-Oriented Structures and Knowledge Management Capabilities. CAR'96—(Computer Assisted Radiology). Amsterdam: Elsevier. p 496–501 (Abstract).
- Benton D. 1996. Bioinformatics—principles and potential of a new multidisciplinary tool. *Tibtech* 14:261–272.
- Bially T. 1969. Space-filling curves: Their generation and their band bandwidth reduction. *IEEE Trans Inform Theory* 15:658–664.
- Booch G. 1993. Next generation methods. Bringing order out of chaos. *Object-Oriented Analysis Design* 1:11–12.
- Bucci G, Cagnoni S, De Dominicis R. 1996. Integrating content-based retrieval in a medical image reference database. *Comput Med Imaging Graph* 20:231–241.
- Cardenas AF, Jeong IT, Barker R, Taira RK, Breant CM. 1993. The knowledge-based object-oriented PICQUERY + language. *IEEE Transactions on Knowledge and Data Engineering* 5(4):644–657.
- Carey MJ, DeWitt DJ. 1996. Of Objects and Databases: A Decade of Turmoil. *Proceedings of the 22th International Conference on Very Large Data Bases*. Mumbai (India): Morgan Kaufmann, pp. 3–14 (Abstract).
- Carson C, Ogle E. 1996. Storage and retrieval of feature data for a very large online image collection. *IEEE Computer Society Bulletin of the Technical Committee on Data Engineering* 19:120–146.
- Cattell RGG. 1996. *The Object Database Standard*. Morgan Kaufmann Publishers, San Francisco, CA.
- Chang NS, Fu KS. 1980a. A relational database for images. In: Chang NS, Fu KS (eds): *Pictorial Information System*. New-York: Springer-Verlag, pp 288–322.
- Chang NS, Fu KS. 1980b. Query-by-pictorial-example. *IEEE Trans on Software Eng* 6:519–524.
- Chock M, Cardenas AF, Klinger A. 1985. Database structure and manipulation capabilities of a picture database management system (PICDMS). *IEEE Trans Patt Anal Machine Intell* 6:
- Chong MN, Mu K, Low KK. 1996. Concurrent computing for picture archiving and communication system. *Med Biol Engin Comput* May:257–261.
- Chu WW, Lee RC, Chen Q. 1992. A temporal evolutionary object-oriented data model and its query language for medical image management. *18th International Conference on Very Large Databases (VLDB)*. Vancouver (Canada): Morgan Kaufmann, pp 53–64 (Abstract).
- Chu WW, Cardenas AF, Taira RK. 1995. KMeD: A knowledge-based multimedia medical distributed database system. *Information Systems* 20:75–96.
- Dionisio JD, Cardenas AF, Taira RK, Aberle DR, Chu WW. 1996. A unified timeline model and user interface for multimedia medical databases. *Comput Med Imaging Graph* 20:333–346.
- Evans AC, Marett S, Torrescorzo J, Ku S, Collins L. 1991. PET-MRI correlation in three dimensions using a Volume-of-Interest (VOI) atlas. *J Cereb Blood Flow Metabol* 11:A69–A78.
- Fox PT, Lancaster JL. 1994a. Neuroscience on the Net. *Science* 266:994–996.
- Fox PT, Mikiten S, Davis G, Lancaster JL. 1994b. BrainMap: A database of human functional brain mapping. In: Thatcher RW, Hallett M, Zeffiro T, John ER, Huerta M (eds): *Functional Neuroimaging Technical Foundations*. San Diego: Academic Press, pp 95–105.
- Heisenberg M, Kaiser K. 1995. The Flybrain Project. *Trends Neurosci* 18:481–483.
- Heller D. 1991. *Motif Programming Manual*. USA: O'Reilly & Associates, Inc, Sebastopol, CA.
- Huang HK, Taira RK. 1992. Infrastructure design of a picture archiving and communication system. *AJR* 158:
- Huerta MF, Koslow SH, Leshner AI. 1993. The Human Brain Project: An international resource. *TINS* 16:436–438.
- Ikeda M, Ishigaki T, Yamauchi K. 1995. Development of distributed image database combined with clinical information in hospital information system. *J Med Sys* 19:305–311.
- Ikeda M, Ishigaki T, Yamauchi K. 1996. Estimation of the size of the media necessary to construct a medical image database. *Comput Biol Med* 26:77–85.
- Jamzad M, Ishii K, Toyama H, Senda M. 1996. A human friendly reporting and database system for brain PET analysis. *Ann Nucl Med* 10:99–104.
- Joseph T, Cardenas AF. 1988. PICQUERY: A high level query language for pictorial database management. *IEEE Trans Software Eng* 14:630–638.
- Joubert M, Fieschi M, Robert J-J, Tafazzoli A. 1994. Users conceptual views on medical information databases. *Int J Bio-Med Comput* 37:93–104.
- Kim W. 1995. *Modern Database Systems: The Object Model, Interoperability, and Beyond*. ACM Press, Redding, MA.
- Korn F, Sidiropoulos N, Faloutsos C, Siegel E, Protopapas Z. 1996. Fast nearest neighbor search in medical image databases. *Proceedings of the 22th International Conference on Very Large Data Bases*. Mumbai (India): Morgan Kaufmann, pp 215–226 (Abstract).
- Lehmann KJ, Walz M, Bode A, Teubner J, Stahl J, Georgi M. 1996. Development of a clinical-radiological CD-ROM database for focal lesions of the liver. *Rofo Fortschr Geb Rontgenstr Neuen Bildgeb Verfahr* 165:380–385.
- Lowe HJ, Buchanan BG, Cooper GF, Kaplan B, Vries JK. 1995a. Image engine: an integrated multimedia clinical information system. *MEDINFO Proceedings* 8, pp 421–425. (Abstract).
- Lowe HJ, Buchanan BG, Cooper GF, Vries JK. 1995b. Building a multimedia database system to integrate clinical information: an application of high-performance computing and communications technology. *Bull Med Lib Assoc* 83:57–64.
- Macura KJ, Macura RT. 1995. Query by pictorial: an alternative to medical image retrieval. *MEDINFO Proceedings* 8:1438–1442 (Abstract).
- Marca D, McGowan C. 1987. *SADT, Structured Analysis and Design Technics*. New York: McGraw Hill.
- Mazziotta JC, Toga AW, Evans AC, Fox PT, Lancaster JL. 1995. A probabilistic atlas of the human brain: theory and rationale of its development. *NeuroImage* 2:89–101.
- Mun SK, Elsayed AM, Tohme WG, Wu YC. 1995. Teleradiology/telepathology requirements and implementation. *J Med Sys* 19:153–164.
- Natarajan K, Claridge E, Davis DN. 1996. The symbolic atlas of the brain: handling non-visual information associated with neuroanatomy. *Med Inf (Lond)* 21:1–21.
- Peterson BE, Healy MD, Nadkarni PM, Miller PL, Shepherd GM. 1996. ModelDB: An environment for running and storing computational models and their results applied to neuroscience. *J Am Med Inform* 3:389–398.
- Pinciroli F, Combi C, Pozzi G. 1995. ARCADIA: A system for the integration of angiocardigraphic data and images by an object-oriented DBMS. *Comput Biomed Res* 28:5–23.
- Roland PE, Zilles K. 1994. Brain atlases—a new research tool. *TINS* 17(11):458–466.

- Roland PE, Zilles K. 1996. The developing European computerized human brain database for all imaging modalities. *NeuroImage* 4:S39–S47.
- Roussopoulos N, Faloutsos C, Sellis T. 1988. An efficient pictorial database system for PSQL. *IEEE Trans Software Eng* 14:639–650.
- Schwarz P, Chang W, Freytag JC, Lohman G, McPherson I, Mohan C, Pirahesh H. 1986. Extensibility in the starburst database system. International workshop on object-oriented database systems. Pacific Grove, pp 85–92 (Abstract).
- Sutcliffe A. 1988. *Jackson System Development*. New York: Prentice-Hall.
- Tagare HD, Jaffe CC, Duncan J. 1997. Medical image databases: a content-based retrieval approach. *J Am Med Inform Assoc* 4:184–198.
- Wimmer MS, Blaine GJ, Jost RG, Moore SM, Studt JW. 1991. A distributed approach to integrated inquiry and display. *J Med Syst* 15:299–309.
- Wong AWK, Taira RK, Huang HK. 1992. Digital archive center: implementation for a radiology department. *Am J Radiol* 159: 1101–1105.
- Wong ST, Huang HK. 1996. Design methods and architectural issues of integrated medical image data base systems. *Comput Med Imaging Graph* 20:285–299.