A Distributed and Collaborative Intelligent System for Medical Diagnosis

Naoufel Khayati[a,b], Wided Lejouad-Chaari[b]
[a] High School of Engineers of Sousse, University of Sousse, TUNISIA.
[b] SOIE Laboratory (Optimization Strategies and Intelligent Computing), University of Tunis, TUNISIA.
naoufel.khayati@soie.rnu.tn, wided.chaari@ensi.rnu.tn

KEYWORD

Program supervision
Mobile agents
Knowledge model
Ontologies
Medical image analysis
Osteoporosis detection

ABSTRACT

In this paper, we present a distributed collaborative system assisting physicians in diagnosis when processing medical images. This is a Web-based solution since the different participants and resources are on various sites. It is collaborative because these participants (physicians, radiologists, knowledge-bases designers, program developers for medical image processing, etc.) can work collaboratively to enhance the quality of programs and then the quality of the diagnosis results. It is intelligent since it is a knowledge-based system including, but not only, a knowledge base, an inference engine said supervision engine and ontologies. The current work deals with the osteoporosis detection in bone radiographies. We rely on program supervision techniques that aim to automatically plan and control complex software usage. Our main contribution is to allow physicians, who are not experts in computing, to benefit from technological advances made by experts in image processing, and then to efficiently use various osteoporosis detection programs in a distributed environment.

1 Introduction

Medicine is considered as one of the large application fields of the image analysis and its processing. For example, image analysis is widely needed in the imagery by magnetic resonance, in the radiology to assist physicians in their diagnosis, and recently in the telemedicine. This analysis is considered by specialists in image processing in order to offer more effective programs and more efficient approaches. We experiment our work in the osteoporosis detection. The most challenging task is to characterize bone micro architecture by parameters that can be automatically estimated from radiographies and that can accurately detect and quantify alterations of bones. For this, many approaches have been developed [SEVESTRE-GHALILA, S. *et al*. 2004] [BOUHLEL, N. *et al*. 2009]. The First one presents an original approach using morphological tools to extract characteristic features of trabecular bone images.

To make such an approach for medical diagnosis, we determined an "image protocol" adapted to bone types (e.g. femur, wrist, vertebrae) and patient types (e.g. male or female, adult or child) for various image resolutions [KHAYATI, N. *et al*. 2008a] [LEJOUAD-CHAARI, W. *et al*. 2007]. Setting such an image protocol consists in planning a sequence of programs and tuning their input values (e.g., threshold values, filter size, etc.). This constitutes a tedious, time consuming task which requires both clinicians and image processing experts to collaborate.

Our solution was to provide an interactive tool which relies on artificial intelligence techniques to build image protocols in different situations. Moreover, we wish to make this system accessible to bone radiologists, a geographically scattered community.

In previous work, we validated our technological and architectural choices [KHAYATI, N. *et al*. 2013] and the morphological analysis [KHAYATI, N. *et al*. 2008b] [LEJOUAD-CHAARI, W. *et al*. 2007]. In this paper, we investigate further, the

resolution of security issues relevant to the use of mobile agents, and our knowledge models choices in terms of ontologies.
Hence, this paper summarizes all that we have done in order to offer a distributed intelligent assistant for osteoporosis detection, relying on mobile agent technology.

2 Osteoporosis Detection by Medical Image Analysis

In most cases, the diagnosis of the osteoporosis detection is based on the Bone Mineral Density, but this method has been proven ineffective in the elderly. To improve this diagnosis, researchers rely on the description of the bone microarchitecture and its quantification. That's why, several approaches and computer-aided methods of X-ray microarchitecture bone analysis have been developed to complement the Bone Mineral Density measurement technique, and thus, to enhance the diagnosis of radiographies in the osteoporosis detection.

2.1. Existing Approaches

Bone radiographies represent the most accessible technique of the imagery, whose images provide information on the bone structure visualized in projection. Two categories of parameters were defined to describe bone micro-architecture from radiographies. The first gathers the parameters resulting from texture analysis which present practical difficulties for their extraction and moreover, they have the disadvantage of being not interpretable by physicians [BENHAMOU, C.L. 1994] [PAQUET, V. *et al.* 1995].
The second category gathers parameters which aim at the description of the trabecular organization.
For this category, there are two types of approaches. The first is the spectral analysis from which we can deduce some parameters describing the presence of spans in some directions [WIGDEROWITZ, C.A. *et al.* 1997]. The second one, in which we are interested, is based on morphological analysis.

2.2. Description of the Morphological Analysis

The osteoporosis detection by a mathematical morphology approach [SEVESTRE-GHALILA, S. *et al*. 2001] [SEVESTRE-GHALILA, S. *et al*. 2004] is based on a process of quantification of the bone microarchitecture given in a radio image. This process consists of four main stages:

- During the first stage, the foreground of the radiographic image (Figure 1(a)) is extracted to make it more uniform (no variation in brightness).
- During the second stage, the skeleton is extracted from the x-rayed microarchitecture. Two choices are possible for this operation: the binary or the grey-level skeletonization. In general, the latter is preferred because the first one can cause a loss of information when binarizing the image.The image (b) of figure 1 shows the skeleton extracted from the micro-architecture with a grayscale skeletonization.

The skeleton is a highly connected network of segments. These segments are of two types:

- Vertical segments locate the compression trabeculae (caused by weight).
- Horizontal segments locate tension trabeculae (due to walking force) which will be the first degraded in the event of osteoporosis.

The Analysis of this skeleton is based on its decomposition into a network of intersecting points called nodes and segments that connect them. The darkest pixels in figure 1(b) are nodes.

- During the third stage, a classification of these nodes is then performed. The aim of this classification is to extract from the skeleton some information about the microstructures located in the image by each of the segments, and to extract also some parameters describing the network connectivity degree of the trabeculae forming the bone micro-architecture.
- During the fourth stage, the bone was analyzed by labeling segments in both directions (tension, compression). This will:

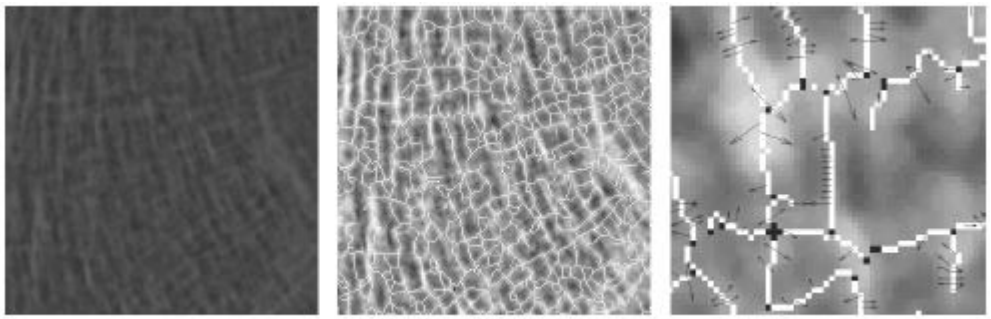

Fig. 1. Image (b) is the gray skeleton extracted from the original image (a). Image (c) is a zoom of the higher right corner of image (b) where the intra-projected distances are represented by arrows.

- Provide two sets of parameters (compression and tension).
- Compare both directions by the ratio of the segments number, the nodes number and the entropy of their spatial distribution in the image.
- Compute the intra-projected distances by measuring the distance between the two lines of maximum gradient directions around a segment. These intra-projected distances are represented by arrows in figure 1(c).

The obtained values for the different parameters will confirm the presence or absence of the osteoporosis disease.

3 Program Supervision Systems

Several methods of medical image processing were born and various approaches of the osteoporosis detection were proposed. For that purpose, various libraries of programs written in different languages (C, MatLab, etc.), were developed by diverse specialized teams with an aim of automating the image processing. But, the user of these libraries of image processing and medical imaging, often a radiologist, does not have competences in data and image processing allowing him to use them in an effective way. Moreover, physicians must focus themselves on the interpretation of the results and not on the way in which these programs are carried out and scheduled. Thus, techniques of Artificial Intelligence were proposed in order to assist a non-specialist in data processing for correct use of these programs in its field. These techniques are known as "Program Supervision" [THONNAT, M. *et al*. 2000].

3.1. Program Supervision

Program Supervision is an Artificial Intelligence approach which consists of the automation of management and the use of pre-existent programs. These programs are considered as “black boxes” and their application domain or their programming language is not relevant. The goal is not to optimize the programs themselves, but to assist program usage [THONNAT, M. *et al*. 2000].

To carry out a supervision task, a subset of programs is chosen, scheduled, and applied to a specific problem. This selection and this scheduling in various configurations are ensured by a supervision system, which, thanks to the reasoning of its engine and the knowledge contained in its base can free the user (the physician) to make this management manually. This enables a physician to run programs, to check the consistency of some image analysis methods, to compare algorithms, to evaluate results, to reconsider some parameters and to readjust them.

3.2. Knowledge for Program Supervision

A Program Supervision Knowledge-based System is composed of an inference engine operating on a knowledge base. The base formalizes the necessary knowledge about

programs and their use, mainly, the two major concepts of operators and criteria:

- Operators are of two types: primitive and composite.
  - A primitive operator represents a particular program i.e. an individual description of a program including its input/output arguments, the types and (default) values of parameters, the syntax to run the code, etc.
  - A composite operator represents a combination of programs and describes the data flows between their corresponding operators. These combinations correspond to decompositions into more concrete operators at various abstraction levels, either by specialization (alternatives), or by composition (sequences, parallels, etc.).
- Criteria represent decisional information, they are implemented by sets of decision rules which play an important role during the reasoning, i.e. choosing among various alternatives (choice criteria), adapting the programs execution (initialization criteria), assessing the results quality (evaluation criteria), and repairing a bad execution (repair criteria and adjustment criteria).

This information is given only once, by experts in the program use (experienced users or program designers), independently of any data. The only information that end-users must provide is the data of the particular cases to run.

3.3. Program Supervision Applications

Program supervision may be applied to different domains related to image, signal processing, or scientific computing.

- Astronomical Imaging (e.g. automatic galaxy classification [VINCENT, R. *et al*. 1997]).
- Vehicle Driving Assistance (e.g. road obstacle detection [SHEKHAR, C. *et al*. 1995]).
- Medical Imaging (e.g. chemotherapy follow-up based on Factorial Analysis of Medical Image Sequences [CRUBEZY, M. *et al*. 1997] [THONNAT, M. *et al*. 1999], and the segmentation of 3D MRI images of the brain [CRUBEZY, M. *et al*. 1997] [THONNAT, M. *et al*. 1999], osteoporosis detection in bone radiographies [KHAYATI, N. *et al*. 2008a] [KHAYATI, N. *et al*. 2008b]).

4 Medical Diagnosis Intelligent System

The proposed support system provides a collaborative framework driven by the following desirable properties:

- To assist clinicians in image protocol management, freeing them from performing computer science tasks, yet giving them access to up to date image processing.
- To allow experts in medical imaging to compare algorithms and to improve them incrementally, thanks to feedbacks from clinicians.
- To allow knowledge engineers to collaboratively construct knowledge bases for medical imaging.
- To do all the above, with people and resources distributed all over the world.

In the following, we focus on the details of the first property.

4.1. System Architecture

Our system for supervising osteoporosis programs, as illustrated in figure 2, includes a Knowledge-Based System (KBS), the osteoporosis programs described before, and a graphic interface. The KBS itself is composed of a knowledge base run by the inference engine called PEGASE [MOISAN, S., 2002] that implements the program supervision task.

The contents of the knowledge base allows the engine to choose, apply and adapt the osteoporosis programs to process numerical radiographic images provided by end users. This work gathers artificial intelligence and medical image processing experts collaborating with physicians specialized in osteoporosis diagnosis; thus, the current knowledge base incorporates the expertise from our image

processing and medical partners [SEVESTRE-GHALILA, S. *et al*. 2004].

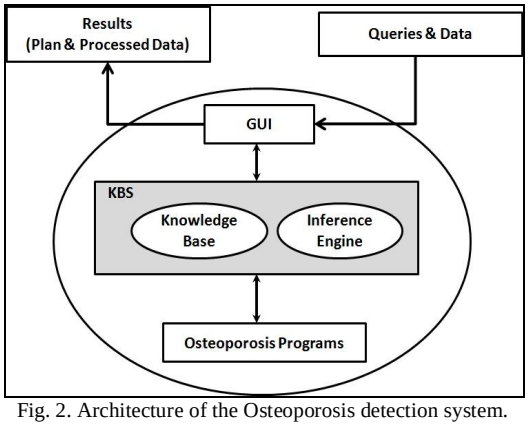

Fig. 2. Architecture of the Osteoporosis detection system.

4.2. Osteoporosis Programs Description

As said in section 2, our osteoporosis detection process belongs to the morphological analysis class. It consists in extracting morphometric features from a gray or binary skeleton. To this end, we propose several image processing programs. These programs often depend on tunable input parameters and some of them correspond to alternative algorithms for the same functionality. Setting a protocol thus requires tuning the parameters and choosing among the alternatives. To illustrate it we give a brief description of two important processing parts, namely, *skeletonization* and *analysis* of the resulting skeleton.

Skeletonization consists in thinning iteratively the image foreground (the trabeculae bone). The foreground is obtained by a thresholding step [SEVESTRE-GHALILA, S. *et al.* 2004]: the authors compare the original image (in this case a calcaneum image, figure 1(a)) to its local p-percentile computed on a sliding window of size w.

Then two thinning methods are proposed: a classical binary skeletonization and a gray thinning close to Mersal algorithm [MERSAL, S.S. *et al*. 1999]. The latter consists in two steps. First, the weakly connected pixels at the interface between foreground and background ("border pixels") are removed and their gray values are stored and ranked. Second, scanning these stored values with a step s, other weakly connected border pixels can be removed. The step s corresponds to the expected thinning precision.

The *analysis* process starts by splitting the skeleton into sets of pixels: the set of intersection points and the segments that connect such points. The result is illustrated in figure 1(b). The segments are labeled according to their orientation as compression (vertical segments, related to the patient's weight) or tension (horizontal, due to walking force). At this stage, morphological features providing reliable information about bone micro-architecture are computed for each segment. An inter-projected distance is estimated as the length of a segment. An intra-projected distance is also computed by measuring the distance between the two lines of maximum gradient directions around the segment. The gradient reliability is controlled by two parameters ε_{ns} for the tolerance of the gradient line position and ε_{nl} for the tolerance of the normal to the gradient line used to increase the reliability of the maximum gradient detection.

This whole analysis sequence corresponds to the *DimProj* program indicated in figure 11. The other programs on the same figure also require the same sort of decisions (parameter tuning, algorithm choice) to set the image protocol.

4.3. Knowledge Base Content

For our tests, the program supervision system for osteoporosis detection, has supervised programs written in MatLab.

The knowledge base (KB) is developed using a dedicated knowledge representation language, named YAKL [MOISAN, S., 2002], which uses both object-based and rule-oriented descriptions. This language allows experts to define domain types, objects or operators, and different rules to be used during reasoning.

For example, it defines the scheduling of the programs and describes the inputs and outputs of each one with its arguments and syntax. These descriptions are provided by the expert in medical image processing and are transparent to clinicians who have only to provide the digital radiographic images to be processed.

A KB can be presented by a tree where the root is the principal composite operator, the intermediate nodes are the other composite operators and the leaves are the primitive operators. Figure 11 shows a simplified tree of our KB about the osteoporosis detection programs. The entry point of this tree is represented by the root operator *OsteoMorph*. This composite operator is decomposed into a sequence of primitive operators (oval forms) and composite ones (rectangular forms).
The tree corresponding to our KB has six abstraction levels with 31 operators (11 composite operators and 20 primitive ones). Three of the eleven composite operators present a choice between operators, the other eight ones including one iterative operator, present sequences of operators. Among the primitive operators, there are:

- Three optional operators, i.e. their planning depends on the corresponding optionality criteria;
- Seven operators belonging to branches that require a choice, i.e. their planning depends on some choice criteria.
- Ten that are planned in all cases.

Adding to this, there are about ten decision criteria. For instance, here follows an example from the osteoporosis knowledge base: a composite operator that describes an alternative decomposition (denoted by a |) into two suboperators: *grey-level* or *binary skeletonization*.

```
Composite Operator { name skeletonization
        comment "skeletonization of gray image"
        Input Data
                Symbol name gray_image
        Output Data
                Symbol name skeleton_image
                Integer name nb_connect_parts
        Body GreySkel | BinSkel
Choice Rule { name grayChoice
        If ask_user "Do you need to compute the number
                of connected parts ?" ["yes" "no"] == "yes"
        Then use_operator GreySkel }
Distribution
        skeletonization.image_gray / GreySkel.gray_image
        skeletonization.image_gray / BinSkel.gray_image
... }
```

A choice rule has been given by the expert to decide which sub-operator should be selected depending on the situation (in this case the choice is left to the end-user). The *Distribution* part displays information about data transfer between the parent operator and its sub-operators.
In other examples, when a composite operator is decomposed into a sequence of operators, the *Flow* part is added and will contain data transfer between the different sub-operators.
The role of rules is essential to the engine strategy at different points during the reasoning process: for instance, to choose between several alternatives (choice rules, as shown above), to adapt program execution, to assess result quality, and to repair a badly assessed execution. These decision points are the key to establishing and adapting the “image protocols”.

5 Distributed and Collaborative Medical System

In our previous applications we used a centralized supervision system in which data, programs and KBS have the same location. However, applying program supervision to osteoporosis detection requires distributing the system. Indeed, both data (images from different hospitals) and programs (developed by different teams) come from various places. In most cases it would be fairly inefficient to move data and executable code to the same place as the KBS, and in some case it is even not possible (e.g., programs may execute only on a specific hardware). Therefore, we have developed a distributed version of the assistant, based on mobile agents, where:

- Clinicians from different countries may safely use the system and work collaboratively to diagnose diseases. This can be possible by running programs of other teams, checking the consistency of image analysis methods and evaluating results.
- Medical imaging experts may manage different versions of their programs,
- Knowledge engineers may capitalize knowledge and adapt it to program changes.

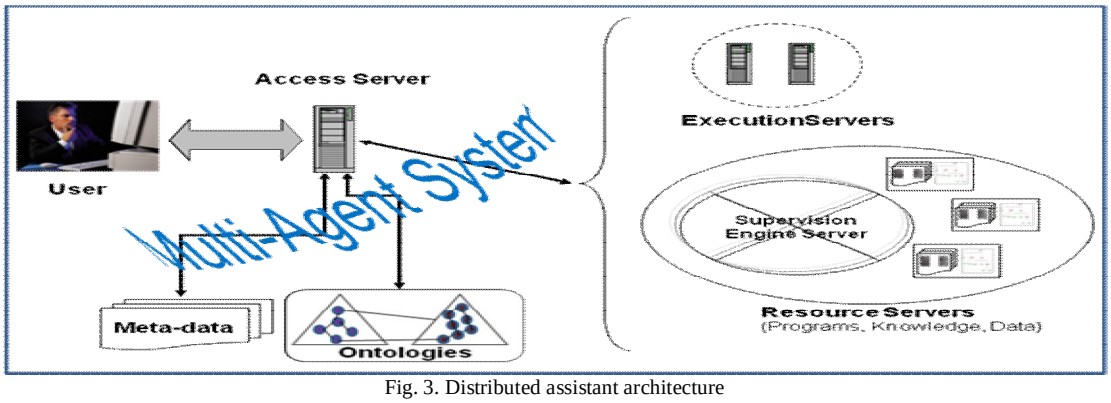

Fig. 3. Distributed assistant architecture

5.1. Distributed System Architecture

In our work, we developed a supervision server via the Web which allows the remote consultation and modification of knowledge bases and provides an authenticated access to different users. It manages their requests, repatriates data, delegates processing, recovers the results and returns them to the user.

The distributed system, whose architecture is proposed in [KHAYATI, N. *et al*. 2006] [KHAYATI, N. *et al*. 2007] and given by figure 3, is a triple (S, A, KM) defined by:

- S, a set of three types of servers :
  - A session server playing the role of an interface allowing end-users to access the supervision services and to communicate with the other components of the distributed system.
  - A set of resource servers hosting programs, knowledge and supervision engine of the centralized system: a supervision server hosting the supervision engine, some program servers, some knowledge servers and some data servers.
  - And possibly a set of execution servers, on which programs are executed. For example, when dealing with MatLab programs, their execution requires at least, the presence of the MatLab tool on one of the servers.
- A, a set of agents who are responsible for updating the previous components and for performing requests. This multi-agent system combines stationary and mobile agents.
- KM, a set of knowledge models in the form of metadata and ontologies used to locate resources in order to define the mobile agents itinerary, to define access permissions and to analyze the user request so that it is properly treated.

5.2. Agent Model

In this section, we will present the adopted multi-agent system in terms of its architecture and behavior of the involved agents.

5.2.1. Architecture of the Multi-Agent System

We first developed a Web server allowing remote users to access the supervision system. Second, we distributed the components of the supervision system itself, implementing the communications with mobile agents [KHAYATI, N. *et al.* 2013]. Meta-data help localize the various resources (programs, data, execution servers, etc.).
The architecture of the multi-agent system as presented on figure 4 consists of several types of servers. First, one *Program Supervision Server* runs the supervision engine PEGASE; then possibly several *Execution Servers* enable to execute the planned programs; *Resource Servers* contain remote resources needed by executions (e.g., data files, scripts); finally, the end-user interacts directly with an *Access Server.*

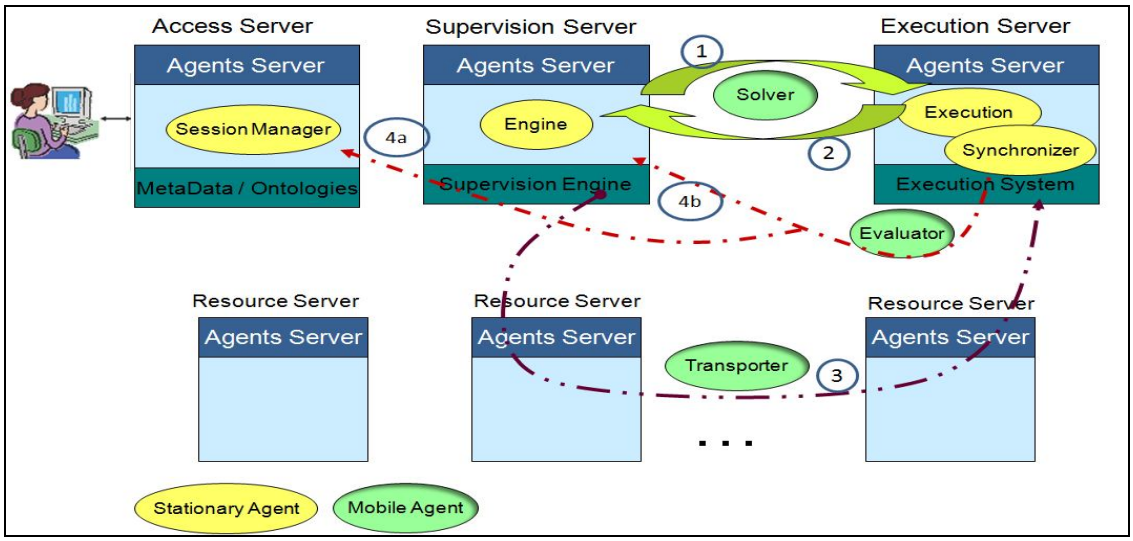

Fig. 4. Architecture of the multi-agent system and scenario of use

5.2.2. Agent Classes and Behaviors

The agents are classified according to their roles into three categories:

- *Interface agents* that are associated to particular servers to simplify communication with them (Supervisor agents, Engine agents and Execution agents).
- *Processing agents* directly involved in the different phases of the general process of a query resolution (Solver agents and Evaluator agents).
- *Communication agents* ensuring the interaction between servers and the transport of the required resources from one server to another (Transporter agents and Synchronizer agents).

a. Interface Agents

Supervisor Agents. They are called also *Session Managers*; they are stationary and are coupled with the access web server. Each one manages the whole user session and the whole process of solving the supervision query in its charge. This means that there is one Supervisor agent by query.
Such an agent has the following tasks:

- Reading the metadata to identify and localize the involved resources in the resolution of the query;
- Creating the other stationary agents for interfacing with the supervision engine (Engine agent) and the different execution servers (Execution agents);
- Determining the number of needed Solver agents and creating them. This computation is done according to some information given by the Supervision Engine: the $Ndep$ dependency sets and the $Npar$ parallel operators by set.

For the last point, in a YAKL code (*YAKL for Yet Another Knowledge Language, used by PEGASE*), we may note parallel tasks in the *Body* section of a composite operator. We may also note dependency links between the sub-operators of a composite one, in its *Flow* section. These concepts are necessary when determining the number of solver agents. Hence, we can determine these dependencies and the number of these parallel tasks. The dependencies will allow building some dependency sets that will serve to find the right number of solver agents to create.
We define a *dependency set* as a set of operators having at least one common parameter. The construction of these sets is done according to the algorithm given in figure 5. Let, for example, the following description of a composite operator.

Composite Operator
{
name *OP*
…
Body *op1 – op2 – op3 – op4 – op5 – op6*
…
Flow
op1.x / op2.x
op3.y / op4.y
op2.y / op3.y
op1.z / op4.z
op5.t / op6.t
}

Applying this algorithm, we obtain the two following dependency sets: *{op1, op2, op3, op4}* and *{op5, op6}*, and thus, we deduce that the supervision process will need only two solver agents, one for each set.

– Let $LeftOp_k$ and $RightOp_k$, respectively the left operator and the right operator of the k^{th} rule of the Flow Section.
– Let n the number of sub-operators of a composite one.
– Let m the number of rules for this operator.

- Create $E_1, E_2, ..., E_n$: n dependency sets (each one is a singleton containing one of the n operators).

- For k in 1 to m do
  If $LeftOp_k \in E_i$ and $RightOp_k \in E_j$
  Then $E_i := E_i \cup E_j$
  Delete E_j
  End if

Fig. 5. Algorithm of the Dependency Sets Construction.

Engine Agents. There is an Engine Agent by query; it represents an "instance" of the supervision engine and is responsible for processing a query. The role of such an agent is:

- Interfacing with the supervision engine, submitting a query to process and getting
  a plan of programs to execute.
- Communicating the parallel treatments and the dependency sets to the Supervisor agent.

Execution Agents. Once sent to their running servers, they stay there. Their role is to interface with these servers by passing the instructions to execute (received from a Solver agent) and retrieving the obtained results.

b. Processing Agents

Solver Agents. They are mobile and have to launch the execution of the remote programs already planned by the supervision engine and received from the Supervisor agent.
For their migration from one machine to another, different policies can be considered:

- Keep the programs where they are, and send data to program sites, launch the execution and gather results to transmit them to the end-user.
- Take the programs (when they are lighter than data) and execute them on the data sites.
- Move programs and data to an execution server, execute the programs on the data and collect the results for their transmission to the user.

For example, for the first case, during its migration, a Solver agent takes with him the data and all necessary parameters for the execution of the planned program, executes it and saves, in its context, the result and the execution parameters for the next programs.
For the third policy, while being under the control of the Supervisor agent, the Solver agent migrates to the supervision server (path 1 on figure 4) looking for the next instruction (the program and its call syntax). Then it seeks the necessary resources for this instruction from the Transporter agents, migrates to the corresponding execution server (path 2 on figure 4), and then waits until all the resources are available (information received from a synchronizer agent). At this stage, it starts executing the instruction, and finally, it sends the results to the Supervisor.
In general, the number of solver agents is determined according to the following rules:

- For a dependency set of *N* elements, if it contains *Npar* parallel elements (*Npar* ≤ *N*), then the maximum number of solver agents will be equal to *Npar* (one agent per parallel task, then one of them will continue with the other tasks of the same set). Otherwise it will be equal to 1.

- For *Ndep* dependency sets the maximum number of solver agents will be equal to $\sum_{i=1}^{Ndep} Npar\_i$. Obviously, if no set has parallel tasks, such number shall be equal to *Ndep*.

Evaluator Agents. They are created by Solver agents on the program sites (for the first two policies) or on the execution sites (for the third policy). They are created only if they are needed, i.e. if some programs require the evaluation of their results.
An agent of this class stores the result to evaluate in its context and then must go to another server to perform its task. For its migration, the destination depends on the assessment type. Thus, if the evaluation is automatic, i.e. made by the supervision engine based on the knowledge base rules, it must migrate to the supervision server (path 4b on figure 4). Otherwise, if the assessment is interactive, i.e. it requires the user intervention; it migrates to the access web server (path 4a on figure 4). In the case of an interactive assessment, the user response will be sent to the engine agent so it can decide the next step (continue with a new program or re-run the same program with repaired values for its parameters).

c. Communication Agents

Transporter Agents. They are created by the solver and are responsible for:

- Searching the necessary knowledge for the supervision engine in order to select the programs and their sequencing.
- Searching the necessary resources to execute the current instruction, and transporting them to an execution server (path 3 on figure 4).

Since an instruction may need resources located on different nodes, there may be, simultaneously, several active Transporter agents.

Synchronizer Agents. They are created by the Solver agent to synchronize the operations of the resources transport performed by the Transporter agents. Indeed, since many Transporters may be active simultaneously, they must register with the associated Synchronizer. When they are all registered, this agent starts them and waits until they do their jobs. Finally, it indicates to the Solver that it can continue its work (all the needed resources are available).

5.3. Security Model

Since we are dealing with medical applications, confidentiality of the patients' information and diagnosis results is of the highest importance. To this end, we have complemented our system with additional information related to this kind of security and confidentiality. Hence, we proposed a three-level security model, as shown in figure 6, which classifies the security problems in three levels:

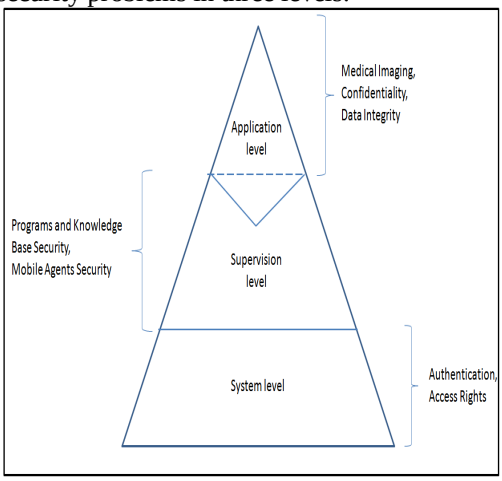

Fig. 6. A three-level security model

- A "*system level*" that must be considered before any system get started. It concerns, the access rights assigned to each user in order to protect of all the components of our distributed supervision system.
- A "*distributed supervision level*" which concerns the use of mobile agents.
- An "*application level*" which concerns the medical domain and especially medical imaging.

The last two levels are intertwined. They shared a few common points in terms of used cryptography techniques to ensure the data confidentiality and integrity when traveling on the network.

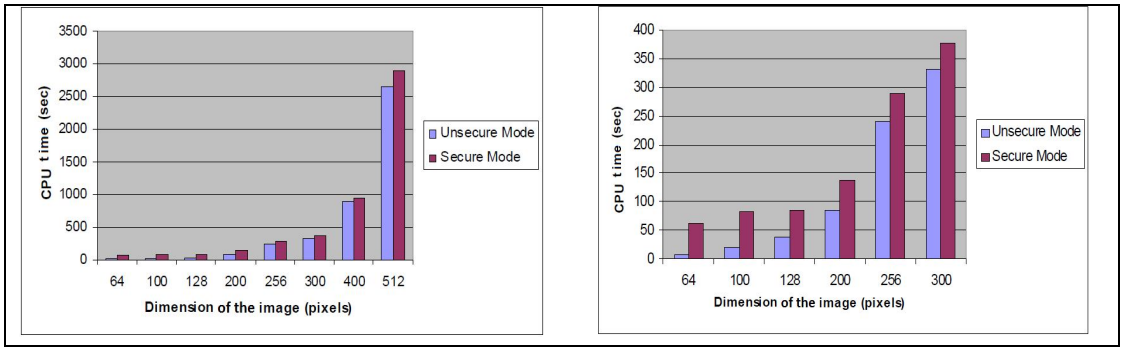

Fig. 7. The average response time in secure mode vs. the average response time in unsecure mode.
(the right part of the figure is a zoom-in of the first six measures given in the left part of the figure)

5.3.1. Security for Agents

Agents can exchange messages containing confidential data, or may migrate across the network by transporting confidential results. Thus, the most important issue, for physicians, is how their confidential data and confidential results circulate on the Net without being caught. The supervision level gives them the answer, it constitutes the security kernel.
In fact, to ensure a secure communication, we opted for RSA cryptography techniques to encrypt the arguments of the message in the source and to decrypt them into the destination.
Concerning the secure migration of agents, the aim is to secure both the agent as its content. That's why we opted for a solution which serializes the agent, encrypts the corresponding file, and dispatches the encrypted file inside a transporter agent. At the destination, the transported file is deserialized then decrypted in order to reactivate the agent.

5.3.2. Performance Evaluation

The response time of the system is a factor which is as important as security. For this, we tested our distributed system in two modes: in secure mode and in unsecure mode. The tests concern the osteoporosis detection on bone images of different sizes (from 64 * 64 to 512 * 512). In these tests, the supervision engine has selected 5 operators to run (figure 11). The execution time depends on several aspects: the image size, the number of programs to run, the chosen mode (secure or unsecure) and obviously the bandwidth of the network on which we conduct our tests.
Figure 7 shows that, for all image sizes, the secure mode takes more time than the unsecure one. Figure 8 shows also that for a small image, the secure mode takes more than 6 times the time taken by the unsecure mode (about 1 minute against 10 seconds). In addition, this factor does not remain the same as for larger image, it gradually decreases until it converges to 1 (approximately 50 minutes against about 45 minutes).

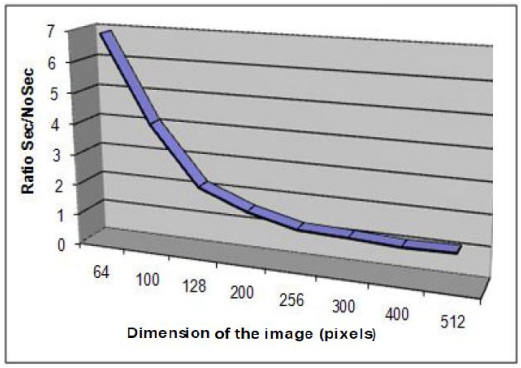

Fig. 8. Evaluation of the ratio secure mode / unsecure mode.

5.4. Knowledge Model

The distributed supervision system is equipped with ontologies in order to define access permissions, to analyze the user request so that it is properly treated, to locate the required resources, to define the mobile agents' itinerary, etc. The current section shows the ontological architecture and how these ontologies are integrated when answering a supervision request.

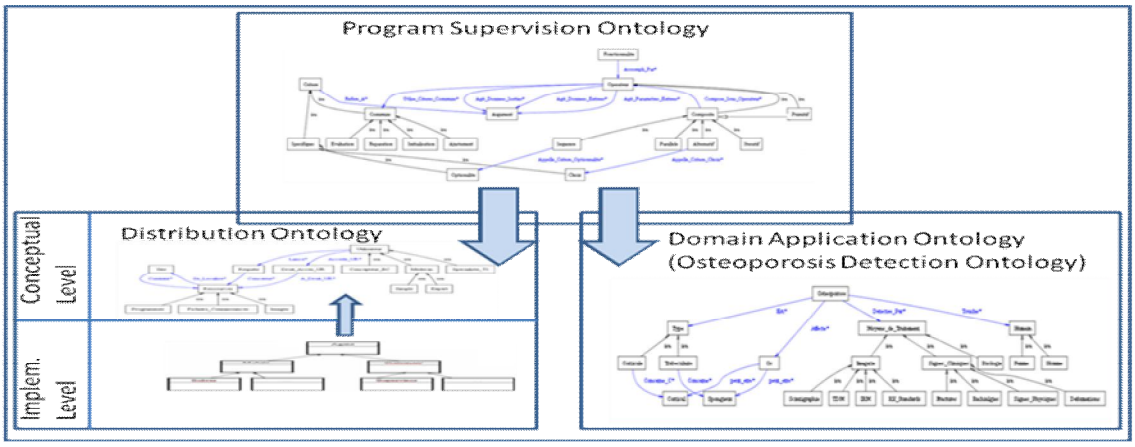

Fig. 9. Ontological Architecture

5.4.1. Ontological Architecture

The need for ontologies is expressed in terms of:

- Sharing a common understanding of our application domains (program supervision, mobile agents, medical imaging, osteoporosis detection, etc.). This guarantees that all our system modules use a term with the same meaning specified in the ontology and facilitates the integration of these modules and their interoperability.
- Solving problems related to the semantic of the supervision queries.

For this, the overall ontological architecture (figure 9) of our distributed system consists of three ontologies integrated and interoperable: an ontology for the supervision domain and its knowledge, a second for the elements involved in its distribution (distribution ontology) and a third for the application domain (medical imaging and osteoporosis detection).

For their integration into the system, our ontologies undergo the application of a reasoner which offers, in case of inconsistency, possible repair operations. Thus, we obtain semantic files reflecting them (Step 1 on figure 10).

5.4.2. Asking the Ontologies

The query language defines the syntax and the semantics required to express queries and the possible forms of the results. We expressed interrogative SELECT queries type, which extract a sub-graph corresponding to a set of resources satisfying the conditions specified in the WHERE clause. Indeed, the user expresses his query in textual form, for example, "check the status of the bone". This sentence will be split into words; neutral words will be eliminated (the, of, etc.). Then, the application domain ontology will be queried. This gives that the word "check status bone" is subsumed by the "Osteoporosis" concept. For testing, we added white concepts such as "diabetes", "Hepatitis", etc. to ensure that the query asked to the ontology receives the good answer. Then, the supervision ontology has to fetch the functionality which is responsible for the osteoporosis detection and subsequently the appropriate composite operator. Finally, the program supervision process can start.

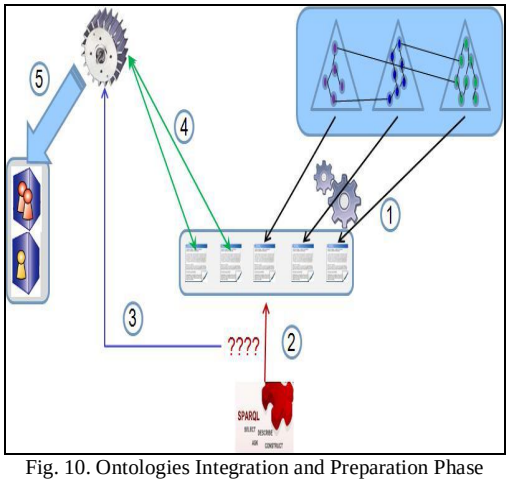

Fig. 10. Ontologies Integration and Preparation Phase

6 Medical Scenario

This section illustrates a rheumatologist processing an image through our distributed medical assistant. After connecting to the supervision server, the rheumatologist simply enters his query as keywords and uploads an image to analyze.

6.1. Preparation Phase

On receiving a supervision query from a user, our system will translate it in a query language for the ontological architecture in order to determine the appropriate functionality (Step 2 on figure 10, cf. §5.4.2). Once determined, this functionality will be communicated to the supervision engine (Step 3 on figure 10) so it can decide the "good" knowledge files (Step 4 on figure 10) and thereafter, the right resources needed for the resolution of the current query. Let, for example, the selected knowledge files deal with a functionality having as a first composite operator, *OsteoMorph*. Then, the knowledge-based system launches this operator and starts a planning phase by decomposing this operator into its suboperators, as shown in figure 11: *Reading, Skeletonization* and *Analysis*.
Reading, (1) in figure 11, is a primitive operator and does not need evaluation. The second sub-operator (*Skeletonization)* is a composite one, the system has to choose between two alternative sub-operators: *BinSkel* or Gray*Skel* (i.e. binary or gray-level skeletonization). Using expert choice rules given in the knowledge base, the engine selects one of them, say *BinSkel* (2). Since the knowledge base requires an evaluation of result for this operator, planning must be suspended and execution performed.
At this stage, agents will be launched (Step 5 on figure 10) to start a supervision process.

6.2. Supervision Phase

The execution starts with the Solver agent moving to the *Reading* program site and running it. Then the Solver goes to the *BinSkel* program site to execute it. Now, the evaluation phase can be performed. An Evaluator agent is created to execute evaluation rules. In case of automatic evaluation, it moves to the engine site to execute them. In case of user evaluation, it moves to the server site in order to ask the questions provided by these rules to the user. Then the Evaluator sends the assessment back to the engine. If the assessment is good, the planning phase continues with the *Analysis* composite operator. Otherwise, repair or adjustment rules are used to decide to modify the plan or to re-execute the same programs with different parameter values, resulting in a new plan. For instance, *BinSkel* assessment is manual. The user is asked about the presence of undesirable segments. If this is the case an adjustment rule is fired, that increases the value of a pruning parameter, so that *BinSkel* may be executed again.
The same process runs up to the last program in the decomposition (5). The Solver agent executes this last operator and moves back to the server with the final result together with the plan established by the engine, for example, the following sequence of programs: *Reading*, *BinSkel*, *DimProj*, *Direction* and *AttributesDir*.
The system may also have other uses depending on the end-user type. For instance, image processing experts can use the supervision server to check their programs, observe their results, and compare them with other approaches. A collaborative construction of knowledge bases on the use of medical image processing programs is also possible.

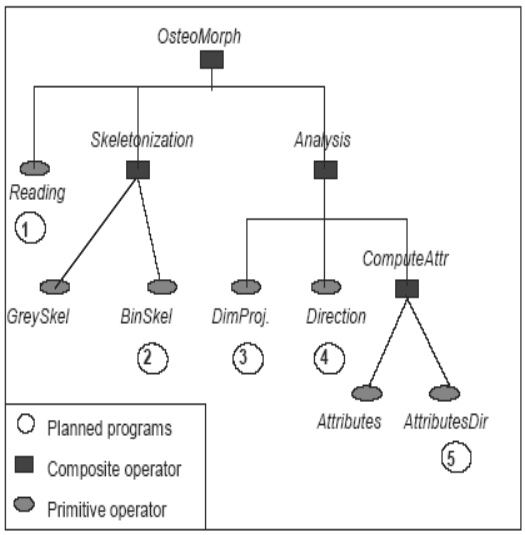

Fig. 11. The *OsteoMorph* operator decomposition.

7 Conclusion

Considering the early osteoporosis detection as a challenge for the medical community, we described in this paper, a distributed intelligent and interactive system relying on knowledge based techniques: (1) to assist physicians and image experts in validating "image protocols"; (2) to facilitate access to up to date image programs by rheumatologists who do not want to bother about medical image analysis details; (3) to allow physicians to submit an image, to obtain resulting medical parameters and also the corresponding execution plan (that is the effective "image protocol"). Therefore, the radiologists who are not specialists in image processing must be freed from the program details in order to focus on the interpretation of the results and their evaluation.

Distributing such systems is fundamental because physicians, image processing programs, images, inference engine, knowledge bases, etc. are generally located at different sites. Our distributed environment is based on a Web server, mobile agents for the communication inter-components and Semantic Web ontologies to facilitate physician access and knowledge exploration.

Hence, the strength of the proposed system comes from the following points:

- Taking advantages from the technological advances in medical image processing.
- Facilitating the experiments by radiologist and physicians and allowing them to adjust and repair the values of the programs parameters.
- Making different experts work together.
- Consolidating the diagnosis of the disease.
- Taking into account the heterogeneity of data (images, knowledge, ontologies, programs, etc.).

As prospects, we plan to improve the performance of our distributed and collaborative intelligent system when scaling with multiple queries, larger and multiple images, etc. and this, in order to test its ability to maintain its functionalities and performance in important demand.

8 Acknowledgment

We kindly thank:

- S. Moisan (ORION project - INRIA Sophia Antipolis) and J-P Rigault (Nice University) for helping in the first part of the distributed system development.
- S. Sevestre-Ghalila (U2S Research team – Tunis El Manar University) for providing osteoporosis detection programs.
- C. L. Benhamou and C. Chappard (CHR Orléans, France) for their interest in our work and for providing radiographic images.
- SOIE Laboratory members for helping in the development of the second part of the distributed system (security, ontologies, etc.).

9 References

[BENHAMOU, C.L. 1994] Benhamou C.L., Harba R., Lespessailles E., Jacquet E., Toulière D. and R. Jennane. *Fractal organization of trabecular bone images on calcaneus radiographs*, J. Bone Mineral Research, 9:1909-1918, 1994.

[BOUHLEL, N. 2009] Bouhlel N., Hajjaji S., Sevestre S. and Laugier P., *Texture analysis using Nakagami-MRF model: Preliminary results on ultrasound images of primary choroidal melanomas*. In Proc. of the 16th IEEE International Conference on Image Processing (ICIP 2009), 4181-4184, Cairo, Nov. 2009

[CRUBEZY, M. *et al*. 1997] Crubézy M., Aubry F., Moisan S., Chameroy V., Thonnat M. and Di Paola, R., *Managing complex processing of medical image sequences by program supervision techniques*, In Proc. of SPIE Medical Imaging 1997, vol. 3035-85, pp. 614-625, Newport Beach, CA, February 1997

[KHAYATI, N. *et al*. 2006] Khayati N., Lejouad-Chaari W., Moisan S. and Rigault J.P., *Distributing Knowledge-Based Systems Using Mobile Agents*, WSEAS Transactions on Computers, Issue1, Volume 5, pp 22-29; January 2006.

[KHAYATI, N. *et al*. 2007] Khayati N, Lejouad-Chaari W, Moisan S. and Rigault J.P, *Agent Model for Distributed Program Supervision Systems*, The Fifth European Workshop on Multi-Agent Systems - EUMAS'2007, pp 155-164, Hammamet, Tunisia, December 2007.

[KHAYATI, N. *et al*. 2008a] Khayati N., Lejouad-Chaari W., Sevestre-Ghalila S., *A Distributed Interactive Medical Diagnosis Support System*, In Proceedings of the 2nd International Conference on Advanced Information and Telemedecine Technologies for Health (AITTH'2008), pp 59-63, Minsk, Belarus, October 2008.

[KHAYATI, N. *et al*. 2008b] Khayati N., Lejouad-Chaari W., Sevestre-Ghalila S., *A Distributed Image Processing Support System: Application to Medical Imaging*, In Proceedings of the IEEE International Workshop on Imaging Systems and Techniques (IEEE-IST'2008), pp 261-264, Chania, Greece, September 2008.

[KHAYATI, N. *et al*. 2013] Khayati N. and Lejouad-Chaari W., *Agent and Knowledge Models for a Distributed Imaging System*, 10th International Symposium on Distributed Computing and Artificial Intelligence (DCAI'2013), Salamanca, 22-24 mai 2013. Advances in Soft Computing, Springer 2013.

[LEJOUAD-CHAARI, W. *et al*. 2007] Lejouad-Chaari W, Moisan S, Sevestre-Ghalila S, Rigaut J.P, *Distributed Intelligent Medical Assistant for Osteoporosis Detection*, In Proc. of the International Conference of IEEE Engineering in Medicine and Biology Society, Lyon – France, August 2007.

[MERSAL, S.S. *et al*. 1999] Mersal S.S. and Darwish A.M, *A new parallel thinning algorithm for gray scale images*, IEEE Nonlinear Signal and Image Proc. Conf., Antalya, Turkey, June 1999.

[MOISAN, S., 2002] Moisan S, *Knowledge Representation for Program Reuse*, ECAI'02, Lyon. France, Jul. 2002.

[PAQUET, V. *et al*. 1995] Paquet V., Battut P., Blanc H.V. et Ferrand D. *On the use of gray run lenth matrices in trabecular bone analysis*. In Proc. of the Conf. on the Image Processing and its Application, pages 445-449, July 1995.

[SEVESTRE-GHALILA, S. *et al*. 2001] Sevestre-Ghalila S., Benazza-Benyahia A., Cherif H. et Souid W., *Texture analysis for osteoporosis detection with morphological tools*, Medical Imaging 2001, SPIE Conf., Milan Sonka, Kenneth M. Hanson Eds., Vol. 4322, pp. 1534-1541, San Diego, California, USA, 17-23 February 2001.

[SEVESTRE-GHALILA, S. *et al*. 2004] Sevestre-Ghalila S., Benazza A., Ricordeau A., Mellouli N., Chappard C. et Benhamou C.L., *Texture image analysis for osteoporosis detection with morphological tools*. In P. Hellier C.Barillot, DR Haynor, editor, MICCAI 2004, volume 1, pages 87-94. LNCS Springer Verlag, September 2004.

[SHEKHAR, C. *et al*. 1995] Shekhar C., Moisan S. and Thonnat M., *Real-Time Perception Program supervision for Vehicle Driving Assistance*, In Okyay Kaynak, Mehmed Ozkan, Nurdan Bekiroglu, and Ilker Tunay, editors, ICRAM'95 Intl. Conference on Recent Advances in Mechatronics, pp. 173–179, Istanbul

[THONNAT, M. *et al*. 1999] Thonnat M, Moisan S. and Crubézy M, *Experience in Integrating Image Processing Programs*, Int. Conf. Vision Systems (ICVS'99). Las Palmas, Spain. January 1999. LNCS 1542, pp 200-215.

[THONNAT, M. *et al*. 2000] Thonnat M. and Moisan S, *What can Program Supervision do for Software Reuse?*, IEEE Proc. Special Issue on Knowledge Modelling for Software Components Reuse, Vol. 5, No. 147, pp.179- 185, Oct. 2000.

[VINCENT, R. *et al*. 1997] Vincent R., Thonnat M. and Ossola J.C., *Program supervision for automatic galaxy classification*, In Proc. of the Intl. Conference on Imaging Science, Systems, and Technology, CISST'97, June 1997.

[WIGDEROWITZ, C.A. *et al*. 1997] Wigderowitz C.A., Abdel E.W. et Rowley D.L. *Evaluation of cancellous structure in the distal radius using spectral analysis*. Clin. Orthop., 335:152-161, 1997.