Borg : A knowledge-based system for automatic generation of image processing programs

Régis Clouard, Abderrahim Elmoataz, Christine Porquet, Marinette Revenu

References

- **Title:** Borg : A knowledge-based system for automatic generation of image processing programs
- **Authors:** Régis Clouard, Abderrahim Elmoataz, Christine Porquet and Marinette Revenu
- **Affiliation:** GREYC - ISMRA, 6 boulevard Maréchal Juin, F-14050 Caen cedex, FRANCE

- **Abstract:** This article deals with the design of a system that automates the generation of image processing applications. Users describe tasks to perform on images and the system constructs a specific plan, which, after being executed, should yield the desired results. Our approach of this problem belongs to the more general category of systems for the supervision of a library of operators. The generation of an application is here considered as the dynamic building of chains of image processing through the selection, parameter tuning and scheduling of existing operators. To develop such a system, we suggest to use a knowledge-rich resolution model and to integrate seven design rules. The BORG system has been developed following these prescriptions. It hinges on hierarchical, opportunistic and incremental planning by means of knowledge sources of the Blackboard model, which enable to take into account planning, evaluation and knowledge acquisition issues.

- **Keywords:** Image Processing, Artificial Intelligence, Knowledge-Based Systems, Composition of Image Processing Programs, Supervision of a library of operators.

- **Contact Author:** Régis Clouard
- **Contact Address:** GREYC - ISMRA, 6 boulevard Maréchal Juin, F-14050 Caen cedex, FRANCE
- **Contact Email:** Regis.Clouard@greyc.ismra.fr
- **Contact Tel:** +33-(0)2-31-45-27-21
- **Contact Fax:** +33-(0)2-31-45-26-98

Borg: A Knowledge-Based System for Automatic Generation of Image Processing Programs

Abstract: This article deals with the design of a system that automates the generation of image processing applications. Users describe tasks to perform on images and the system constructs a specific plan, which, after being executed, should yield the desired results. Our approach of this problem belongs to the more general category of systems for the supervision of a library of operators. The generation of an application is here considered as the dynamic building of chains of image processing through the selection, parameter tuning and scheduling of existing operators. To develop such a system, we suggest to use a knowledge-rich resolution model and to integrate seven design rules. The BORG system has been developed following these prescriptions. It hinges on hierarchical, opportunistic and incremental planning by means of knowledge sources of the Blackboard model, which enable to take into account planning, evaluation and knowledge acquisition issues.

Index Terms: Image Processing, Artificial Intelligence, Knowledge-Based Systems, Composition of Image Processing Programs, Supervision of a Library of Programs.

1. INTRODUCTION

The design of knowledge-based systems exclusively dedicated to image processing continues to present a major challenge [11]. The role of such systems is to automate the performing of image processing tasks restricted to image transformations without any interpretation of the image content. Tasks refer to general image processing objectives such as segmentation, restoration or enhancement of 2D or 3D images. Even if the number of tasks recognized by the system is quite limited, these tasks must operate robustly in widely varying contexts. The design of such systems is an important issue because it opens perspectives in two major trends:

- To make image processing accessible to end-users, by the way of programming environments that can cover a wide range of tasks and contexts, while limiting cognitive and skill requirements of users ;
- To improve performances of vision systems and interpretation systems. Draper [13] points out that the weakness of the image processing level is one reason for explaining the failure of vision systems.

This being a relatively recent concern (first research on this subject dates back to years 1985), together with relatively scarce results (very few commercialized systems, except Toriu's system [31]) can account for the difficulty of the undertaking.

Indeed, image processing is a domain where there exist no general-purpose methods that can be described by algorithms applicable to a wide variety of tasks and contexts [25]. Processing an image is a complex process, from which a human expertise is gradually emerging.

Therefrom, Artificial Intelligence methods and tools can advantageously be introduced to automate such a process. They allow an explicit representation of the problem to be solved as well as an operative modeling of human expertise, thereby providing self-configuration capabilities to adapt the system behavior in accordance with the problem specifications.

However, human expertise is difficult to exhibit, because it is non-structured and relies on different theoretical reference domains (e.g., signal processing, mathematical morphology) dealing with competencies belonging to several levels of abstraction (e.g., numerical, symbolic). That may explain why several different knowledge-based modeling approaches can be found in the literature:

- *Competition of competencies* (e.g. [6]) exploits concurrency between several predefined processing strategies executed in parallel. The result is composed by a fusion of regions from different images obtained by these different strategies, each region being assessed as the best one according to contextual criteria. In such an approach, the reasoning focuses on a posteriori analysis of fusion criteria thanks to the use of a region classifier and a strong modeling of the application objects.
- *Incremental construction of the result* (e.g. [3], [23]) proceeds by progressive and controlled evolution of input data toward desired output data. Image processing algorithms are completely split into a set of production rules or rational agents. In such an approach, there is no explicit analysis strategy; the reasoning keeps on focusing on a priori analysis of the current state of data, in order to decide on the next process to apply, in the case of rules, or to solve conflicting access to data, in the case of agents.
- *Case-based reasoning* (e.g. [5]) relies on retrieval and adaptation of existing processing strategies constructed by users in order to solve new problems. The reasoning focuses

on problem features to retrieve a past case assessed to be similar.

- *Operator supervision* (e.g. [30]) consists in dynamically building a convenient processing strategy by selecting and scheduling existing operators from a given library. The reasoning focuses on problem features to select adequate operators, to tune their parameters and to schedule them.

None of these approaches can be rated as the best of all; in fact, each of them addresses particular issues:

- *Competition of competencies* mainly emphasizes the operative aspect of knowledge and takes advantage of computer performances. However, explicitness of knowledge is not really taken into account. For instance, it is difficult to revise fusion criteria when adding new strategies.
- *Incremental construction of the result* is aiming at minimizing the introduction of human expertise into the system; as such, it avoids knowledge acquisition issues. However, execution time, global handling of errors and global control of the result consistency are the main defaults of this approach.
- *Case-based reasoning* is concerned with making knowledge acquisition easier. New cases can be easily introduced into the case base because there is no need to explain the underlying reasoning. Only information necessary for reusing is modeled, such as the problem specifications. However, as the number of cases increases, retrieval and adaptation of cases become more and more difficult and questions such as how to generalize cases to avoid accumulation of too specific cases must be tackled.
- *Operator supervision* tries to combine the operative and explicit aspects of knowledge and to offer good solutions to the increase of the knowledge base and of the operator library. However, knowledge acquisition implies a complete elicitation and modeling of human expertise.

This paper presents BORG, a system belonging to the operator supervision approach because we are both aiming at making knowledge operative, as well as making it explicit. We want to consider our system as an experimentation tool enabling users or interpretation systems to build their own image processing programs, and also as a knowledge acquisition tool

providing experts with a framework to express, criticize and exchange their knowledge.

The specifications of our system follow the conclusions of a study, presented Section 2, on the issue of operator supervision in the specific context of image processing. In Section 3, our own motivations and objectives are detailed and the general architecture of the system, based on the Blackboard model, is described. In Section 4, the conceptual model of BORG is given and illustrated through the example of a biomedical application dealing with cytological images. In conclusion, the major contributions of our work are discussed.

2. THE SUPERVISION OF IMAGE PROCESSING OPERATORS

In this section, we put forward the specifications of a knowledge-based system model consistent with the operator supervision approach in the specific field of image processing. After analyzing the problem situation, a problem-solving model is proposed. From a survey of different relevant systems, essential features of this problem-solving model are then exhibited, and enacted as design rules.

2.1. Problem situation

The prerequisite for operator supervision is a library of operators *a la* Khoros [26]. Operators code image-processing algorithms as operating-system commands that operate on image files. Using such operators imposes to set a numerical value for each of their parameters and to provide a list of image files for each of their inputs and outputs. If necessary, the return-code of operators can be used to store a value or a vector of values.

Operator supervision consists in composing and controlling the execution of a chain of operators in order to perform a given task. It is assumed that any processing strategy can be split into a sequence of individualized operators and that data flows can be restricted to normalized image files and parameter values. For instance, Fig. 1 shows a chain of operators intended to perform region segmentation based on image binarization with a threshold value that maximizes image contrast.

The execution of such a chain of operators often implies dynamic adjustment of parameter values and dynamic control of operator iterations in order to optimize results for each image.

According to its above-mentioned definition, the operator supervision issue encompasses the following activities [30]:

1. Selection of adequate operators ;
2. Determination of their parameters ;
3. Scheduling of operators to compose a chain ;
4. Generation of the corresponding executable script ;
5. Execution of this script ;
6. Control of results.

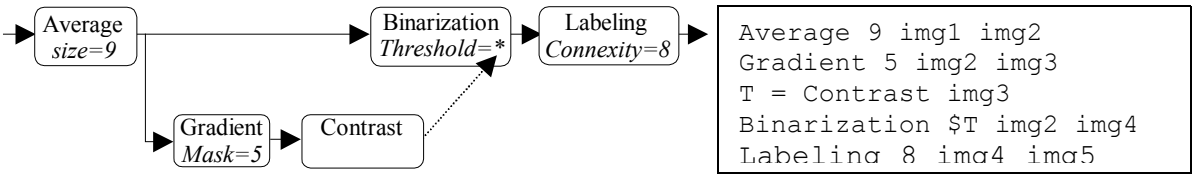

Fig. 1. Image binarization based on the computation of a threshold value that maximizes image contrast. On the left is shown the chain of operators and on the right the corresponding sequence of executable commands.

2.2. Problem-solving model

The automatic composition of chains of operators can be considered as being a complex problem in the context of systemics, because "the whole (the chain) is more than the sum of its parts (the operators)" [28], in the sense that, given the properties of operators and the laws of their interaction, it is not a trivial matter to infer the properties of the whole chain.

Of course, the problem is formulated in problem-solving terms, i.e., the building of a sequence of operators able to transform a given initial state (input image) into a final desired state (output image). However, well-known difficulties encountered during the development of image processing applications [21], [13] -mainly due to the ambiguous nature of images, the instability of operators and the empirical flavor of analysis methods- restrict the scope of potential problem-solving models. To manage complexity, two solutions can be used [24]: knowledge-rich problem-solving and hierarchy.

2.2.1. Knowledge-rich problem-solving

Knowledge-rich problem-solving relies on the existence of an expertise modeled in the knowledge base. An expertise describes an abstract plan adapted to a given task. It includes knowledge about conditions of applicability and knowledge about behavioral adjustment to particularities of the context. In the paradigm of knowledge-rich problem-solving, the composition of chains of operators results from a recognition process: first recognize the

relevant plan and then adjust its behavior.

By contrast, knowledge-poor problem-solving, such as combinatorial search or means-ends analysis, relies on the modeling of a type of expression that can be propagated along the chains of operators. Each operator or group of operators is modeled by a list of preconditions and a list of effects on the expression. The composition of chains of operators generally results from a generate-and-test process: first generating the possible future expressions (operators or groups of operators), then evaluating them. Evaluation is based on the analysis of the resulting expression in order to directly recognize it as being the final expression in the case of a combinatorial model or to choose the action that minimizes the distance to the final desired expression in the case of means-ends analysis. In the field of image processing, this paradigm raises three major issues:

1. *The definition of an expression modeling*. Expressions must be both simple to be automatically propagated and complex to take into account the intrinsic complexity, the variability and the abstraction of image data. An image in itself is far too complex and a description of image data in terms of image features is far too insufficient to accurately define such an expression.
2. *The a priori estimation of operator effects*. Effects of operators and groups of operators are heavily dependent on many factors and are difficult to forecast. Therefore, it is no easy matter to estimate expression changes.
3. *The tuning of sensitive parameters*. Knowledge-poor problem-solving does not take into account the problem of parameter tuning, were it only to estimate a plausible interval of values for these parameters.

Despite these limitations, knowledge-poor systems can be used to solve simple objectives, that can be formulated as a direct relation between input and output data, and that directly contain in their expression the way to compute the parameter values of operators. The decomposition of the initial problem into a sequence of elementary objectives can either be left to the user (e.g., EXTI [12]), or to a knowledge-rich system (e.g., MVP [7], LLVE [21]).

The knowledge-rich paradigm avoids previous issues, because, at each step of the resolution, the recognition process does not search what the next state of the solution should be, it does

know it [24]. Consequently:

1. A description of image data in terms of image features is generally sufficient for the recognition process.
2. The recognition process is not based on the analysis of the propagation of the image transformations.
3. Strategies for tuning parameters are directly included within plans.

2.2.2. Hierarchy

In order to manage complexity, image-processing problems are often described in terms of decomposition into sub-problems. Hierarchy can be organized along functional decomposition or abstraction decomposition of the task to perform. In the knowledge-rich paradigm, hierarchy takes the form of hierarchical plans. All the available expertise is coded into plans associating along a hierarchical decomposition the task to perform to a relevant sequence of operators.

Furthermore, in image processing hierarchy is natural. The processing of an image is not a monotonic process, for which each operator makes intermediate data progress towards the solution in a continuous and uniform way. One generally has to consider several changes of data abstraction (e.g., pixel into boundary, region into adjacency graph) and also resort to intermediate steps which are not directly related to the final goal (e.g., contrast enhancement, noise reduction, pre-segmentation).

2.2.3. Problem-solving definition

From the present study, one may conclude that the composition of chains of operators to perform a complex task can be considered as a planning problem. One thus has to go beyond the *symbol-level* to implement reasoning based on several hierarchical levels. A recognition process associated with knowledge rich problem-solving, consists of five steps:

1. *Planning*- Determine a suitable plan of actions ;
2. *Instantiation*- Adapt the plan to the particularities of the context and produce the corresponding chain of operators ;
3. *Execution*- Control the execution of each operator as well as the overall chain of operators. This step can be separated from the planning one, because the initial image

defines a static environment (it does not evolve in the course of time) ;

4. *Evaluation*- Assess the relevance of intermediate results ;
5. *Correction*- Propose corrections for unsatisfactory parts of the plan.

2.3. Design rules for a knowledge-based system for operator supervision

Research related to operator supervision in image processing can be divided into three major concerns, depending on what is required from users:

1. Some researchers give importance to the construction of image processing chains, in order to provide some assistance to users in the building and control of these chains (e.g., EXPLAIN [29], Carel's system [4]).
2. Others try to partially or totally automate the resolution process and only the control of execution is left to the user's responsibility (e.g., DIA-ES [27], Toriu's system [31], EXTI [12], MVP [7], COLLAGE[18]).
3. Projects with broader ambitions deal with the specifications of systems that are in charge of the complete building process. It implies the composition of chains of operators in accordance to the user objectives, the control of their execution, the tuning of parameters, the evaluation of intermediate results and the correction of possible errors (e.g., CONNY [19], OCAPI [8], VSDE [2]).

From a study of systems of the third category, seven general design rules have been derived, that should provide some inspiration and guideline for the design of a knowledge-based system for the supervision of image processing operators. Systems belonging to the first and second categories are not integrated in our study because they do not take into account the execution phase, and consequently, do not share the same constraints. For instance, alternatives or dynamic adjustments are not integrated in their knowledge modeling.

2.3.1. A request is composed of a list of the tasks to perform, the description of the data to process and the image data.

The tasks to perform define the user's intentions. Tasks refer to specific image processing objectives at some abstraction level (e.g., *Segmentation* in VSDE or *Coarse-Objects-Extraction* in OCAPI). They are generally itemized in a predefined list of tasks that constitute entries into the system. Often, constraints are associated to tasks in order to restrict their

scope. Three types of constraints are taken into account [10]:

- *Levels of detail* fix higher and lower limits on features of the desired results (e.g., geometrical tolerances on region features in VSDE) ;
- *Criteria to be optimized* put the focus of attention of tasks onto result features to be privileged (e.g., precise localization of the region boundaries in LLVE) ;
- *Restrictions* define boundaries of the task coverage (e.g., regions must be greater than 200 pixels in OCAPI).

The description of the data to process defines the context of application. It gives a meaning to the input image, by telling which data are relevant with regards to the tasks and which are not. It is represented as a list of symbolic and numeric attributes describing invariant, as well as specific image features. Three description levels are taken into account [32], [14]:

- *Physical* description details image formation such as technical image features (e.g., size, type of coding), conditions of acquisition (e.g., type of camera, photometric features) and image quality (e.g., nature of noise, signal/noise ratio) ;
- *Perceptual* description deals with the image contents such as the type of data (e.g., clouds of points, density image) and the specific features of structural elements (e.g., nature of boundaries, contrast on boundaries, region texture) ;
- *Semantic* description defines the image representation and gives a sense to objects for the application under study, in terms of individual object properties (e.g., minimal size of objects, form of objects) and inter-object relations (e.g., object distribution within the image, inclusion and topological relations).

Image data is given as a data file that is used during the execution phase.

The request can be formulated beforehand (e.g., OCAPI, VSDE, CONNY) or interactively during the resolution process (e.g., DIA-ES). In any case, the description of the context must be done in close connection with the specifications of the tasks to perform, because the latter highlight several specific features that must be defined more precisely. For instance, in the case of an objective of boundary detection, it is obvious that the nature of boundaries must be provided.

2.3.2. Planning proceeds along at least three successive abstraction levels: intentional, functional and operational.

Planning must avoid two pitfalls: to associate too early or too late the initial task to operators. The former needs little large-grain knowledge that is difficult to define and that is hardly adaptable. By contrast, the latter needs a lot of small-grain knowledge that is too scattered and that is difficult to control.

To manage image processing problems three levels of abstraction are considered [10] [16]:

1. *Intentional*- This level is concerned with the description of what to do and why ;
2. *Functional*- This level describes techniques that can process information with respect to the specifications of task at the intentional level ;
3. *Operational*- This level details the way to implement techniques. It corresponds to the selection of operators and the tuning of their parameters with respect to the specifications of task at the functional level.

These three levels can be found in a more or less explicit way in many systems (e.g., CONNY, VSDE, OCAPI (named Goal, Complex-operator and Executable-operator levels)). These levels are also comparable to those that Marr [20] used in its theory of vision: the *computational theory* level, the *algorithm* level and the *implementation* level.

2.3.3. A conceptual solution is represented by a hierarchical tree of tasks ordered by data flow.

A solution plan is represented as a tree of tasks accounting vertically for decomposition relations of a task into subtasks, and horizontally for execution order. Each task of the tree broadly describes an abstract decision to transform input data into output data. Its level of abstraction is one of the three previous levels (e.g., in OCAPI *Noise-removal* and *Filtering* are tasks corresponding to two successive abstraction levels of a preprocessing task; in VSDE the same tasks are called *Noise-reduction* and *Non-linear-filter*).

This solution plan generally results from the instantiation of skeletal plans coded as AND/OR trees of tasks (e.g., CONNY, OCAPI and VSDE). The resolution process then roughly consists in finding an optimal path in the tree indexed by the task selected by the user and in instantiating the behaviour of the tree to the particularities of the context and the task (by

means of parameter values, constraints on tasks, control variables). Production rules attached to the various nodes of the tree are here to decide which OR branch to choose, by analyzing the context and constraints of the application given beforehand by the user.

2.3.4. Execution of operators integrates iterative and repetitive control structures.

As the a priori determination of parameter values is a complex problem [21], one has to propose mechanisms enabling an operator to compute its own parameter values. This can be done by successive executions of the operator with various potential parameter values. Potential values can be taken, either around default values, or around values computed from preceding operators. The selected values must satisfy (e.g., OCAPI), or optimize (e.g., LLVE) some given evaluation function. This evaluation function is based, either on measures calculated directly on the output images (e.g., if *number-of-regions<100, then increase the parameter value by 2* in OCAPI), or on comparison measures between output images and some reference images (e.g., *take the threshold value providing the best consistency between an edge image and region boundaries obtained by the binarization operator* in LLVE).

This mechanism must be explicitly represented within the tree of tasks. The flow of images, the flow of parameter values and the flow of reference images must be distinguished into the tree of tasks.

2.3.5. Masking implements a focus of attention mechanism on specific parts of images.

The concept of data masking enables to focus on parts of images [21]. A mask refers to image areas, on which to focus attention. Masking consists in applying an operator only on the pixels of the input data belonging to the focus of attention areas. When masking is associated to operators for combining results, it is then possible to implement top-down analysis strategies corresponding to (1) the selection of regions of interest through the masking mechanism, (2) the application of various local treatments on these regions and (3) the reconstruction of a final result by spatial or logical combination of previous results.

The masking mechanism must be explicitly represented within the tree of tasks. Operators that create these masks are standard operators and the flow of masks is made explicit.

2.3.6. Evaluation is distributed within the plan and organized into a hierarchy.

Evaluating or optimizing the output of each operator is not enough to assess the quality of a

whole plan of actions. Complete evaluation must be based on an assessment of each part of the plan [10], [8]. Each time a decomposition choice is made, one should try to define rules which will serve to verify its relevance (e.g., "*if :assess too-small object-size and no other-object absent, then :assess ambiguous detection and :failure*" is the evaluation rule of the task *coarse-object-detection* in OCAPI). Thanks to this principle, one can benefit from an evaluation along several abstraction levels. Such a delocalized and hierarchical evaluation means that operator results must be brought up toward higher levels.

Very few systems really take this evaluation stage into account, because defining assessment rules is one of the most difficult problems in image processing.

2.3.7. Correction combines a global and local failure handling mechanism to deal with respecialization and replanning of defective parts of plans.

The evaluation mechanism can highlight some unsatisfactory tasks. A correcting stage should thus propose new alternatives for these parts of the plan. A good failure handling mechanism must perform two types of operations [22]:

- *Respecialization* of defective parts of plans with new parameter values ;
- *Replanning* through backtracking, so as to change defective parts of the plan.

A distributed evaluation makes the localization of errors easier, but it does not necessarily account for the origin of errors. Adjustment rules can be added to propose a fresh plan execution by changing constraints or parameter values, or to propose backtracking toward previous choices (e.g., OCAPI).

In most systems however, only local failure handling is performed by backtracking. This mechanism does not necessarily account for the real origin of the failure. A global failure handling mechanism should advantageously be introduced to detect global impasses.

3. OVERVIEW OF THE BORG SYSTEM

BORG is a knowledge-based system for the supervision of a library of image processing operators. It was designed following the prescriptions of our study on knowledge-based system for operator supervision in image processing described in Section 2. In the present section, the motivations and the specific objectives of our approach are first presented, then the global architecture of the system, in charge of the complete process of planning,

instantiation, execution, evaluation and correction of plans is described.

3.1. Motivations and objectives

BORG can perform several image-processing tasks in widely varying application domains. Its role is to build a chain of operators that can process every image of a given class. The notion of image class is here restricted to a series of images of the same scene, taken with the same camera and in the same acquisition conditions. In particular, all images have the same resolution. To ensure that the same chain of operators can process every images of the class, such a chain of operators must integrate control structures to adjust its behavior.

Our system must be considered, both as an experimentation tool (Fig. 2a) enabling users to prototype their own applications, and as a knowledge acquisition tool (Fig. 2b) providing image processing experts with a framework for the representation of knowledge [9].

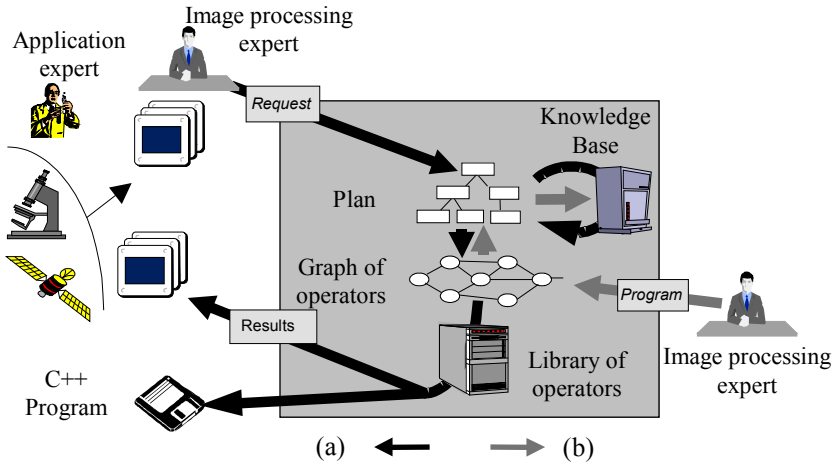

Fig. 2. The Borg system viewed as an experimentation tool (a) and as a knowledge acquisition tool (b).

3.1.1. An experimentation tool

Designing an application involves the active participation of two types of experts: an expert from the domain of application (biology, geography...) to set the problem, and an expert in image processing to formulate the corresponding request by selecting relevant tasks and judicious quantitative and qualitative image features. This problem/request "translation" is not straightforward; several comings-and-goings are generally needed to get a satisfactory request. In this respect, our system can be seen as an experimentation tool because it allows rapid production of results and favors dialogue and cooperation between both protagonists. The design of an application with BORG follows a three-stage process:

1. *Initialization*- This first step aims at getting a realistic vision of the difficulties of the

tasks to be performed and at identifying relevant perceptual indices. From a summary description of the application, given by the domain expert, the image-processing expert can suggest a first version of the request. Accordingly, a first version of the chain of operators can then be built by the system and first output images produced;

2. *Improvement*- Through visual assessment of these results, the domain expert is now in a position to refine the request. As a matter of fact, images are the best communication medium between both experts. The request can then be progressively refined, by analyzing various results corresponding to slight variations on its formulation;

3. *Validation*- Once results are judged satisfactory for several images, the validation step consists in testing this same request on the whole set of images representative of the application, to make sure of its robustness. If necessary, the request may be further refined, to take into account particular cases not encountered before.

At the end of this third step, one may hope that the chain of operators actually represents the application under study. The generator of C++ code then produces a program corresponding to this application, following the same approach as [27].

3.1.2. A knowledge acquisition tool

This knowledge acquisition step is necessary, on the one hand to enrich the system itself by allowing to tackle new tasks in new contexts, and on the other hand, to accumulate image processing know-how in an explicit and reusable form. In this respect, our system can be seen as a knowledge acquisition tool that favors dialogue between various image-processing experts. As experts must share the same library of operators and the same knowledge representation, they can refine and criticize the knowledge base, in order to update it. The acquisition of new knowledge is taking place through the resolution of new applications. Within the system, this process is done in three steps:

1. *Programming*- An application is built "manually" as a chain of operators taking the form of a script file, by coding, if need be, new operators and adding them to the library. One must take care to test this program on several images representative of the application

and to try to optimize the chain of operators and their parameters;

2. *Abstraction*- Once the program built and tested, it must then be abstracted into a plan of actions, so as to exhibit and represent the know-how associated to the application. One must here attempt to reuse preexisting tasks as much as possible;

3. *Modeling*- The last step consists in creating knowledge enabling to "replay" this plan of actions. The knowledge in charge of the evaluation process should be updated by comparing the various ways to reach a same goal. One also has to enrich the vocabulary used to formulate the request, with new tasks and new descriptive attributes.

BORG offers user-friendly interfaces allowing experts to add new knowledge and new operators in the system as well as the visualization of all potential trees that it can produce to help expert to maintain the knowledge base and the library of operators.

3.2. Architecture of the system

The architecture of the system (Fig. 3) takes up the BB1 Blackboard architecture [17] and uses the PANDORE library of operators [10]. This architecture makes a distinction between the resolution of image processing problems and the resolution of control of the resolution problems. For each category of problem, there are distinct databases and knowledge bases.

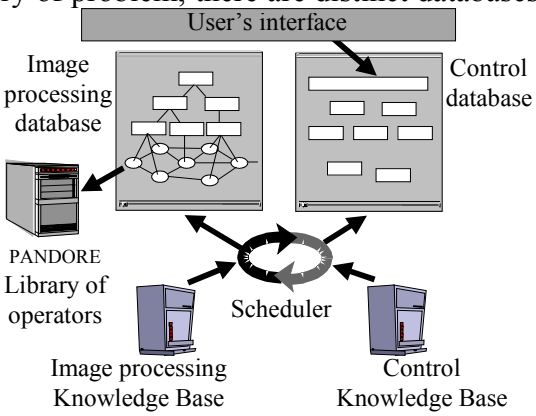

Fig. 3. The architecture of the Borg system.

Each knowledge base is composed of knowledge sources. A knowledge source can be viewed as a large rated rule. The condition part indicates data that must be present in the current plan before the action part is activated. Additionally, an aptitude part is defined to rate the confidence in the knowledge source contribution. The solution is built one step at a time, by successive activation of action parts of several knowledge sources. The control mechanism

uses the aptitude part of knowledge sources to decide which knowledge source to execute at each step of the resolution process.

3.2.1. Image processing knowledge base

The image processing knowledge sources are in charge of building the solution plan, of its instantiation with specific PANDORE operators, of controlling the execution of operators, of evaluating and, if necessary, correcting the plan, by updating tasks in the plan. The plan is built one block at time, instantiated one operator at a time, executed one operator at a time and evaluated one block at a time. Each knowledge source performs only one of these actions and brings only one small contribution to the global solution.

3.2.2. Image processing database

The image processing database contains the plan of actions that will be built to answer a request specified by the user through an appropriate interface. The plan of actions is represented as a tree of tasks, the leaves of which form a chain of operators. Such a tree differs from the AND/OR trees of tasks described in section 2.3.3, which embody all known decomposition alternatives for a given task; with AND/OR trees, the solution is built by resolving the tree. In our case, the tree corresponds to a solution plan and does not embody alternatives. It is entirely built from scratch for a specific user's request. At the beginning of the resolution, only the initial task defined by the user is present in the database. At the end of the resolution, the resulting tree describes the complete plan down to the operators.

3.2.3. Control database

The control database contains decisions that will be used to direct the choice of the knowledge sources to be executed at each step of the scheduler cycle. This choice results from a compromise between desirable actions and feasible actions [17]. Desirable actions are described by a general resolution strategy, and general-purpose behavioral heuristics. The strategy is implemented as successive decisions specifying on which level of the image processing database the attention should be focused. Behavioral heuristics are implemented as independent decisions heuristics indicating which knowledge source aptitude should be preferred. Feasible actions are stored in a list of executable knowledge sources that is also represented as a decision in the control database.

3.2.4. Control knowledge base

The control knowledge sources can create, modify or delete control decisions, as the problem–solving is making progress. In particular, they define the general resolution strategy and various behavioral heuristics. Thanks to this separation between domain and control, image processing knowledge sources and control knowledge sources are continuously competing with one another.

3.2.5. Scheduler

Control decisions are used by the scheduler to compute the priority of each pending knowledge source. The knowledge source with the highest priority is chosen for execution. Each execution of a knowledge source corresponds to one scheduler cycle, which can be decomposed into the three following steps:

1. Update the agenda of the pending knowledge source (i.e., with conditions satisfied) ;
2. Choose from the agenda the knowledge source with the highest priority ;
3. Execute the chosen knowledge source.

4. CONCEPTUAL MODEL OF THE SYSTEM

The conceptual model of the system is inspired by the Blackboard model and based on the prescriptions highlighted in Section 2. It is composed of a model for request formulation, a model for the plans of actions, a model for image processing knowledge sources and a model for the control of resolution. In this section, these various components of the conceptual model are successively described, justified and illustrated thanks to the following example.

4.1. Example of a biomedical application

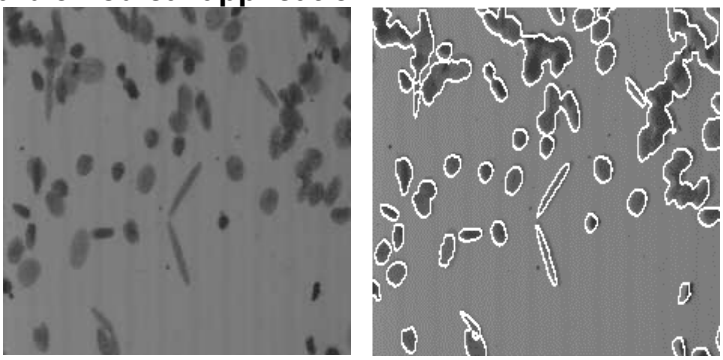

Fig. 4. On the left, an example of an initial image of the cytological class of images, on the right the expected result superimposed on the initial image.

Let us consider an example of a biomedical application dealing with microscopic images of

human esophagus cells (e.g., Fig. 4). This application, which is detailed in [15], has been developed within the framework of an image analyzer dedicated to the detection of abnormalities of ploidy in human cancerous tumors. The request related to this application consists in isolating cellular objects scattered on the image background, for further classification into several classes of cells (lymphocyte, epithelial, stromal).

4.2. Request formulation

The formulation of requests to the BORG system adheres strictly to the principles of **Rule 2.3.1**. A request is composed of (1) constrained tasks to perform, (2) a context characterizing the image to process and (3) one image from the image class.

4.2.1. Expression of request

(1) The tasks refer, at a high abstraction level, to image processing goals (e.g., region extraction, image enhancement, and image segmentation). Tasks deal with image processing objects (e.g., boundaries, regions, groups of objects), that can be particularized (e.g., high convexity, low surface). Requests can formulate problems such as "extract areas corresponding to groups of objects", or "extract objects exhibiting a high convexity degree". (2) Constraints correspond either to a quality degree, a restriction or a criterion to be optimized. They are expressed as relations on object attributes. (3) Context is described with attribute-value pairs where attributes are image descriptors belonging to the three level of description: physical, perceptual and semantic.

Formally, users have to express their request, in conformity with the grammar of Table 1.

```
task:  (Isolate objects)
constraints:  (Object-Separation = partial)        ; Level of detail
 (Boundary-Localization = precise); Optimization
 (Clusters = to-be-kept)                ; Restrictions
context:   ; Physical description
 (camera-type = microscope)             (image-length = 256)
 (image-width = 256)                    (image-noise = low)
 (image-quality = sharp)                (image-type = density)
 (background? = yes)
          ; Perceptual description
 (background-texture = homogenous       (background-colour = light)
 (contour-type = step)                  (grey-level     =     discriminative)
 (region-texture = homogeneous)         (region-boundaries = contrasted)
 (object-number-of-classes = 1)
          ; Semantic description
 (object-size = average)                (object-color = dark)
 (object-repartition = dissociated)     (object-min-size = 36)
image: ("/images/cytology11.hrs")
```

Table 1. Formal grammar used for specifying a request.

4.2.2. Example of a request in biomedical application

The request corresponding to the cytological application is shown in Table 2. In this case, "isolating" means extracting regions corresponding to cellular objects, but also to clusters of cells and debris. Constraints specify that a distinction should be made between isolated objects and clusters of objects and also that boundaries should be localized as precisely as possible. The application context lays the emphasis upon the fact that objects and background can be separated and characterized thanks to their gray level, and that isolated cellular objects are convex-shaped and medium-sized.

4.3. Image processing database

Answering such a request is achieved through the dynamic construction of a hierarchical

```
request::= ((<task> <constraint>*)+ <context> <image>)
task::= (<goal><object-type> <specific-feature>*)
specific-feature::= <attribute> <value>
constraint::= (<attribute> <relation> <value>)
context::= ((<attribute> = <value>)*)
goal::= enhance | extract | isolate | segment...
object-type::= region | boundary | background | contour | zone | object...
attribute::= region-boundary | object-color | camera-type | contour-type...
relation::= < | > | <= | = | <> ...
value::= <numeric> | <symbolic>
symbolic::= high | big | small | precise | sharp...
image::= "filename"
```

plan.

4.3.1. Abstraction levels

The plan of actions associates a request to a chain of PANDORE commands along five levels of abstraction. These levels correspond to a more detailed hierarchy than the three levels of **Rule 2.3.2**. Now, the intentional level is composed of two levels: *Request* and *Objective*. The functional level is called *Functionality*. And the operational level is decomposed into two levels: *Procedure* and *Operator*. Let us describe these five levels more precisely:

1. *Request*- The request represents the coarse problem as it is formulated by the user (e.g., (isolate objects) or (count objects)).
2. *Objective*- At the objective level, one has to specify elementary tasks that are parts of the analysis strategy selected to solve the request. Objective tasks are described with verbs (e.g., (eliminate background)).
3. *Functionality*- It is the description of an abstract image processing function that refer to some category of image processing operators. Functionality tasks are described with substantives (e.g., (pixel classification) with the constraint (number-of-object-classes = 2).
4. *Procedure*- At this level can be found all image-processing techniques, as they are referred to in the literature. A procedure task is the description of an algorithm (e.g., (binarization by maximizing contrast on boundaries)).
5. *Operator*- An operator reifies an executable command of the PANDORE library. This object describes the syntax that must be used to call the corresponding command, and provides values for its parameter (e.g., (binarization) with the constraint (threshold = 125).

4.3.2. Plan representation

The five levels of a plan of actions constitute the corresponding five levels of the image processing database. A plan is represented as a five-level high AND/THEN tree of tasks. This tree embodies the decomposition relations of the tasks at one level, into subtasks at the next lower level, as well as data flow exchanged between tasks.

The THEN link is used to group together subtasks that must be executed sequentially

(because the outputs of the first subtasks are inputs of the next ones in the sequence). The AND link is used for groups of subtasks that can be executed in parallel (because there is no dependence between inputs and outputs of subtasks).

The decomposition link used in the tree translates a refinement decomposition and not a functional decomposition. Therefore, the tree of tasks has exactly five levels. The difference between the two decompositions can be illustrated with the problem of tuning operators. With functional decomposition models, this problem is generally seen as a new task, leading to an additional decomposition. In abstraction decomposition, this problem is considered as a task that is decided at the same time as the task associated to the use of this operator. In other words, both tasks are parts of the same decomposition and can be found at the same level.

All tasks of the plan, whatever their level in the tree, share the same list of attributes (Table 3), that turn them into self-sufficient entities.

The *goal* and *constraints* define the task nature and the *context* points to the description of the image class. They are formulated by means of the same grammar and vocabulary as requests. The weight, a symbolic value of the interval [very-low, low, medium, high, very-high], gives an estimation of the goal importance. The creation and deletion of tasks is managed thanks to the *status* attribute, which takes the value “operative” when the task is created, and turns to “inoperative” when it is to be deleted. Thus, there is no physical destruction of a task, which enables its examination after it has been deleted, for explanation or debugging purposes. At the Operator level, *constraints* either contain values for each parameter of the operator, or the way to get them from the *return-value* of preceding operators.

Table 3. Attributes describing image processing tasks.

Attribute	Description
Goal	Name of the task to be performed
Constraints	List of specific constraints associated to this task
Context	Pointer to the global context of the application
Status	Status (operative/inoperative) of the task
Inputs	List of input images
Decomposition	Decomposition relation (and/then) of the task into subtasks
Path	Path telling how to get output images
Outputs	List of output images
Evaluation	Evaluation rules to assess the quality of outputs
Result	Result of evaluation (success/failure)
Weight	Estimation of the goal's importance.

4.3.2.1 Data flow

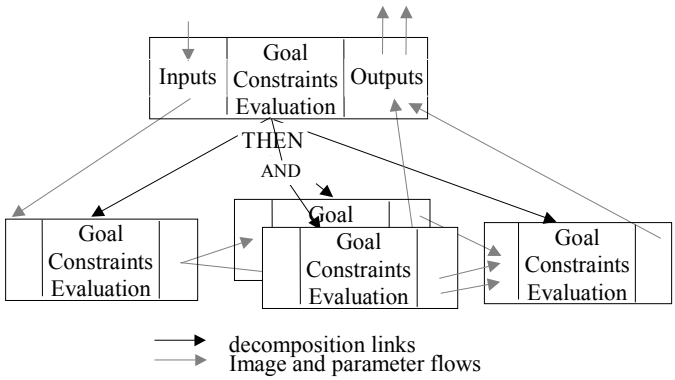

Fig. 5. Data flow between a task and its decomposition into subtasks.

The *inputs* and *outputs* attributes define the data flow (Fig. 5). Input images of a task either correspond to the inputs of its parent task, or to the outputs of one of its preceding tasks at the same level and within the same decomposition. The *path* is a convenient means to tell where to get the outputs of the task, among the outputs of its subtasks. In the case of an operator, its execution directly produces the operator *outputs*.

An image is here represented as a pair (*pixel image, region map*), both being stored as image files. The region map, if it is given, is considered as the mask to be applied to the pixel image. This principle enables to generalize the masking process introduced in **Rule 2.3.5**, a same task being performed on a complete image, or only masked parts of it. When bringing results back up the tree, it is possible to compose the output image from the region map of an image and the pixel image of another. In the same way, all PANDORE commands have an optional argument, an input region map, which is used as a mask for the pixel image. Moreover, one can define tasks or PANDORE commands that directly take region maps instead of pixel images as their inputs, thus bringing some standardization to our approach.

4.3.2.2 Operator execution modes

Operators are described by three additional attributes: *prototype* contains the operator syntax of call and its number of parameters, *cost-in-time* gives an estimation of the operator complexity (in terms of the time required for its execution) and *mode* indicates its execution mode. Three modes are available (cf. **Rule 2.3.4**):

- The *normal* mode corresponds to a unique execution of the operator ;
- The *loop* mode corresponds to successive executions of the operator with the same parameter values, each execution taking as inputs, the outputs of the previous one ;
- The *optimization* mode corresponds to successive executions of the operator with the same inputs but with various sets of parameter values. The operator *constraints* specify, for each parameter, the interval of values to be tried, and the increment step.

4.3.2.3 Evaluation rules

The output images of a task are considered as an acceptable *result* when they satisfy the task *evaluation* rules (cf. **Rule 2.3.6**). Each task is thus in a position to assess the result of its own decomposition into subtasks. There are two exceptions to this rule: at the Request level, the user is in charge of the final visual evaluation of the whole plan, and at the Operator level, evaluation rules are only used to control the execution of the command (see Section 4.3.2.1). This assessment consists in checking whether results are in conformity with expectations, as they are described by the context and constraints of the application; the objective being mainly to avoid aberrations.

Evaluation rules are production rules following the “if…then…else” format. The condition part deals with statistical comparisons between quantitative measures performed on the task outputs and reference measures deduced from the goal specification. Measures on outputs are computed by PANDORE commands, for which there is no tuning problem. The action part simply consists in setting the *result* attribute of the task either to “success” or to “failure”.

4.3.3. Example of a plan in cytology

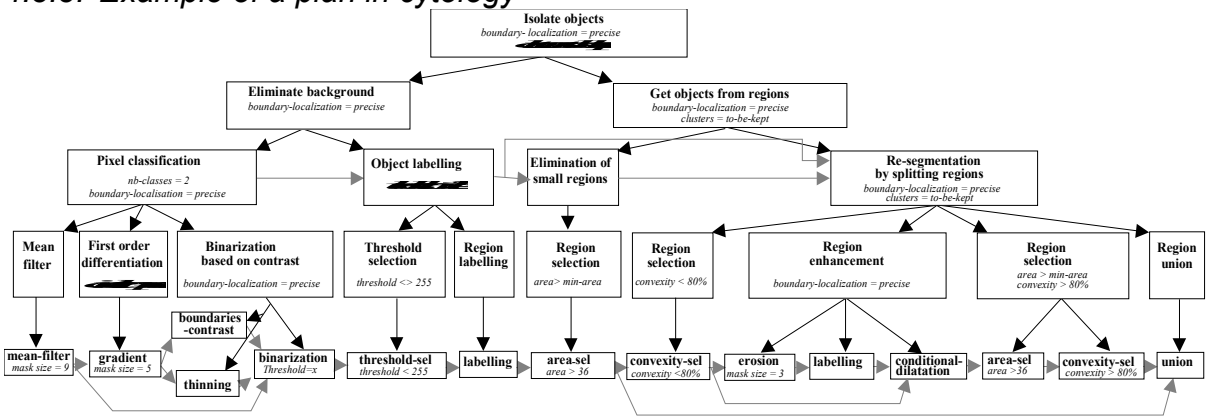

Fig. 6. The plan for the cytological application. Image data flow at the functionality level and at the operator level are by drawn with gray arrows.

Fig. 6 gives an example of the plan built to answer the request of Table 2. This plan implements a top-down image analysis strategy consisting in, first eliminating the image background, in order to keep regions of interest only, then extracting objects – the cells – from these regions.

One should notice that in this plan, the binarization operator is executed in "optimization" mode, so as to optimize the localization of boundaries. The interval of the threshold values to be tried is built around the return-value of the Boundary-Contrast operator and the evaluation function count corresponds to the situation when the number of pixels common to the region boundaries and to contours detected by a Thinning operator is maximal.

The erosion operator is repeated twice ("loop" mode), so as to separate objects that are partially touching each other, but without separating objects that are clearly overlapping (which correspond to clusters of cells). The number of iterations is calculated from the minimum size of the objects, which is given in the application context.

The eight last operators take as their input a mask which is the result of the first objective.

Evaluation rule of the "pixel classification" task check whether the output image actually has two classes of pixels, by calling a PANDORE command that counts the number of object classes in an image: *If (count-classes (output)=2) Then success Else failure.*

4.4. Image processing Knowledge Sources

Each small action contributing to the construction of a plan is performed by applications of

successive knowledge sources. In our knowledge base, there are five categories of knowledge sources: *planning-KS*, *instantiation-KS*, *execution-KS*, *description-KS,* and *evaluation-KS*. With these categories of knowledge sources, correcting the plan is done by proposing new decompositions for tasks that failed, and assessing results is done in two steps: a step when evaluations rules for each task are defined by *description-KS*, and a step when results are built and evaluated by *evaluation-KS*.

4.4.1. Knowledge sources model

Image processing knowledge sources are represented by the same list of attributes (Table 4). Knowledge sources can take four different states: *triggered*, *executable*, *executed* or *obviated*, depending on what condition parts are satisfied. Only triggered knowledge sources can become executable, and only executable knowledge sources can be executed.

Table 4. Attributes describing image processing knowledge sources.

Attributes	Description
Trigger-conditions	Predicates describing events that signal applicability.
Pre-conditions	Predicates describing data that must or must not existed on the plan to permit applicability.
Obviation-conditions	Predicates describing data of the plan that obviated applicability.
Condition-variables	Local variables used to parameterize each of the knowledge source parts.
Input-Level	Level of triggering.
Output-Level	Level of action.
Cycle	Cycle of the triggering.
Action	Rules that implement the contribution in terms by updating the plan of tasks.
Importance	Rules that rate the execution priority for the resolution process.
Credibility	Rules that rate the adequacy between the contribution and the requirements.
Reliability	Rules that rate the likelihood to get satisfactory results.
Complexity	Rules that rate the number of operations to perform and the number of parameters to tune.

The *trigger-conditions* part refers to creation or modification of one task at one abstraction level. This is a means to identify all the competitive knowledge sources attached to the task.

The *pre-conditions* and *obviation-conditions* parts refer to expected attribute values of one or several tasks of the plan, thereby they can be used to manage competitive knowledge sources. Preconditions are used to suspend execution of alternative knowledge sources while one of them is being trying (i.e., the *decomposition* attribute of the related task is not null). Obviation-conditions are used to discard all remaining alternatives when results are validated (i.e., the *result* attribute of the related task is set to "success").

The *condition-variables* part allows multiple activation of a knowledge source, in response to each occurrence of the event described by the trigger-conditions. Each activation is recorded

into a knowledge source activation record (KSAR) that represents the knowledge source with the context in which the knowledge source can act. Parts of the variables are bound during the trigger-conditions validation and others during the pre-conditions validation.

The *aptitude* part is described by seven independent attributes: *importance*, *credibility*, *reliability, complexity*, *input-level*, *output-level* and *cycle*. These attributes are intrinsic characterizations of knowledge source contribution, and not measures of comparison between knowledge source contributions. They are set to a traditional IF-THEN rule, which consists in producing a symbolic value among the interval [*very-low, low, medium, high, very-high*], with regards to the characteristics of the application context and the constraints of the task to perform. These rules are interpreted when the knowledge source is triggered and the resulting condition-variables values are stored into the related KSAR.

The *action* part of a knowledge source consists in a set of parameterized changes of the plan to be instantiated with the *conditions-variables* values. Changes are creation, deletion or modification of tasks of the plan. In practice, it is the related KSAR that is interpreted. As far as knowledge acquisition is concerned, knowledge sources must absolutely ignore each other and be as modular as possible. There is no a priori grain-size for the knowledge included in the action part, except that the control problem inside a knowledge source must be completely solved [1]. It means that one knowledge source codes one version of an action on the plan. However, a knowledge source has to make all adjustments necessary for its correct execution, with regards to the application context and constraints.

4.4.2. Planning Knowledge Sources

Each planning-KS (see example Table 5) codes a know-how to perform some given task, by proposing a decomposition of this task into an ordered sequence of subtasks at the next lower level. For each subtask, the goal, constraints, weight and data flow must be completely specified. There are as many planning-KSs attached to a task that there are alternatives for decomposing this task.

Table 5. An example of planning-KS: Contrast-Classification performs pixel classification through an image binarization based on the maximisation of the contrast on boundaries.

Attributes	Values
Trigger-conditions	There is a new functionality F / F's goal = (pixel classification) and (nb-classes=2) ∈ F's constraints.
Pre-conditions	F's decomposition =()
Obviation-conditions	F's status = inoperative or F's result = success
Condition-variables	(F)
Input-Level	Functionality
Output-Level	Procedure
Action	Create Procedure P1 P1's goal = (mean filter) P1's constraints = () P1's weight = medium P1's inputs = (1st of F's inputs) Create Procedure P2 P2's goal = (first order differentiation) P2's constraints = ((gradient-direction-image = present)) P2's weight = medium P2's inputs = (1st of P1's outputs) Create Procedure P3 P3's goal = (binarization based on contrast) P3's constraints = F's constraints P3's weight = medium P3's inputs = (1st of P1's outputs, 2nd of P2's outputs, 3rd of P2's outputs) F's decomposition = (THEN P1 P2 P3) F's path = ((1st pixel image of P3's outputs, 1st region map of P3's outputs))
Importance	Medium
Credibility	IF (gray-level=discriminative)∈ F's context THEN high ELSE very-low
Reliability	IF (object-boundary = contrasted) ∈ F's context and (object-texture = homogeneous) ∈ F's context THEN high ELSE very-low
Complexity	Low

Planning-KSs can be associated to different semantics, according to their input and output levels: Planning-KSs linking the Request and Objective levels correspond to various image processing analysis strategies (bottom-up, top-down, mixed). Planning-KSs linking the Objective and Functionality levels represent an image processing know-how. Planning-KSs linking the Functionality and Procedure levels are coding image processing technical skills.

4.4.3. Instantiation Knowledge Sources

Instantiation-KSs are very similar to planning-KS, in the sense that some part of their action also deals with the decomposition of a task into subtasks. But, they are also in charge of assessing the operator cost-in-time, providing values for its parameters, or the way to get them from others operators, and evaluation functions for operators that must be executed in the "loop" or "optimization" mode.

4.4.4. Execution Knowledge Sources

Execution-KSs are responsible for the execution of the PANDORE operators. Their first role is to build the command line to be sent to the Shell interpreter, and the second one, to gather

results, i.e., return-values of operators and images files. There are three execution-KSs, one for each execution mode ("normal", "loop" and "optimization").

4.4.5. Description Knowledge Sources

Description-KS (see example Table 6) are in charge of dynamically building evaluation rules. There is only one description-KS for each task, because evaluation rules are only depending on the context and constraints of the task, and not on each decomposition alternative that can be proposed. Our evaluation rules are build with calls to PANDORE commands.

Their dynamic building takes advantage of the hierarchical nature of the plan to produce rules adapted to each level. Generally, at the highest levels, rules are dealing with the characteristics of objects as they are given in the application context, whereas at the lowest levels they are rather performing verifications on the goal constraints. When no relevant rule can be defined, the default-rule "IF true THEN success" is used.

Table 6. An example of description-KS: Pixel-Classification-Description builds an evaluation rule for the goal "pixel classification", in order to check right number of pixel classes in the output image.

Attributes	Description
Trigger-conditions	There is a Functionality F / F's goal = (pixel classification)
Pre-conditions	()
Obviation-conditions	F's status = inoperative
Condition-variables	(F)
Input-Level	Functionality
Output-Level	Functionality
Action	F's evaluation = IF the attribute nb-classes ∈ F's constraints THEN "If (count-classes(F's outputs)=nb-classes) Then success Else failure" ELSE Default-Rule
Importance	High
Credibility	High
Reliability	Medium
Complexity	Very-Low

4.4.6. Evaluation Knowledge Sources

Evaluation-KSs (see example Table 7) are in charge, of building the task results from the results of some of its subtasks, and then of applying the evaluation rules that have been defined by description-KSs. Owing to the fact that evaluation-KSs do not contain distinctive expertise, one single evaluation-KS for each hierarchical level of the plan is sufficient.

Evaluation-KSs are applied in two cases: when all subtask results are judged acceptable and when one of them has failed. In the first case, the task outputs are obtained thanks to the *path* attribute, then evaluation rules are applied to these outputs to check whether the

decomposition failed or not. In case of failure, the evaluation-KS deletes the current task decomposition, in order to clear the ground for another decomposition if there exists one. In the second case, one proceeds as in the first case when evaluation rules failed.

Table 7. An example of evaluation-KS: Functionality-Evaluation builds and evaluates the result of a functionality.

Attributes	Description
Trigger-conditions	There is a Functionality F / F's path ≠ ()
Pre-conditions	IF (FOR all task T of F's decomposition, T's result = success) OR (there is one task T of F's decomposition / T's result = failure) AND (F's evaluation ≠ ())
Obviation-conditions	F's status = inoperative
Condition-variables	(F)
Input-Level	Functionality
Output-Level	Functionality

Action	IF (FOR all tasks T of F's decomposition DO T's result = success)
	THEN F's outputs = Interpret F's path
	F's result = Application of F's evaluation rules
	IF F's result = failure
	THEN FOR each subtask ST of F's decomposition DO
	ST's status = inoperative
	ELSE F's result = failure
	FOR each subtask ST of the F's decomposition DO
	ST's status = inoperative
	F's decomposition = ()

Importance	IF (FOR all tasks T of F's decomposition DO T's result = success) THEN high ELSE very-low
Credibility	IF F's evaluation = Default-Rule THEN high ELSE low
Reliability	Medium
Complexity	Very-low

4.4.7. Knowledge base for the cytological application

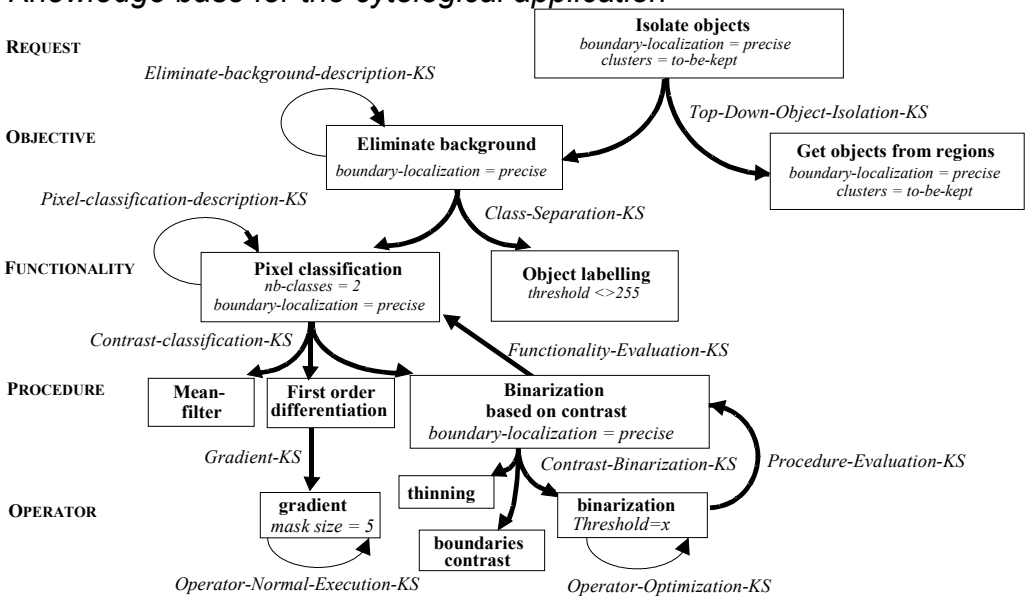

Fig. 7. Knowledge sources used in the cytology application.

Fig. 7 shows some of the knowledge sources that took part to the construction of the plan of Fig. 6. One should notice that the "Top-Down-Object-Isolation" planning-KS implements a top-down analysis strategy: first eliminating the image background, then building objects

from the regions resulting from the first stage. Another alternative approach could have been considered: first detecting region germs (for instance by detecting minimal gray-levels), then performing region-growing around these germs. The first strategy was preferred, because credibility and reliability attributes of the corresponding planning-KS are high when the image background is homogenous and can be easily separated from objects. The "Class-Separation" planning-KS has been selected, because the two classes corresponding to the image background and the objects can be separated on the simple basis of their gray-level. This separation is achieved through a pixel classification into two classes, followed by the background elimination. The "Contrast-Classification" planning-KS performs a pixel classification based on the image contrast and has been retained, because there is a high contrast at the boundaries of objects. Another quite as probable alternative would have been to use the inter-class variance instead of the contrast as classification criterion. In our plan, the first planning-KS was selected, simply because it was the first to be tried. In the plan, one should also notice that the "Pixel-Classification-Description" description-KS builds an evaluation rule to check that the output image actually contains two classes of pixels.

4.5. Control of resolution

The general analysis strategy we suggest for solving image processing problems is a depth-first strategy, corresponding to a plan construction in a top-down manner. This choice is completely empirical. This strategy is implemented as temporary focuses of attention, on the successive levels of the image processing database. The heuristics we use give preference to control knowledge sources versus image processing knowledge sources, to the most recently triggered knowledge sources, to knowledge sources with the highest credibility, the highest reliability, to the most important, the least complex ones, and to the knowledge sources that work on tasks that have the highest weight.

A weight is associated to focus-decisions or heuristic-decisions on the control blackboard that defines their importance in the specification of the desirable knowledge source. Thus, the strategy can be dynamically adapted to current constraints (e.g., cost-in-time, reliability). These constraints on control can be adjusted from the system interface, but are not included in image processing requests. Focus-decisions and heuristic-decisions are used to compute a

priority value for each executable knowledge source (i.e., KSAR), in order to select the next one to be executed: the KSAR (domain or control) with the highest priority.

Thanks to this type of control, the resolution behavior is both incremental and opportunistic. It is incremental because the agenda of pending KSAR is updated at each cycle. The selection of KSAR is based on the current state of progress of the resolution, which means that the resolution influences itself. The control is also opportunistic because the choice of the knowledge source to be executed is based on a combination of the general strategy with intrinsic preferences on some characteristics. It does not strictly follow the strategy. For instance, an execution-KS, which is considered as very important, priority to a planning-KS, even if the strategy is currently focusing on planning.

Our control enables to handle local failure through backtracking. A failure can be passed from one subtask to its parent task at the next higher level, if none of the potential alternatives gives complete satisfaction, or if the credibility and reliability values of the knowledge source dealing with these alternatives are lower than those of the corresponding evaluation-KS. Besides, it is not possible to re-execute some parts of a plan with new constraints, contrary to the prescriptions of **Rule 2.3.7**, because we here consider that each knowledge source proposes an optimal decomposition; if this decomposition fails, it implies that the whole decomposition is not adapted and must be changed.

Our control also enables to handle global failure through detection of impasse. It is not necessary to wait until all pending alternatives have been executed before the control reacts. By analyzing the aptitude value of pending knowledge sources, the control mechanism can force the propagation of failure upward higher levels. If the failure is propagated directly to the request task, this means that the application cannot be solved.

5. CONCLUSION

In this paper, we have studied the problem of automatic generation of image processing programs, through the supervision of a library of operators. We have shown, from a survey of different relevant systems, that this problem requires, on the one hand, the use of knowledge-rich problem-solving and on the other hand, the taking into account of seven design rules. The BORG system was developed on the basis of this study.

Compared to other systems that take into account the whole building process including the control of execution, our system presents several original aspects, in order to improve the planning, execution, evaluation, failure handling and knowledge acquisition stages.

- *Planning*- Representing decomposition knowledge as separated knowledge sources can bring an answer to the problem of the increasing number of alternatives associated to one single task. Knowledge sources are responsible to dynamically assess their own performance, in terms of features that do not depend on the domain of application (importance, credibility,...), this rating being also completely independent from the other knowledge sources. Each knowledge source only contains knowledge about the decomposition alternative it handles. Moreover, the selection of knowledge sources by the control mechanism is only based on those features. Thus, an increase in the amount of knowledge does not affect the problem-solving performances, on the contrary, it contributes to the improvement of the system abilities. Besides, the knowledge source model is a good means of favoring the reuse of tasks, because knowledge sources only perform local operations that can be reused in other contexts and applications.
- *Execution*- Our control of operator execution explicitly integrates all possible types of execution mentioned in other systems: tuning of parameter values through optimization, as in LLVE, or through simple constraint satisfaction, as in OCAPI, iterative or repetitive execution of the same operator, with the same parameter values, as it is done (but not in an explicit way) in OCAPI.
- *Evaluation*- As in OCAPI, the use of individual delocalized evaluation rules associated to each task makes their formulation easier and allows the use of criteria at several abstraction levels. Even if it remains particularly difficult to find out rules in the domain of image processing, the fact that our evaluation mechanism is both hierarchical and delocalized indisputably constitutes some guideline for the formulation of evaluation rules. Furthermore, BORG fixed five-level hierarchy of tasks provides a natural typology of rules, the formulation of evaluation criteria being based on the vocabulary used for the specification of tasks at the same of abstraction.
- *Correction*- The correction of errors is dynamically decided by the control.

Decomposition alternatives of a task are competing with the evaluation-KS in charge of propagating failure towards the next higher level. Thus, only the alternatives corresponding to knowledge sources showing a priority higher than the priority of the knowledge sources propagating the failure are kept. The computation of priority values is based on the user preferences (heuristics) and the current state of the resolution (focus of attention). In other systems, this control mechanism is static and pre-programmed: try all possible alternatives before propagating a failure and the rules used to select alternatives include their own control mechanism to decide which alternative to select.

- *Knowledge acquisition*- The acquisition of new knowledge is made easier thanks to our fixed number of hierarchical levels and to the use of autonomous and independent knowledge sources. Our five-level hierarchy of tasks provides some guidance for a more accurate identification of the knowledge associated to each task. Representing decomposition alternatives as knowledge sources enables to add new competence, without questioning what already exists, because each knowledge sources has a limited role and each alternative corresponds to one knowledge sources. When rules are used instead of knowledge sources, each new competence to be taken into account implies a complete rule modification. When the number of alternatives increases, rules become more and more difficult to write.

BORG presents a major drawback due to the use of such a knowledge-rich problem-solving. It leads to an almost continuous knowledge acquisition process. Each time an expertise is available, it is modeled and added to the knowledge base. As a matter of fact, one can only succeed in solving problems corresponding to well-identified tasks, contrary to knowledge-poor problem-solving, where one can hope "discovering" new solutions to problems that have never been tackled before.

Until now, BORG has mainly been used in the framework of biomedical applications. We have developed two major applications, the first one dealing with the segmentation of cell nuclei in cytological images (this application is used all along the paper as an example) and the second one with the localization of tumoral lobules in histological images. To solve these

two applications, we have written 59 image processing knowledge sources and 8 control knowledge sources. Our major objective and first source of satisfaction when developing these applications has been the reuse of a fair amount of tasks.

REFERENCES

[1] B. Bachimont, *Le contrôle dans les systèmes à base de connaissances*, Hermès, Paris, 1992.

[2] R. Bodington, "A Software Environment for the Automatic Configuration of Inspection Systems", *Proc. Int'l Workshop on Knowledge-Based systems for the reUse of Program Libraries*, pp. 100-108, Sophia Antipolis, France, Nov. 1995.

[3] A. Boucher and C. Garbay, "A multi-agent system to segment living cells", *Proc. IEEE Int'l Conf. on Pattern Recognition*, Vol. 3, pp. 558-562, Vienna, Austria, Aug. 1996.

[4] D. Carel, "Système Expert d'aide à l'utilisation d'algorithmes de traitement d'images", PhD Thesis, Univ. Rennes, France, 1989.

[5] D. Charlebois, "A planning system based on plan re-use and its application to geographical information systems and remote sensing", PhD Thesis, Univ. Ottawa, Canada, Oct. 1997.

[6] B. Charroux and S. Philipp, "Interpretation of Aerial Images based on Potential Functions", *Proc. 9th Scandinavian Conf. on Image Analysis*, pp. 671-678, Uppsala, Sweden, June 1995.

[7] S.A. Chien and H.B. Mortensen, "Automating Image Processing for Scientific Data Analysis of a Large Image Database", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 18, No. 8, pp. 854-859, Aug. 1996.

[8] V. Clément and M. Thonnat, "A Knowledge-Based Approach to Integration of Image Processing Procedures", *Computer Vision, Graphics and Image Processing: Image Understanding*, Vol. 57, No. 2, pp. 164-184, Mar. 1993.

[9] R. Clouard, M. Revenu, A. Elmoataz, and C. Porquet, "A Software Workshop for Knowledge Acquisition in Image Processing", *Proc. Int'l. Workshop on the Design of Cooperative Systems*, pp. 298-312, Juan-les-Pins, France, Jan. 1995.

[10] R. Clouard, C. Porquet, A. Elmoataz, and M. Revenu, "Why building Knowledge-Based Image Segmentation is so difficult", *Proc. Int'l Workshop on Knowledge-Based systems for the reUse of Program Libraries*, pp. 137-148, Sophia Antipolis, France, Nov. 1995.

[11] D. Crevier and R. Lepage, "Knowledge-based image understanding systems: a survey",

Computer Vision and Image Understanding, Vol. 67, No. 2, pp. 161-185, Aug. 1997.

[12] P. Dejean and P. Dalle, "Image Analysis Operators as Concept Constructors", *Proc. IEEE Southwest Symp. on Image Analysis and Interpretation*, pp. 66-70, San-Antonio, Texas, Apr. 1996.

[13] B.A. Draper, A.R. Hanson, and E.M. Riseman, "Knowledge-Directed Vision: Control, Learning and Integration", *Proceedings of IEEE signal symbols*, Vol. 84, No. 11, pp. 1625-1637, Nov. 1996.

[14] A. Elmoataz, "Mécanismes opératoires d'un segmenteur d'images non dédié: définition d'une base d'opérateurs et implantation", PhD Thesis, Univ. Caen, France, 1990.

[15] A. Elmoataz, M. Revenu, and C. Porquet, "Segmentation and Classification of Various Types of Cells in Cytological Images", *Proc. IEE Image Processing and its Applications*, pp. 385-388, Maastricht, Netherlands, Apr. 1992.

[16] L. Gong and C.A. Kulikowski, "Composition of Image Analysis Processes Through Object-Centered Hierarchical Planning", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 17, No. 10, pp. 997-1009, Oct. 1995.

[17] B. Hayes-Roth, "A Blackboard Architecture for Control", *Artificial Intelligence*, Vol. 26, No. 3, pp. 251-321, 1985.

[18] A.L. Lansky, M. Friedman, L. Getoor, S. Schmidler, and N. Short Jr., "The Collage/Khoros links: Planning for image processing tasks", *Proc. AAAI Spring Symposium: Integrated Planning Applications*, pp. 67-76, Menlo Park, Calif., 1995.

[19] C.E. Liedtke and A. Blömer, "Architecture of the Knowledge-Based Configuration System for Image Analysis *Conny*", *Proc. IEEE Int'l Conf. on Pattern Recognition*, pp. 375-378, The Hague, Netherlands, Aug. 1992.

[20] D. Marr, *Vision: A computational investigation into the human representation and processing of visual information*, Freeman and co., San Francisco, Calif., 1982.

[21] T. Matsuyama, "Expert Systems for Image Processing: Knowledge-based Composition of Image Analysis Processes", *Computer Vision, Graphics and Image Processing*, Vol. 48, No. 1, pp. 22-49, Oct. 1989.

[22] S. Moisan, R. Vincent, J. van den Elst, and F. van Harmelen, "Towards an Intelligent

Failure Handling Mechanims in Program Supervision", *Proc. Int'l Workshop on Knowledge-Based systems for the reUse of Program Libraries*, pp. 109-118, Sophia Antipolis, France, Nov. 1995.

[23] A.M. Nazif and M.D. Levine, "Low Level Image Segmentation: An Expert System", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 6, No. 5, pp. 555-577, Sep. 1984.

[24] H.P. Nii, "Introduction", V. Jagannathan, R. Dodhiawala, and L.S. Baum Eds, *Blackboard architecture and applications*, pp. xix-xxix, Academic Press, 1989.

[25] T.A. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory", *Nature*, No. 317, pp. 314-319, Sep. 1985.

[26] J. Rasure and S. Kubica, "The Khoros Application Development Environment", H.I. Christensen and J.L. Crowley Eds, *Experimental Environments for Computer Vision and Image Processing*, pp. 1-32, World Scientific, Singapore, 1994.

[27] K. Sakaue and H. Tamura, "Automatic Generation of Image Processing Programs by Knowledge Verification", *Proc. IEEE on Computer Vision and Pattern Recognition*, pp. 189-192, San Francisco, Calif., Jun. 1985.

[28] H.A. Simon, *The sciences of the Artificial*, M.I.T press, Cambridge, Mass., 1969.

[29] T. Tanaka and N. Sueda, "Knowledge acquisition in image processing expert system Explain", *Proc Int'l Workshop on Artificial Intelligence for Industrial Applications*, pp. 267-272, Hitashi, Japan, May 1988.

[30] M. Thonnat and S. Moisan, "Knowledge-based systems for program supervision", *Proc. Int'l Workshop on Knowledge-Based systems for the reUse of Program Libraries*, pp. 4-8, Sophia Antipolis, France, Nov. 1995.

[31] T. Toriu, H. Iwase, and M. Yoshida, "An Expert System for Image Processing", *Fujitsu Science and Technical Journal*, Vol. 23, No. 2, pp. 111-118, Jun. 1987.

[32] J. van den Elst, "Knowledge modeling for program supervision in image processing", PhD Thesis, Univ. Nice, France, 1996.