Development of Image Processing applications within a cooperative Knowledge-Based Workbench

Christine Porquet, Régis Clouard, Valérie Ficet-Cauchard, Marinette Revenu

DEVELOPMENT OF IMAGE PROCESSING APPLICATIONS WITHIN A COOPERATIVE KNOWLEDGE-BASED WORKBENCH

Christine PORQUET, Régis CLOUARD, Valérie FICET et Marinette REVENU

GREYC-ISMRA - 6 Bd du Maréchal Juin - F14050 CAEN cedex
tel: 31-45-27-01
fax: 31-45-26-98
e-mail: Christine.Porquet@greyc.ismra.fr

Abstract

Solving an Image Processing problem involves three main categories of experts: experts from the domain of application (biologists, geographers...), Image Processing experts and designer-programmers. Domain experts are requested to express clearly and exhaustively the purpose of their application. Image Processing experts have to propose some kind of answer in terms of Image Processing plans adapted to the specific features of the request and the nature of images. Designer-programmers have to provide a framework for an efficient implementation.

We are interested in setting up a Knowledge-Based workbench directed toward the cooperation between these experts and the system. This approach is motivated by the development of real-size applications, while being concerned in generic principles and knowledge acquisition and integration concerns. It hinges on the supervision of a library of Image Processing operators. Solving a specific application is a planning process which implies selecting the suitable operators, linking them and tuning their parameters. An application is thus represented as an Image Processing plan.

After giving the specifications of our system, which belongs to the more general class of Knowledge-Based systems for program supervision, our software architecture favoring cooperation as well as knowledge integration will be presented. This reflexive architecture is based on the TASK/METHOD/TOOL paradigm and provides conceptual models of applications that are actually computational. Several Image Processing plans dealing with biomedical (cytological / histological) images have already been built within our workbench, which constitutes a first convincing validation of our approach.

Keywords

Image Processing

Supervision of a library of Image Processing operators

Computational conceptual model

1 - Introduction

Image Processing (IP) is a domain where experts of various origins have to cooperate (experts from the domain of application (biology, cartography, geology, ...), IP experts and designer-programmers. Knowledge-Based systems can bring an answer to this cooperation requirement by favoring communication and insuring incremental knowledge acquisition. The objective of our research project of a software workbench for knowledge integration in Image Processing and Image Understanding is to develop such a Knowledge-Based system applied to Image Processing and Image Understanding [Revenu 95]. Our workbench hinges on the supervision of a library of Image Processing operators. Solving a specific application is a planning process which implies selecting the suitable operators, linking them and tuning their parameters. An application is thus represented as an Image Processing plan. We are neither aiming at realizing a general-purpose IP system, nor at only solving dedicated real-size applications, but chiefly at formalizing expert knowledge. First validations are done on biomedical (cytological / histological) images.

The work described in this paper deals with the construction of IP plans thanks to our reflexive architecture based on the TASK/METHOD/TOOL paradigm. After giving the specifications of our workbench (§. 2), we will detail the various types of knowledge that must be handled and explicitly represented (§. 3), then the TASK/METHOD/TOOL architecture will be described (§. 4). The last paragraph will be devoted to the validation of our approach as well as future prospects.

2. Specifications

Three types of experts play a part in the conception of IP systems and the resolution of IP problems: experts from the domain of application, IP experts and designer-programmers.

First, the domain expert has to define his/her problem by providing one or several images, a request and an application context. Once the corresponding plan is built and in order to test it, the system must then execute it by proposing choices at various levels, as well as possible backtracking from these choices (for instance, choice on methods or on parameters).

The role of the IP expert is to build a plan answering the problem set by the domain expert. Within our workbench, the IP expert no longer has to program in the classical sense of the term, or to have access to computer codes: he/she just has to create plans by organizing predefined basic building blocks.

The designer-programmer is in charge of implementing the control architecture and these basic blocks in order to create the system. Afterwards, the IP expert can appeal to him/her if new operators must be added.

These three roles are not independent from each other, which implies cooperation and communication requirements between experts. This communication first consists in making the decomposition of the initial problem into sub problems explicit. In order to favor dialogue between experts and the system, and to cut oneself off as much as possible from technical IP particulars, we propose to program at the *knowledge level* [Newell 82], i.e. to describe the resolution method by using a vocabulary than can be comprehensible for all experts. A computational conceptual model of the system enables the experts to intervene in the evaluation and tuning of the resolution method [Delouis 93]. Moreover one good means of favoring communication is to develop a graphical interface for visualizing IP plans and intermediate images [Willamowski 94].

Within our workbench, a first work was devoted to the conception of an automatic planning module named BORG [Clouard 95a]. Difficulties in the writing of knowledge sources led us to a more pragmatic approach of step-by-step construction of IP plans. However, our objective is to have our system and BORG cooperate, so as to build a plan, either manually with the help of the IP expert, or automatically by BORG.

The system can appeal to the domain expert or the IP expert when some data are missing, or to make choices between several techniques. But experts can also intervene when they want explanations or in order to modify some choices made by the system. To that purpose, users must be directly involved in the resolution loop, providing they wish to take part in the resolution process and detain the required knowledge. So the system has two functioning modes when executing a plan, *automatic* mode when the system makes choices, and *step-by-step* mode when it is the user that makes choices.

Our research takes place within the context of supervision of library of operators [Thonnat 95]. Contrary to systems such as OCAPI [Clément 93] or VSDE [Bodington 95], the objective of which is problem solving, in our system, the manual construction of new applications does not impose writing production rules or knowledge sources. This writing may take place afterwards, or not at all. When executing a plan, if a conflict arises between several methods, and no decision rule is available, a default-choice mechanism will apply. An IP expert can thus create or modify his/her own applications without affecting the other existing applications.

3 - Knowledge-Based systems in Image Processing

Knowledge-based systems can manage "abstract" data such as the description of the steps to solve some problem [Wielinga 92]. They bring into play strategic knowledge while authorizing the use of traditional data processing tools and proposing a natural means of interaction with users. Thus, they answer our requirements concerning multiexpert cooperation, knowledge modeling and linking of IP operators. Before building our Knowledge-Based system, the choice of a conceptual model implies a thorough study of the various kinds of knowledge that is needed.

Our system has three knowledge levels: domain knowledge, control knowledge and metacontrol, the first level being decomposed into three categories:

- knowledge about the application, to give a meaning and a subject to an image. This knowledge is mainly given at the beginning of the resolution but can also be provided in case of ambiguity or incompleteness,
- IP knowledge to determine the strategies to be used to perform a task and evaluate results,
- knowledge on operators to select them, tune their parameters and form syntactically correct sequences of operators.

Control knowledge is knowledge that facilitates the management of domain knowledge. It is dealing with the methodology followed to solve the application and its explicit representation. Two categories can be distinguished:

- domain control dealing with the management of IP plans (creation, modification, execution, ...),
- system control dealing with the visualization of operations, the selection of the treatments accessible to the user, the control of sequences of operations,....

Finally, metacontrol knowledge define the behavioral rules of the system with regards to users. The control must be adapted to the user (one cannot propose the same activities to an expert and a beginner) and his/her wishes (automatic or step-by-step mode, explanations and assistance during a session, ...).

We have chosen to create an interactive environment favoring the incremental development of applications. This environment is the place for the acquisition of new knowledge. Integrating new knowledge in the system is done along two axes: *procedural integration* to achieve tasks and *semantic integration* to make them explicit. On the one hand, we want to collect domain and control knowledge (and may be metacontrol knowledge) in an independent but similar way. But we must also account for the fact that an IP request has no unique solution: there generally exist several techniques to satisfy it. It is in this multiplicity of solutions that the integration of ontological and structural knowledge takes all its importance. Thanks to this knowledge, the user will be in a position to decide when a technique should be preferred to the others.

Our study of the problems set by the realization of IP applications, as well as our desire to have the three types of experts cooperate, have led us to choose an architecture answering these requirements: the TASK/METHOD/TOOL architecture.

4 - Conceptual model: the TASK/METHOD/TOOL architecture

As indicated by its name, the TASK/METHOD/TOOL is based on three notions we are now going to detail.

A **task** is the representation of a goal or a sub goal in the system. It describes this goal together with information necessary to reach it: input data, type of results, resolution methods. When a task describes a general problem, it is decomposed into sub tasks describing more elementary problems, these sub tasks being in turn decomposable. As a task can be solved in several ways, one or more methods are associated to it.

Tasks can represent all the goals of the system, whatever their level:

- *domain* level: tasks represent an IP objective or some part of it (e.g.: extract clusters of cells or eliminate the background of the image).
- *control of the domain* level: tasks model operations to be performed on IP plans (e.g.: execute a plan, store a plan). They can either be executed by the system or by the user.
- *control of the system* level : they regroup tasks controlling the good progress of a work session (e.g.: initialize system, control the interface). Only the system can execute them.
- *metacontrol* level: tasks represent control on control operations, i.e. metatasks (e.g.: execute a task, choose a method).

Thanks to our uniform model of tasks at all levels, we can say that the triplet {metacontrol, {control, domain}} defines a reflexive system, which provides richness, flexiblity and adaptability to our model.

A **method** specifies how to perform a task. Each method is associated to a single task but a task can be associated to several methods. (fig. 1-a). When modeling a task, one has to tell when it is possible and desirable to use it. So, although in some contexts, a method can be triggered because all the data necessary to its execution are present, it can be judged irrelevant and unsuited to the current problem. The choice of a method can be done either by the user (step-by-step mode), or by the system (automatic mode).

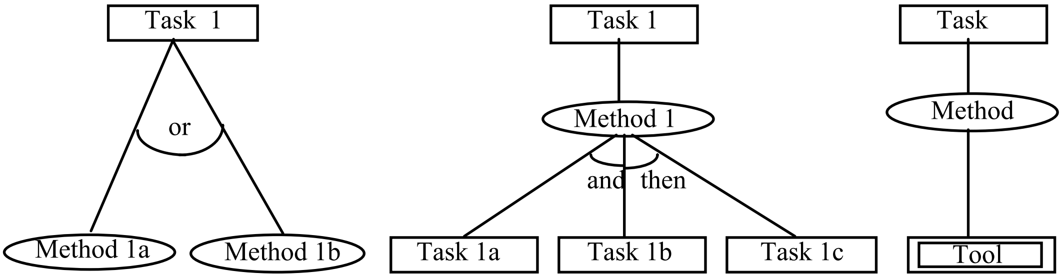
The body of a method can take two forms:

- a decomposition into sub tasks described as an AND/THEN tree, in the case of a complex method (fig. 1 b).
- the call to a run-time program through the medium of a tool in the case of a terminal method (fig. 1-c).

We have defined the concept of **tool** for reifying computer codes, i.e. in our case mainly IP operators. A tool models a computer code by describing its goal, inputs, parameters, outputs, syntax of call, performances, resources ... For the user, the code associated to a tool can be seen as a black box: he/she only has to know the transformation performed on inputs to produce outputs.

Three executing modes are available: *normal* mode (simple execution of the operator), *loop-for* mode (execution of the operator a fixed number of times) and *optimization* mode to dynamically calculate the values of parameters on the basis of measures made directly on images. The latter mode correspond to classical loop constructs (while, until, best-before ...) and uses an evaluation function to check the relevance of the result with regards to desired criteria. We are here inspired by [Matsuyama 89] and [Clément 93]. Our representation of control loops and the fact that the source code is available enables us to generate the C++ program corresponding to a complete plan.

When wishing to perform some processing on an image, the user first has to select the corresponding task in a menu. Secondly, he/she is asked to enter parameters and input data for the plan corresponding to this task. The system can then execute the task by proposing choices among existing methods (in the *step-by-step* mode) or by selecting by itself the most relevant method according to the context (in the *automatic* mode). If results are unsatisfactory or even non existing, the system proposes or does itself a backtrack on these choices.



(1-a) (1-b) (1-c)

Figure 1: Various possible organizations of tasks, methods and tools
(1-a): Several methods can be associated to a task
(1-b): Task / sub task decomposition (complex method)
(1-c): Method calling a tool (terminal method).

5- Validation of the approach and future prospects

The first prototype of our workbench for manual construction of plans and their execution is implemented in CLOS (Common Lisp Object System). Several Image Processing plans dealing with biomedical (cytological / histological) images provided by the cancer-research center of Caen have already been built and integrated within this environment, which constitutes a first convincing validation of our approach.

In the case of cytological images, we have built manually a plan to solve a task of detection of nuclear cells. This plan was also built automatically by our planning module BORG [Clouard 95a]. We could thus show the compatibility of the two aspects of our workbench: manual versus automatic construction and execution of IP plans.

In the case of histological images, we studied the problem of detection of tumoral clusters of nuclei. The fact that we modeled the analysis methodology as a TASK/METHOD/TOOL plan, rather than just linking operators, brought new ideas into light, so as to improve some sub tasks (e.g.: replacement of a sub task for computing distances of graphs by another more) and it also gave us a more global view of the whole process.

The ease of coding these plans, and the rapidity of their integration into our environment, are reinforcing our conviction that our approach is promising. While continuing our knowledge acquisition process by integrating new IP plans, we are also studying how to reinforce man/machine cooperation between experts and the system. Recent works about the role of cooperative agents in Knowledge-Based systems [Monclar 96] [Hadj Kacem 96] are particularly interesting.

In order to improve cooperation, a graphical interface (fig. 2) adapted to the various types of users is being developed, including a plan-editor for the visualization and manipulation of the tasks, methods and tools of a plan.

Then, we shall have to imagine how to establish cooperation between the two aspects of our workbench: manual construction and automatic construction of IP plans by BORG, in order to solve more and more complex image processing and image understanding problems.

References

[Bodington 95] Bodington R., "*A software environment for the automatic configuration of inspection systems*", KBUP'95, Sophia Antipolis, France, novembre 1995, p100-108.

[Clément 93] Clément V. & Thonnat M., "*A knowledge-based approach to integration of image processing procedures*", CVGIP : Image Understanding, Vol. 57 (2), Academic Press, New York, USA, 1993, p164-184.

[Clouard 95a] Clouard R., Revenu M., Elmoataz A. & Porquet C., "*A software workbench for knowledge acquisition and integration in image processing*", International Workshop on the Design of Cooperative Systems, Juan-les-Pins, France, janvier 1995, p298-313.

[Clouard 95b] Clouard R., Porquet C., Elmoataz A. & Revenu M., "*Why building knowledge-based image segmentation systems is so difficult?*", KBUP'95, Sophia Antipolis, France, novembre 1995, p137-148.

[Delouis 93] Delouis I., "*LISA, un langage réflexif pour la modélisation du contrôle dans les systèmes à base de connaissances, application à la planification des réseaux électriques*", Thèse de doctorat, Université de Paris-sud, Centre d'Orsay, France, 1993.

[Hadj Kacem 96] Hadj Kacem A., "*Les SBC coopératifs, vers une architecture fondée sur une formalisation du contrôle*", RFIA'96, Rennes, France, janvier 1996, p64-72.

[Matsuyama 89] Matsuyama T., "*Expert systems for image processing : knowledge-based composition of image processes*", CVGIP, Academic Press, New York, USA, Vol. 48, 1989, p22-49.

[Monclar 96] Monclar F.R., "*ELICO pour le développement de systèmes à base de connaissances coopératifs*", RFIA'96, Rennes, France, janvier 1996, p73-82.

[Newell 82] Newell A., "*The knowledge level*", Artificial Intelligence, n°18, 1982, p87-127.

[Revenu 95] Revenu M., Clouard R., Elmoataz A. & Porquet C., "*Un atelier logiciel d'acquisition et d'intégration de connaissances en traitement d'images : une approche méthodologique*", 10è Journées Acquisition, Validation et Apprentissage, Grenoble, France, avril 1995, p315-328.

[Thonnat 95] Thonnat M. & Moisan S., "*Knowledge-based systems for program supervision*", KBUP'95, Sophia Antipolis, France, novembre 1995, p4-8.

[Wielinga 92] Wielinga B.J., Schreiber A.T. & Breuker J.A., "*KADS: a modelling approach to knowledge engineering*", Knowledge Acquisition, vol. 4, 1992, p5-53.

[Willamowski 94] Willamowski. J., "*Modélisation de tâches pour la résolution de problèmes en coopération système-utilisateur*", RFIA'94, Paris, France, janvier 1994, p305-316.

Figure 2: Hard copy of the graphical interface showing windows to create

tasks, methods and tools and to execute a plan