Knowledge Management in Image Processing by Means of Case-Based Reasoning

Valérie Ficet-Cauchard, Marinette Revenu & Christine Porquet

GREYC-ISMRA - 6 bd du Maréchal Juin - F14050 Caen cedex FRANCE

Valerie.Ficet@greyc.ismra.fr

1. Introduction

Our research takes place within an image team where Image Processing (IP) experts are working on various types of IP problems (restoration, segmentation, feature extraction, …) and on images from various origins (MRI: Magnetic Resonance Imaging, cytology, histology, industry, …). We are concerned with issues such as the acquisition, modelling and capitalisation of the knowledge used by those experts, with a view to help them reuse and share such knowledge. To achieve this objective, we are advocating for an interactive system that enables each IP expert to represent and memorise his/her know-how on the one hand, and to reuse the whole set of knowledge stored into the system by all experts on the other hand.

So as the knowledge acquisition step be as user-friendly as possible, the IP expert is required to represent his/her know-how in the very course of application development. Thus, knowledge can progressively be integrated as it is discovered and any new application that is represented in the system can be executed at once. The implementation of knowledge reuse is achieved by means of Case-Based Reasoning (CBR) techniques.

Capitalising the knowledge of our different IP experts into a unique system is a major challenge for two main reasons. On the one hand, it enables experts to share their knowledge, thus avoiding to focus on the same techniques, as is often the case in IP. On the other hand, the system can act as an “educational” tool when new researchers join the team, by offering them to browse through the whole set of strategies that have been represented, together with examples of previously solved problems.

The issues raised by this kind of approach are manifold. First, the model used for knowledge representation must be easy to understand and completely explicit, in order to ease expert/machine and expert/expert interactions. Secondly, all modelled applications must be understandable by any potential system’s user. They must thus include semantic knowledge and be described in generic terms that are not application specific. Finally, reuse can only be considered if users are given a means to retrieve all the knowledge they need at any moment. Applications must consequently be searched and retrieved thanks to a set of discriminative criteria.

We first start with describing the context of our research through an experimental example, so as to justify our choices regarding models and techniques (section 2). Then our objectives are presented, the system’s architecture is given (section 3) and the two major system’s modules are detailed (section 4). Finally, the approach followed for the system’s validation is explained and discussed.

2. Objectives and preliminary choices

In IP, the tuning of an application is a complex activity that involves two categories of experts (domain expert and IP expert) and can be broadly decomposed into three stages: formal definition of the problem, working out of a solution, and result assessment.

The knowledge that is used all along this process is difficult to be made explicit. The know-how that each expert has accumulated is non-structured and extremely diffuse. There exists no theory to back the domain expertise, which compels experts to act in a relatively empirical way. They are often enable to make all reasoning steps explicit. Two explanations can account for such obstacles. First, problems are generally given in terms of the application domain, and thus have to be translated into IP terms; moreover they are seldom completely specified from the beginning. Secondly, knowledge about the effects of IP algorithms is imprecise and difficult to make the most of; either is there any quantitative function for result assessment. Consequently, many trial-and-error steps must be considered, together with optimisation ones. Initial objectives often have to be revised and completed in the course of the resolution, which involves both categories of experts.

The domain expert plays an important part during the development of the application. First, he/she has to define the problem, for which the IP expert is intended to build a solution i.e. an IP program; the program's inputs are one or several images (from which "objects" must be extracted) and a request (describing the goals to be reached). Once a first solution has been built by the IP expert, the domain expert is asked to evaluate it by visualising output images. As IP is a domain where no ideal evaluation function exists, only the domain expert can judge the relevance of the final result. This validation can either be done visually, or by using results coming from a broader testing protocol (e.g. statistics about the objects extracted from input images). The roles of the domain expert and the IP expert during the development of an application are thus interweaved, which implies co-operation and communication requirements between both of them.

Besides, the multiple and shaky nature of solutions can make the tuning of applications even more difficult. For a given problem, there often exist several solutions that are more or less adapted to input images, and more or less sensitive to their type and amount of noise. To address such issues, one often has to introduce a priori knowledge about expected results (semantic description of the image or of the objects to be looked for) and several IP strategies have to be tried and tested.

In the rest of this section, we are going to explain how our applications are represented, then give a concrete example that well accounts for the relevance of our approach and explain our technical choices regarding the reuse of applications.

2.1. Plan-based representation

The type of approach we are advocating for the development of IP applications is based on the "smart" supervision of libraries of programs, that are called IP operators. The building of an application here consists in chaining and tuning a set of operators from a given library. This approach is more and more widespread in IP, on the one hand because it enables an IP expert to built his/her application without bothering about programming details (an IP expert is not always a computer "wizard"), and on the other hand because the algorithms

implemented as operators are the only knowledge, for which there exists a consensus in the IP community.

Such a design mode based on the “assembling of building blocks” is often used in Knowledge-Based Systems. Building blocks can correspond to “primitive inferences” (actions that can be applied to domain objects) in CommonKADS [Breuker 94] or to Chandrasekaran’s “generic tasks” [Chandrasekaran 87].

As IP operators generally are of relatively small grain-size, an actual application can often result in the chaining of several dozens of operators. In order to make explicit the reasoning used during the development of such sequences, our approach consists in representing applications by means of plans.

An IP plan is formed by the hierarchical decomposition of a given problem into more simple sub-problems. Each problem or sub-problem is associated to one IP task, which, according to its level within the plan, either corresponds to a goal to be solved, a technique to be used or an algorithm to be applied. Such a modelling by the means of generic and decomposable tasks is a widespread approach in problem-solving [Jacob-Delouis 96] [Talon 96] [Steels 91] , but is seldom used in IP. On the one hand, the representation of strategic knowledge at several levels of abstraction provides a first way to make the reasoning explicit; on the other hand, the task concept can be easily understood by users and groups together procedural and semantic knowledge, which makes man/machine communication easier.

An IP plan can be schematised as a tree (fig.1). Not only does it represent the sequence of execution of IP operators, corresponding to the leaves of the tree, but also all the additional reasoning implied in the creation of such a sequence, corresponding to IP tasks visualised as grey rectangles.

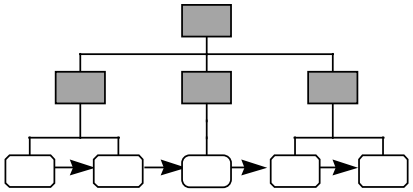


Fig.1 : representation of an IP plan

In the next paragraph, we are going to show how worthwhile it is to use hierarchical plans for sharing and comparing strategies.

2.2. Example showing the importance of modelling strategies

In this paragraph, an example of strategy reuse is shown, corresponding to two independent applications that have been finalised within our research team. The first problem deals with detection and localisation of cortical sulci in MRI, and the second one with the localisation of cephalometric landmarks in skull radiographs. Here, we are only concerned with the first stage of each application, which consists, for each sulcus/landmark, to find out a search area, in which the sulcus/landmark must be looked for, according to a learning base of images that have been manually segmented by a domain expert. For the automatic recognition of sulci, this first stage is done in three main steps:

- determine the Talairach reference system (R) on the input image: this reference system, calculated from anatomical data, enables to get a common reference system for comparing images of the learning base,
- register all sulci from the images of the learning base onto the input image, by the means of reference system (R),
- for each sulcus, determine a search area, in which the sulcus will be looked for, according to the knowledge about the registered sulci.

Representing the technique used for the determination of the search area as a tree of tasks, is a good means to cut off oneself from the domain of application, so that a more general image analysis strategy can emerge (fig.2).

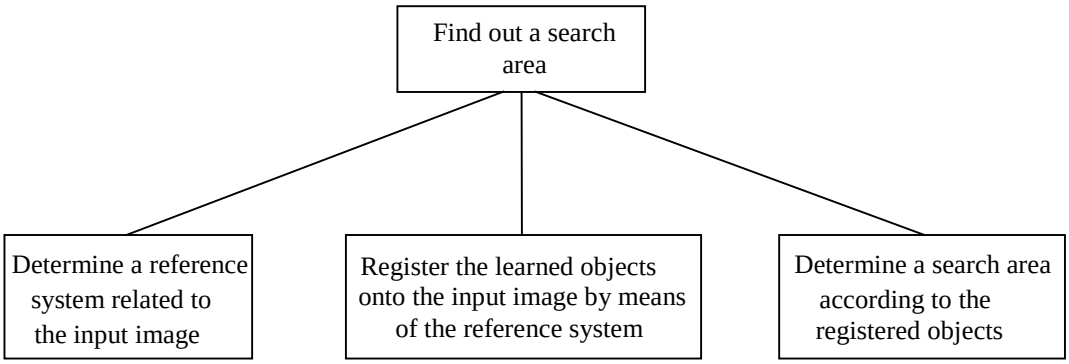


Fig.2 : Representation of a strategy for the determination of a search area

The localisation of cephalometric landmarks is quite similar to the localisation of cortical sulci: in both cases, one has to localise specific anatomical structures that are characterised by a rather good stability of localisation. One also has the same kind of a priori data: existence of a base of images that have been manually treated by a domain expert, and knowledge about anatomical invariant structures. We have thus tried to reuse the strategy of the first application for the second one.

In the case of the localisation of cephalometric landmarks on skull radiographs, we now get the three following steps:

- determine a reference system (R) on the input image by using anatomical knowledge,
- register all landmarks from the images of the learning base onto the input image, by the means of reference system (R),
- for each landmark, determine a search area, in which the landmark will be looked for, according to the knowledge about the registered landmarks.

This example shows the importance of representing strategies, in order to favour knowledge reuse and dialogue between experts.

2.3. CBR and assistance to reuse

Reusing plans or parts of plans for building new applications has two advantages for an IP developer:

- Saving time: the tuning of an application is a long and complex chore, and reuse can reduce the time spent for some steps (whether it is for the choice of a general strategy or for a technique applied to a very specific part of the application).
- Sharing knowledge and experiments: reuse may concern a complete application created by another user, or only parts of the expertise corresponding to another domain of application.

To implement that kind of reuse, one can consider a CBR approach. The idea is to reason on past experiments or previously-solved cases to solve new problems. As a matter of fact, we, human beings often try to solve a problem by recalling how a similar problem was previously solved: we will diagnose a breakdown or a disease because of symptoms that draw attention to the breakdown or the disease, we will build a work-plan because a similar plan gave good results in the same circumstances. When, we, human beings, use this style of reasoning, our limited and selective memory does not always allow us to recall the most appropriate cases. CBR can address such an issue by providing means to memorise all cases and retrieve the best adapted ones. Besides, a user can not only retrieve its own cases, but also those solved by others. The advantages of CBR in domains characterised by a weak or ill-structured theory, such as the IP domain are manifold:

- representing exceptions,
- making the most of missing or noisy data,
- solving a complex problem, through interactions between solutions of more simple problems,
- dynamic learning,
- using the case base for educational purposes.

By using the same kind of reasoning as CBR automatic systems, CBR interactive systems [Kolodner 91] can provide assistance to users by increasing their memory and selecting the nearest cases to the current problem. But contrary to CBR automatic systems, the final decision is left to users, i.e. select the most appropriate case and decide whether adaptation is needed or not. This kind of assistance gives priority to interactivity, while helping users all along their work.

3. System's architecture

The overall architecture of our system is shown on figure 3. This system is mainly dedicated to an IP expert that wants to solve an IP problem in a specific domain of application. Several functionalities are presently available:

- interactive creation of an IP application,
- storage of applications into a base of IP plans,
- interactive execution of an IP application,
- creation of an IP application by a CBR approach [Ficet-Cauchard 98].

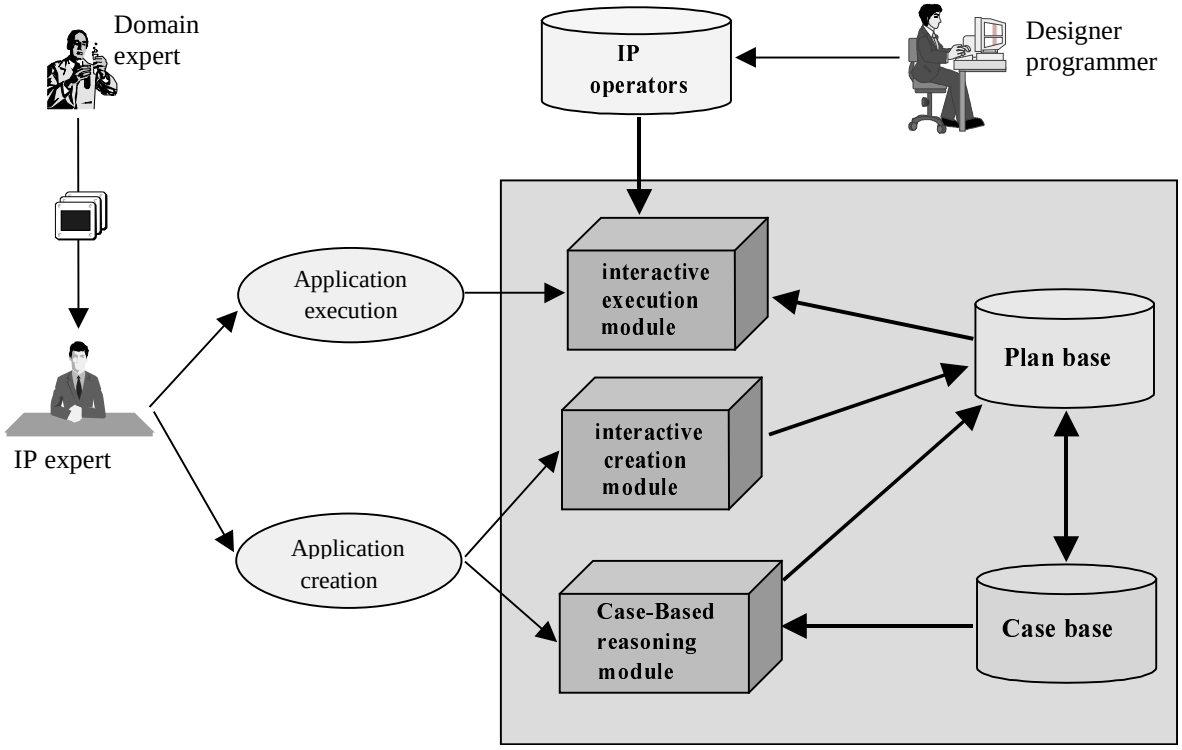


Fig.3 : overall architecture of the system

Once images and a request have been given by the domain expert, the IP expert can build step by step an IP plan with the interactive creation module. With our system, he/she no longer has to program in the classical sense of the term: he/she is programming by means of what Van de Velde calls “KL-models” [Van de Velde 93], i.e. by using predefined building blocks. KL-models are inspired by Newell’s “knowledge level” [Newell 82], which accounts for the “why” aspects of the system behaviour, but they also include structures such as tasks and methods for the description of its “what” and “how” aspects. Contrary to classical graphical IP programming environments [Iris 93] [Rasure 94], in which users are compelled to adopt a bottom-up approach when building their plan, in our system, one can choose between a bottom-up strategy (by grouping operations together), top-down strategy (by decomposing problems into sub-problems) or a mixed one. The latter well corresponds to the IP developer approach, who will willingly proceed in a bottom-up manner for parts of treatments that he/she knows well, and in a top-down matter for others. It is also possible to represent several techniques that solve one given problem or sub-problem, so as to later be in position to test and compare the different solutions. Applications are stored into a base of plans, in the course of their development.

Plan execution is achieved by dynamically choosing between the available techniques, and thus evaluating results in co-operation with the domain expert. Then, it is possible to modify the plan as long as results are unsatisfactory. The representation of plans by means of decomposable tasks makes this modification stage easier, in particular, by providing a communication medium between experts and by authorising the visualisation of intermediate results.

When the user is satisfied with his/her plan, he/she can define the cases associated to the different problems and sub-problems that are solved by the new plan, so as to further reuse his/her work. New applications can also be created thanks to the CBR module. This module provides assistance for the retrieval of the nearest cases to the current problem and during the adaptation stage. Adapting a solution consists in replacing some parts of a plan by parts of other plans, and by performing local modifications thanks to the interactive creation module.

On figure 3, the designer-programmer is also shown. An IP expert that has insufficient programming competencies can appeal to a designer-programmer for the implementation of new IP operators that must be integrated into the library of operators. It is also the designer-programmer that is in charge of implementing the various functionalities of the system. Each functionality corresponds to a control task (i.e. a task that controls domain knowledge). To enable further evolutions, control knowledge is also represented as trees of tasks. Thus, a user that has the requisite competencies can define new functionalities, exactly in the same way as the IP expert defines new applications. For instance, a designer-programmer may define the control plans necessary to the reuse module by assembling basic blocks.

4. Major modules of the system

The implementation of our system was done in two stages: 1. construction of a system for the interactive creation and execution of IP applications, called the TMT (Task-Method-Tool) system; 2. integration into the TMT system of a CBR module. In this section, we first describe our knowledge representation and the functionalities of the TMT system. Then, the CBR module that provides assistance to application reuse is detailed.

4.1. The TMT system

After a presentation of the different knowledge levels of the system's architecture and of the TMT model used for knowledge representation, the functionalities for the interactive construction and the interactive execution of IP plans thanks to a graphical interface are described.

4.1.1. Architecture and model

In our system, three knowledge levels are defined: the domain level (Image Processing), and the control and metacontrol levels that manage the representation and use of domain knowledge.

Domain knowledge includes knowledge about the application (to describe a priori information about image domain or expected results), knowledge about IP (to find out which strategies to carry out) and knowledge about operators (to select them, tune their parameters or form syntactically correct sequences of operators, …).

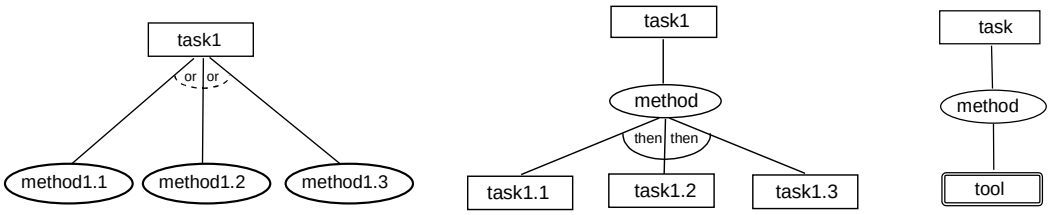
Control knowledge is in charge of managing domain knowledge (plan creation, plan execution, …) and controlling the system behaviour (visualisation of the operations to perform, selection of the treatments accessible to users, supervision of sequences of operators, …).

Metacontrol knowledge is knowledge about how to control the system's control and defines the system behavioural rules with regards to users.

In our system, the all three levels are uniformly represented thanks to the "TASK – METHOD – TOOL" (TMT) architecture.

A TASK represents a goal or sub-goal and contains all elements that are necessary to satisfy this goal (type of data, type of result, solving methods, ...). When a task solves a general-purpose problem, it is defined as a decomposition of sub-tasks that solve more elementary problems, and each sub-task can further be decomposed into simpler ones. When a task can be performed in several ways, it is associated with several methods (fig. 4-a).

A method specifies how to perform a task. Each method is associated with a single task, but a task can be solved by several methods. So one has to tell, for each method, when it can be used. The method body can take two forms: either a decomposition into sub-tasks (fig. 4-b), described as a "THEN" tree - which is called a complex method, or the call to a run-time program through the medium of a tool (fig. 4-c) - which is called a terminal method. The set of sub-tasks that will be executed to perform a given task depends on the selected method; so it is methods that manage task/sub-task and task/tool data flows.



a) b) c)

Fig. 4 : various possible links between TASKS, METHODS AND TOOLS

A tool reifies a computer code, which means it is a user-oriented representation of this code in conceptual terms (goal, inputs, parameters, outputs, syntax of call, performances, resources, ...) with a link to this code enabling to run it. From a user's point of view, this code can be seen as a black box: all that is known about it is the nature of transforms performed on inputs to produce outputs and the only possible action is the tuning of parameters. At the domain level, tools mainly reify basic IP operators of the PANDORE library [Clouard 97] (e.g. image thresholding, image smoothing, ...), but they can reify external Lisp or C functions likewise.

4.1.2. Graphical interface

Our knowledge-based system is provided with a graphical interface favouring man/machine co-operation for plan creation and execution.

When wishing to develop a new application, users can build their tasks and tools by filling in fields in adapted windows. Then, tasks and tools are linked by means of methods. The task associated with a method is selected in a menu, as well as its associated sub-tasks or tool. All components are automatically stored in files and data flows are defined in a user-friendly way by drawing lines between elements on a diagrammatic figure of the current plan.

To perform some processing on an image, users have to select a task in a menu; the corresponding plan is then instantiated when that image is given as input to the plan. The task is executed by proposing a choice between existing methods, when several methods exist to

solve one task, and asking for parameter values if necessary. During execution, users can visualise intermediate images that are input or output of any task of the plan, or have access to data about tasks and solving methods, through the graphical figure of the plan as a tree of tasks.

4.2. CBR module

Integrating a case base and a CBR module into the TMT system is intended to provide additional assistance to the IP experts, by enabling them to reuse applications developed by the various system's users. In this section, after a description of our case representation and our selection criteria, the similarity functions for comparing cases are defined and the reuse process is detailed.

4.2.1. Case representation

A case is broadly composed of two parts: description of the solution and description of the problem. In our system, a solution is represented as a TMT tree, which can be accessed through its root task. In Case-Based Planning [Veloso 96] [Prasad 95] or Case-Based design [Smyth 96], a solution is generally built by combining parts of several plans coming from several cases. In order to make this kind of design possible, we have decided to associate several cases to one single plan: the first case is associated to the root task of the plan, the others to some sub-tasks of the same plan, that are considered as representative of specific IP techniques.

The problem's description is made thanks to a set of discriminative criteria, which have been found out from a thorough study of the IP domain. The major issue of this study was to choose a vocabulary that can be shared and accepted by any IP expert. The criteria we put forward come from a classification of the most often encountered terms used to describe IP actions and data. We have made a distinction between two broad categories of criteria: criteria related to the task definition and criteria related to the image description.

Criteria related to the task definition

This first category includes data related to the operation performed by a task and to its position in the plan in relation to other tasks. Such criteria include IP type or phase, problem definition and abstraction level.

According to the task's abstraction level, one can take into account either the IP **type** (*detection*, *segmentation, classification*…), or the IP **phase** (if the IP type is segmentation: *pre-processing*, *seed determination*, *region determination,* …).

The various IP phases correspond to a vertical division of the plan for one type of problem. Some phases may be optional.

The **definition of the problem** is composed of a set of keywords, independent from any domain of application, selected among three pre-defined lists: 1. a list of **verbs** describing the operations performed by the task (*detect*, *classify*, *binarise, smooth,* …), 2. a list of **nouns** corresponding, either to objects on which the action is performed (*contours*, *regions, image background,* …), or to IP techniques (*region growing, region division,* …) and 3. a list of

adjectives qualifying, either the objects on which the action is performed *(small, local, …)*, or the action itself (*partial*, *strong, …*).

Finally the **abstraction levels** that correspond to a horizontal division of the plan are based on the abstraction levels of the automatic planner BORG [Clouard 99]. The first level is the *intentional level*. Tasks of this level answer question such as "what to do ?" and deal with IP objectives. The second level is the *functional level.* Tasks of this level answer questions such as "how to do ?" and refer to some IP technique, leaving aside technical constraints related to their implementation. The third and last level is the *operational level.* Tasks of this level answer questions such as "by means of what ?" and represent IP technical know-how that can be implemented as algorithms.

Criteria related to the image description

Among the criteria related to the context of images, some correspond to physical knowledge (related to image formation) and describe image quality (e.g. **type of noise**, **amount of noise** and **quality of contrast**). These criteria are of paramount importance for the choice of the pre-processing steps.

Other criteria rather correspond to perceptual knowledge (symbolic description in terms of visual primitives). They include the **presence** or **absence** of an image **background** and the **aspect of objects** (*homogeneous grey level, light colour, texture, thick boundaries,…*).

The third group of criteria correspond to semantic knowledge (scene analysis and components of the scene) and describe the appearance of what is to be detected, but in abstract terms, independent from the domain of application. These latter criteria include the **form** of objects (*convex*, *concave*, *elongated*, *compact*, *square, round*, …), the **relative size of objects**, their **position** (*left, middle, right, top, bottom, centre)* and **inter-object relations** (*proximity, connectivity, inclusion, …*).

4.2.2. Similarity calculation

Our first group of criteria (i.e. criteria related the task definition) are here to characterise the action performed by a task, and are thus closely dependent on the TMT model. Such criteria define a set of tasks that can solve one "type of problem". They are "compulsory" (each criterion of the target case must have a value) and are used to reduce the search space. A first similarity function Φ_t using the criteria related to the task definition will thus be applied to reduce the set of candidate target cases. This function is defined by formula (1) as the weighted average of the similarity results for each criterion: S is the source case, T is the target case, α_{Cr} is the importance coefficient associated to criterion Cr and $\varphi_{Cr}(S,T)$ is the similarity between S et T related to criterion Cr. The result value of any φ_{Cr} function is between 0 (if values of Cr between both cases are very different from each other) and 1 (when they are deemed identical). All α_{Cr} coefficients are also comprised between 0 and 1, in order to normalise the Φ_t function to return values between 0 and 1.

$$\Phi_t(S,T) = \frac{\sum \alpha_{Cr} \times \varphi_{Cr}(S,T)}{\sum \alpha_{Cr}} / Cr \in \{ criteria\, related\, to\, the\, task\, definition \} \qquad (1)$$

The second group of criteria (i.e. criteria related to the context of images), characterise the objects to be detected and depend on the current image. Such criteria are not meaningful for any application: for instance, contrast quality has no sense when processing a region map. This second group of criteria are "optional" ones (all criteria of the target case need not be filled in); they enable to select the nearest cases among the candidates obtained after applying function Φ_t. The second similarity function Φ_i is thus used to reduce the set of selected cases, in order to get a list of reasonable size (that can be presented to the user). This function is defined by formula (2) as the weighted average of similarity results on each criterion; notations and properties are the same as in formula (1).

$$\Phi_i(S,T) = \frac{\sum \alpha_{Cr} \times \varphi_{Cr}(S,T)}{\sum \alpha_{Cr}} / Cr \in \{ criteria related to the image description \} \qquad (2)$$

It is clear that the list of criteria related to the context of images cannot be exhaustive: the criteria we put forward are coming from our study of IP literature and from the development of our own applications. In order to easily integrate new criteria, we have defined criterion types. Each criterion type is associated to a generic similarity function corresponding to comparison mode. Figure 5 shows a table summarising all criteria currently in use, with their type, importance coefficient, value range and value example.

Criterion	Coef.	Type	Value range	Value example
problem definition (verbs)	0,4	multi-valued	Set of possible verbs	{extract, separate}
problem definition (nouns)	0,3	multi-valued	Set of possible nouns	{object, grouping}
problem definition (adjectives)	0,3	multi-valued	Set of possible adjectives	{little}
type or phase	1	strict symbolical	Set of possible phases and types	segmentation
abstraction level	1	strict symbolical	{operational, functional, intentional}	intentional
type of noise	0 to 1	strict symbolical	Set of possible types of noise	gaussian
amount of noise	0 to 1	gradual symbolical	{null, very-low, low, medium, strong, very-strong}	low
quality of contrast	0 to 1	gradual symbolical	{very-low, low, medium, strong, very-strong }	medium
presence of background	0 to 1	strict symbolical	{yes, no}	yes
aspect of objects	0 to 1	multi-valued	Set of terms describing aspect	{homogeneous grey level, light grey }
form of objects	0 to 1	multi-valued	Set of terms describing form	{convex, round}
size of objects	0 to 1	gradual numerical	[1, 5]	3
position of objects	0 to 1	multi-valued	{left, center, right, low, middle, high}	{center, middle}
inter-objects relations	0 to 1	multi-valued	Set of terms describing relation	{connectivity}

Fig.5 : schema of our selection/adaptation process

Besides, as it is the case in ISAC [Bonzano 97], all criteria need not be taken into account in any application. A missing criterion value should be taken into account in different ways

whether one considers the source case or the target case: the absence of value in a target case will have no consequence on similarity calculation (criterion is irrelevant for this case), the absence of value in a source case should lower the result of similarity calculation (condition is not respected).

4.2.3. The reuse process

Our CBR module must allow users to build an IP plan from their current problem defined by the set of criteria given in the previous section. Contrary to most CBR planning systems [Prasad 95] [Smyth 96] [Veloso 96], which proceed by progressive refinement of an abstract plan, in our case, each tuning step produces a complete plan that can be tested and assessed. This choice of ours is due, on the one hand to our will to develop a conversational system and not a completely automatic problem-solver, and on the other hand, to issues raised by result assessment of IP applications (there exist no general function for comparing produced results with desired ones). One can thus very rapidly obtain a solution that will serve as a starting point for the IP expert and the only evaluation method that can generally be applied to any IP application is used: visual evaluation of output images by the expert.

In the selection/adaptation process of most CBR systems, one can notice, on the one hand, the existence of a preliminary step in the selection process, aiming at reducing the search space [Netten 97] [Bonzano 97], and on the other hand, the fact that the selection/adaptation cycle must be applied iteratively, in particular in CBR planning [Prasad 95] [Smyth 96]. Our approach (fig. 6) is also based on a selection/adaptation cycle, iteratively applied at various levels of the plan, but in addition, at each cycle loop, a reduction step of the search space has been included.

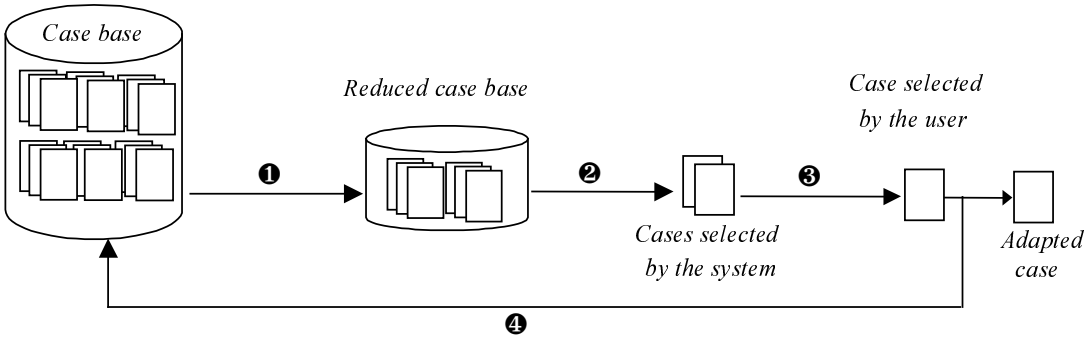


Fig.6 : schema of our selection/adaptation process

The reduction of the search space can be achieved, either by using criteria corresponding to strict constraints, or by considering that two cases can only be compared when defined by the same set of criteria. The latter technique is not adapted to our domain. As a matter of fact, among the criteria related to the context of images, some of them bring nothing new about the target case, without disqualifying the source case

The reduction step (❶) can thus be achieved by means of function Φ_t using the "compulsory" criteria related to the task definition, while the selection step (❷) will use function Φ_i with the "optional" criteria related to the image description. The user is responsible for the selection of the best source case (step ❸). Finally, step ❹ consists in adapting the plan associated to the selected source case to the current problem.

Principles for selection, adaptation an memorisation of cases used in our algorithm are detailed in the rest of this section.

Selection of a source case

In the course of step ❶, the reduction of the search space consists in selecting source cases S that solve the same type of problem as target case T. It corresponds to a selection of cases *S* such that Φ_t (*S*,T) > α_t where α_t is a threshold fixed beforehand. The weights of each criterion in function Φ_t are also fixed: the same importance is granted to all criteria. This step provides a first set of cases S.

So that the user can choose a case at step ❸, the set of cases resulting from step ❷ must be of reasonable size large (by default between 2 and 5 cases). If the set is too small, the user's choice will loose importance, and if it is too large, the user's choice will be difficult. The iterative nature of step ❷ enables to get a set whose size can be shown to the user as a list. A relaxation process is used to increase or decrease constraints at each iteration. As the final choice is left to the IP expert, the intuitive character of his/her working habits can be kept.

Interactive plan adaptation

Case adaptation by means of parts of other cases is particularly important in CBR planning. In our system, a case can be adapted at several levels and in several ways: locally or globally, either by means of the CBR module (step ❹), or by means of the interactive creation module.

The plan solution to a case may only require minor local modifications, for instance, an operator should be replaced by another one that better matches the current problem. This first type of modification can be achieved with the interactive creation module, thanks to its modification functionalities.

But a plan may also require broader modifications, i.e. necessitate the replacement of a whole sub-plan by another one. To achieve such modifications, step ❹ offers a means to adapt the solution of the current case by replacing the root task of any sub-plan of the current plan by another task. The substitution task can either be obtained by re-running the algorithm in order to retrieve a similar case, or by building it from scratch, via the interactive creation module.

The recursive nature of the selection/adaptation process is especially interesting because a plan can be adapted whatever its levels within the tree of tasks and as long as necessary.

The memorisation step

Once a new plan is completed, one has to decide whether new cases associated to this plan should be added to the case library.

Several cases associated to one complete plan can be integrated into the base: in fact, if the complete plan represents the solution of a high-level problem, its various sub-plans represent solutions of problems at lower levels. When the integration of a case is required, a first step consists in determining the list of plans and sub-plans that are candidates to integration. This list corresponds to the plans that have been adapted, i.e. the ancestors of replaced sub-plans

that are large enough (at least three levels of tasks). If the substitution plan has been built with the interactive module, it will also be inserted into the list.

Then, for each plan in the list, the user has to provide values for the criteria of the corresponding case that have been modified. The system searches if the new case is different enough from all cases of the base, and, in the event of the answer being yes, integrates it in the case base.

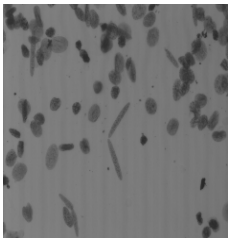
5. System' validation

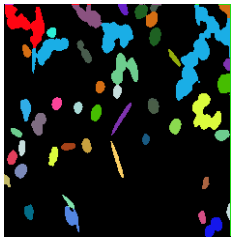
In this section, a few examples of applications developed thanks to the TMT system are presented. The corresponding IP plans served to test our system along three main axes: validation of the model and architecture, experimentation of the interface by an inexperienced user and search for similarities between applications from different fields.

5.1. Validation of model and architecture

Our first two IP plans have enabled to validate the TMT model and the system's architecture by taking into account actual applications. They were developed in our research team by A. Elmoataz [Elmoataz 96] and F. Angot [Angot 99] to process biomedical images of cytology and histology provided by the cancer-research centre F. Baclesse of Caen.

The former (plan A) works on cytological images. The goal is to detect epithelial cells. A precise description of the "objects" that can be found in such images was done in collaboration with a domain expert (e.g. image background is homogeneous and its grey level is lighter than the rest of the image). Plan A returns a region map where the various objects are labelled with different colours. From this region map, it is then possible to do further processing, such as eliminating objects that do no correspond to epithelial cells, according to size, shape or grey level criteria.

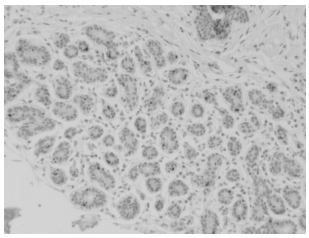


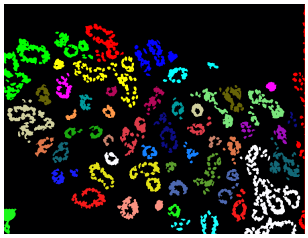


Plan A was the first plan that was integrated into our system. It has enabled to validate the TMT architecture by showing, on the one hand, that the decomposition of tasks into several abstraction levels gave a good representation of strategies and a good medium for dialogue between experts, and on the other hand, that the resulting plan was directly computational (i.e. could be immediately executed). It has also allowed to check the good functioning of the various execution modes of tools (simple execution, multiple execution, execution until a constraint is satisfied).

Besides, it has enabled to define more precisely some functionalities of the graphical interface concerning plan creation and execution. Its integration has also brought to the fore the need for syntactic verifications in the course of the modelling: as a matter of fact, problems due to the absence of syntactic consistency checking, only occur at the time of execution, and it is then difficult to determine what cause them.

The second plan (plan B) was created for an histological application. The goal was to detect significant groups of cells, such groups suggesting the presence of tumoural lobules. The plan returns a region map where each group of cells is labelled with a different colour.





As this second plan was relatively complex, it has enabled to complement and validate our functions for checking syntactic consistency of plans.

The input image of plan B is of the same type as the input image of plan A (same domain: biology, same acquisition device: microscope). The representation of both applications as TMT plans revealed the use of different IP techniques during the first step with corresponds to background "elimination" (use of contrast on boundaries in plan A, and use of inter-class variance in plan B). As a matter of fact, as both techniques can be applied to plan A, we have added a second method to the task "select background" of plan A (fig. 7), this additional method being a technique used in plan B. Such an operation shows, on the one hand, that it can be worthwhile to choose between several methods, in order to try various techniques for the same application, and on the other hand, that the representation of applications as TMT trees can be a medium for knowledge share and discussion between experts.

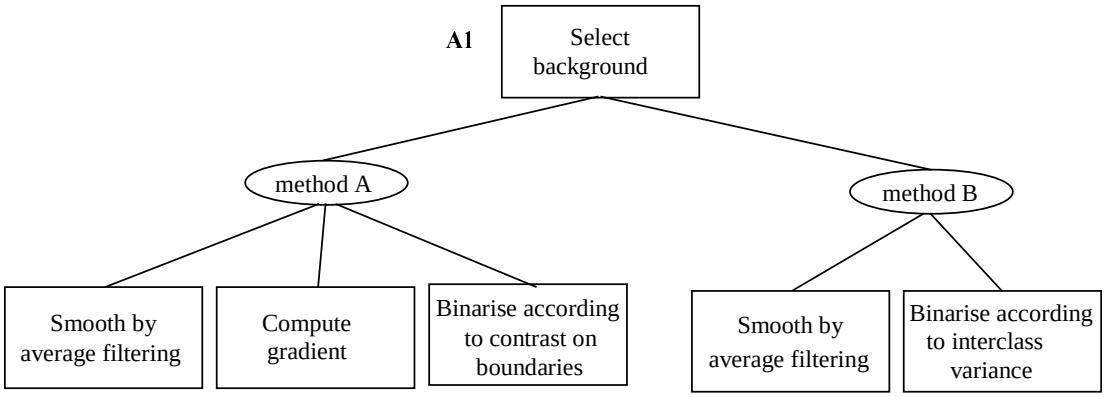


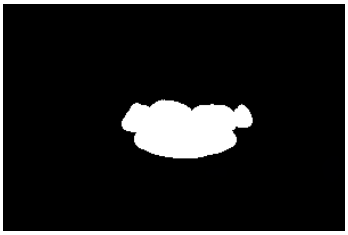
Fig.7 : addition of a second method to a sub-plan of plan A

5.2. Experimentation by a novice

A third plan (plan C) was integrated into the system by a novice who was both inexperienced in the system and in IP. The goal of this experimentation was to test if a new user could rapidly take the system in hand and also to validate the functionalities of the graphical interface.

Plan C was created to deal with images of human faces. The problem was set within the framework of "GDR-PRC ISIS", which is a French research group in Signal and Image Processing. It consists in localising the inner mouth corners. The novice developed three different versions (C1, C2, C3) to achieve this goal. The results presented in this paper correspond to plan C1, which performs the first step of the whole processing, i.e. extraction of

the region corresponding to the teeth, and only works for open mouths. The area to be extracted is defined as a light area situated in the centre of the image.

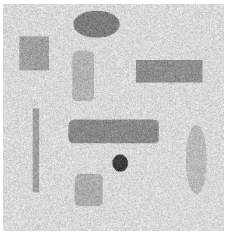
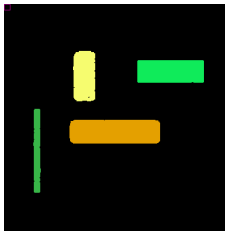
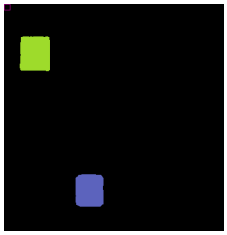
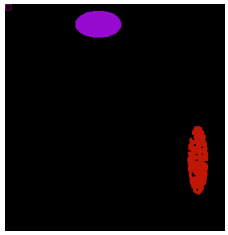
The integration of this plan by a novice has enabled to notice that system TMT was actually easy to take in hand: even if it is not always easy to give relevant names to high-level tasks, the modelling principles appeared clear and handy. This work has also raised new issues about the validation of the integrated knowledge (need to check keyboard errors as much as possible) and about man/machine communication (inadequate vocabulary or erroneous order of operations). This plan has also shown that the TMT system was not limited to operators of the library (although our library is rather exhaustive), as it includes a tool implemented as a C function and written on this occasion.

To become acquainted with our Pandore library of operators [Clouard 97], the novice first implemented its application as a Shell script. Then it was modelled as a TMT plan so as bring to the fore the underlying strategy, which demonstrated the educational aspect of our approach. As a matter of fact, if the first plan (C1) was built in a bottom-up manner, the two others (C2 et C3) were developed in a top-down manner, by relying on the strategy discovered in the former.

5.3. Search for similarities between applications from different fields

The two last plans we are now going to describe are working on images from two different origins (synthesis image and industrial image). Their integration into the TMT system have enabled to determine descriptive criteria that should be common to applications from different fields, and also to test the CBR module extensively.

The fourth plan (plan D) has been developed for testing the TMT system and, more precisely, some operators for selecting objects along shape criteria. It works on synthesis images, with the objective to sort objects according to their geometrical shape. The plan results are three distinct images, containing respectively rectangles, squares and ellipses.

This plan has enabled to test the interweaving of tools of various types: tools calling to Pandore operators and tools calling to Lisp functions. It shows that programming at the knowledge level facilitates the reuse of programming blocks written in different languages. Besides, this plan performs a new type of processing (detection of a specific shape) and works

on a new image type. It has thus enabled to enrich the case base with cases related to pattern recognition tasks and to increase the set of indexing terms.

The fifth plan (plan E) was entirely built thanks to the CBR module. The goal is to isolate and separate the objects of an industrial image. The result is a region map, where each object is labelled with a different colour.

The first case selected by the CBR module was the case associated to plan A; it was then adapted by using parts of plan B. We could thus demonstrate the relevance of selection criteria and the efficiency of the selection/adaptation algorithm: thanks to its interactive and recursive nature, this algorithm enables to rapidly get a first solution. However, the number of further local adaptations that we had to do, revealed the scarcity of our present case base, which must be enriched with plans performing more varied treatments.

5.4. Experimentation assessment

The experiments we have just briefly described have been conducted in the very course of the system's design, in order to detect weaknesses as soon as possible, to determine their cause and correct them.

Future experiments must include the integration of a wide variety of applications in different domains, with a view to enrich the vocabulary used for case description and increase the CBR module role. In particular, we are presently complementing the Pandore library with image interpretation operators, which should enlarge our field of investigation. We must also consider testing the system in "real conditions", i.e. have it validate by actual IP experts and not only by our team mates.

6. Conclusions

In this paper, a computational system for knowledge capitalisation and reuse in IP has been described. This system is used by the various IP programmers of our research team for:

- representing their know-how thanks to the module for the creation of applications,
- testing their applications through the interactive execution module,
- reusing knowledge previously modelled by different users, thanks to the CBR module.

The various experiments that have been conducted have demonstrated that our approach is quite promising. Criteria for indexing cases turned to be quite relevant and the recursive nature of the selection/adaptation algorithm enables to design a new solution by assembling parts of others.

To restrict the problem's scope, our experiments have been segmentation applications only. Now, it is time to diversify the content of our data bases (plans and cases), by

integrating applications performing more varied treatments (from image restoration to image interpretation) and working on images from more varied fields. This will be the occasion to enrich the indexing vocabulary and thus complete our set of criteria in order to present more exhaustive lists of terms to users. The major difficulty is to choose a common vocabulary for all IP programmers, because except for low-level actions (corresponding to operators from an IP library), there really exists no consensus on IP terms. In particular, it turns to be quite difficult to cut oneself off from the domain of application (most IP programmers work on one type of application at a time and thus only use terms from their current domain of application).

The issue of failure handling, regarding the retrieval algorithm must also be addressed. If no suitable task can be found, or none seems convenient to users, the system should propose to adopt a working process. For instance, one way to tackle a segmentation problem is to chain the following steps “pre-processing”, “detection”, “localisation” et “grouping together”, each task being an “abstract IP problem” (abstract in the sense that it must be more precisely specified in order to be solved). One could help users by proposing them to select a working process defined as a “suite of abstract problems”. The notion of “suite of problems” is inspired by CommonKADS approach [Breuker 94], in which reuse is based on a typology of problems and on the fact that there exist dependencies between the various types of problems. Once a “suite of abstract problems” has been selected by the user, the system could then re-run the retrieval/adaptation algorithm on each problem of the series, by asking user to define the corresponding target cases more precisely. We are convinced that “suites of abstract problems” well mirror the usual working processes of users. Such an abstract representation of working habits of different experts would thus be an additional asset for expertise share.

Acknowledgement

Our research on IP is carried out within the frame of the “Pôle Traitement et Analyse d’Image, TAI, de Basse-Normandie”. Special thanks to our IP experts F. Angot, R. Demoment, A. Elmoataz and N. Royackkers for their co-operation.

References

[Angot 99] F. Angot, Segmentation d’images 3D; application à la quantification d’images de tissus biologiques obtenues par microscopie confocale, PhD Thesis, Caen, 1999.

[Bonzano 97] A. Bonzano, P. Cunningham & B. Smyth, Using introspective learning to improve retrieval in CBR: A case study in air traffic control, *ICCBR’97*, Rhode Island, USA, pp. 291-302, July 97.

[Breuker 94] J.A. Breuker & W. Van de Velde, *The CommonKADS Library for Expertise Modelling*, IOS Press, Amsterdam, The Netherlands, 1994.

[Chandrasekaran 87] B. Chandrasekaran, Towards a functional Architecture for Intelligence Based on Generic Information Processing Tasks, *IJCAI’87*, Milan, Italy, pages 1183-1192, 1987.

[Clouard 97] Clouard R., Elmoataz A. & Angot F., PANDORE : une bibliothèque et un environnement de programmation d’opérateurs de traitement d’images, Rapport interne du GREYC, Caen, France, Mars 1997. http://www.greyc.ismra.fr/~regis/Pandora/

[Clouard 99] Clouard R., Elmoataz A., Porquet C. & M. Revenu, Borg : A knowledge-based system for automatic generation of image processing programs, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, n° 2, pp. 128-144.

[Elmoataz 96] A. Elmoataz, P. Belhomme, P. Herlin, S. Schupp, M. Revenu & D. Bloyet, Automated segmentation of cytological an histological images for the nuclear quantification: an adaptive approach based on mathematical morphology, Microscopy, Microanalysis, Microstructure, Vol. 7, pp. 331-337, October - December 1996.

[Ficet-Cauchard 98] V. Ficet-Cauchard, C. Porquet & M. Revenu, An Interactive Case-Based Reasoning System for the Development of Image Processing Applications, *EWCBR'98*, Dublin, Ireland, September 1998, pp. 437-447.

[Iris 93] *IRIS Explorer User's Guide*, Silicon Graphics, Inc., Mountain View, California, n°007-1371-020, 1993.

[Jacob-Delouis 96] I. Jacob-Delouis & O. Jehl, LISA-runtime (LIS@RT) : vers une utilisation industrielle du langage LISA, *JAVA'96*, Sète, France, May 1996.

[Kolodner 91] J.L. Kolodner, Improving Human Decision Making through Cased-Based Decision Aiding, *AI Magazine*, vol. 12, pages 52-68, 1991.

[Netten 97] B.D. Netten & R.A. Vingerhoeds, Structural Adaptation by Case Combination in EADOCS, *5th German Workshop on Case-Based Reasoning*, G*WCBR'96*, Bad Honnef, Germany, March 1997.

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

[Rasure 94] J. Rasure & S. Kubica, The Khoros Application Development Environment, Experimental Environments for Computer Vision and Image Processing, editor H.I. Christensen and J.L. Crowley, *Word Scientific*, Singapore, pages 1-32, 1994.

[Prasad 95] Prasad, Planning With Case-Based Structures, *AAAI Fall Symposium*, MIT Campus, Cambridge, Massachusetts, November 95.

[Smyth 96] B. Smyth, *Case-Based Design*, Doctoral Thesis of the Trinity College, Dublin, Ireland, April 1996.

[Steels 91] L. Steels, COMMET: a computational methodology for knowledge engineering , *ESPRIT project CONSTRUCT*, Deliverable WP/2/3/4, 1991.

[Talon 96] X. Talon & C. Pierret-Golbreich, TASK: from the specification to the implementation, *ICTAI'96*, Toulouse, France, November 1996.

[Van de Velde 93] W. Van de Velde, Issues in knowledge level modelling, *Second generation Expert Systems*, Editors: J.M. David, J.P. Krivine & R. Simons, Spring Verlag, 1993.

[Veloso 96] M. Veloso, H. Munoz-Avila & R. Bergmann, cased-based planning: selected methods and systems, *AI Communications*, vol. 9, n. 3, September 1996.