An interactive Case-Based Reasoning System for the development of Image Processing Applications

Valérie Ficet-Cauchard, Christine Porquet, Marinette Revenu

An Interactive Case-Based Reasoning System for the Development of Image Processing Applications

Valérie Ficet-Cauchard, Christine Porquet, Marinette Revenu

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

Valerie.Ficet@greyc.ismra.fr
Christine.Porquet@greyc.ismra.fr
Marinette.Revenu@greyc.ismra.fr

Abstract. In this paper, an interactive system for the development of Image Processing applications is described. This system is intended to provide some assistance to Image Processing experts during the construction as well as the execution of their applications. Through the system's graphical interface, an application can gradually be built and represented as a tree of tasks, and strategies can be made explicit so as to become reusable in future applications. In order to increase the assistance provided by the system, developers must be in a position to reuse parts of pre-existing treatments. To implement reusability, a Case-Based Reasoning approach is used. After defining two sets of criteria related to the task's description and the context of images, that enable us to classify a task in order to know when and how to reuse it, we describe our similarity functions and propose a search / adaptation algorithm for the retrieval of appropriate cases.

1. Introduction

In Image Processing (IP), the tuning of an application is a complex activity : difficulties arise when it comes to precisely defining a request, working out a solution and evaluating results. To overcome such difficulties, the IP expert has to make the most of the know-how acquired when developing previous applications. But the multiplicity of IP problems and domains of application, as well as the selectivity of human memory do not always allow the IP expert to reuse prior knowledge. Case-Based Reasoning (CBR) provides some kind of answer to the selective memory issue by increasing the IP expert's memory and by helping him to select previously solved cases that are close to his current problem.

In section 2, the distinctive features of IP and their implications on our problem-solving model are presented. Then, our system which enables, on the one hand, knowledge acquisition in Image Processing (IP) and on the other hand, the reuse of this knowledge is described. The acquisition step is done thanks to an interactive system (TMT) (section 3) that allows users to create and execute IP applications,

while representing the reasoning brought into play in these applications. The reuse step is managed by a Case-Based Reasoning module (section 4) that provides assistance to users so as to let them build new applications by reusing, as much as possible, previously developed ones.

2. Problem solving in Image Processing

2.1. Distinctive features of IP

In IP, the domain expert plays an important part during the development of applications. First, he / she has to define the problem (by providing one or several images associated with a request) that the IP expert will try to solve. Then, once a solution is proposed, it must be assessed by the domain expert, by viewing the resultant images. In fact, IP is a domain where no ideal evaluation function exists and no one but the domain expert can perform the final validation of an application. This validation can either be done visually, or be based on a more global testing protocol (e.g. statistics on object features extracted from the image). The roles of the domain expert and the IP expert are inter-dependent, thus requiring strong co-operation and communication between experts.

Moreover, the multiple and shaky nature of solutions makes the tuning of applications harder. Actually, any given problem has several solutions that can be more or less adapted to the current image and are more or less sensitive to the noise type and noise level of this image. To tackle these issues, a priori knowledge about expected results must be introduced and various strategies must be tried and tested. A priori knowledge consists of data about the image domain (e.g. in cytology, « cell nuclei are convex ») or data that comes from a thorough scene examination (e.g. « cell nuclei are dark objects laying on a light background »).

2.2. The use of operator libraries

The kind of approach we are advocating for the development of IP applications is based on the « intelligent » management of elementary programs stored in libraries, that are called IP operators in our context. Following this approach, building an IP application consists in chaining and tuning a set of operators selected from a given library.

Program libraries are not only used in IP. For instance, several systems developed thanks to the SCARP environment [18] solve problems by creating sequences of elementary programs stored into libraries (e.g. the Myosis system for electromyography diagnosis, or the Said system for diagnosis of vibrations in off-shore oil drill platforms).

Among the IP systems based on such libraries, a distinction should be made between graphical programming environments [9] [10], which enable users to select and organise operators into sequences (but not to justify or represent their reasoning) and automatic planning modules [6] [5] that are based on an explicit knowledge representation enabling reuse (but hardly - or not at all - allow users to intervene during the search for a solution). In order to take advantage of the interactive nature of graphical programming environments, our system should enable users to graphically select and link IP operators. But it should also give them a means to represent and make explicit the reasoning that has led them to this series of operators, with a view towards reusing the implemented strategy, in the same way as automatic systems do.

Our system uses IP operators from the PANDORE library [7] developed at the GREYC. Any operator from this library is a program performing an operation that cannot be further decomposed. Thus, any complex operation has to be decomposed into a sequence of several operators. Each operator takes as inputs images and numerical parameters and produces as outputs images and other (non-image) results. It can perform a specific operation on various image formats (pixel image, label image, region map, ...) and inputs / outputs are normalised.

2.3. Plan-based representation

An actual application can lead to sequences of up to tens of operators. In order to make explicit the reasoning brought to light during this process, our approach is to represent applications by means of IP plans. An IP plan is built by hierarchical decomposition of an original problem into simpler sub-problems. Each problem or sub-problem is associated with an IP task which, according to its level within the plan, will either describe the goal to be reached, the technique to be used or the algorithm to be applied. Such an approach based on the use of generic and decomposable tasks is widespread in problem solving [1] [4].

A plan can be schematised as a tree (fig. 1) which not only represents the linking and execution of IP operators corresponding to the leaves of the tree, but also the reasoning that lead to the building of such sequences of operators; this reasoning is represented by means of IP tasks schematised as grey rectangles.

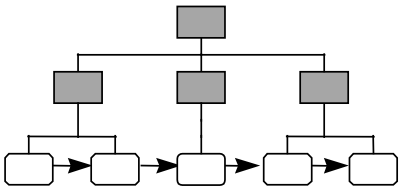
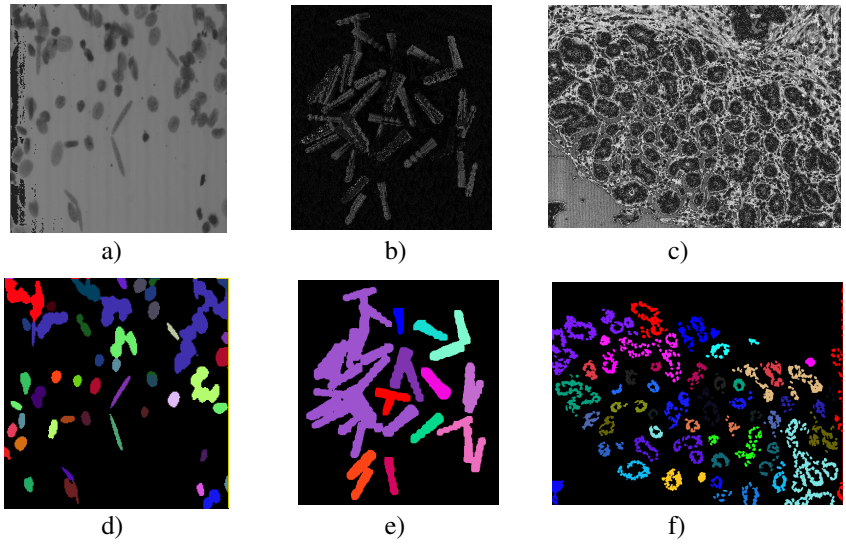


Fig. 1. plan representation

2.4. Reusability of applications

When IP experts are building applications in various domains, and representing them as IP plans, they should be able to reuse previously built parts of plans. They can actually be faced with a problem that was previously solved in other applications. For instance, the search for a particular type of cell nucleus in a cytological image (fig. 2-a and 2-d) corresponds to the following problem : « extraction of rather large and convex objects showing a grey level quite different from the grey level of the background ». Such a problem may have already been solved in the case of sorting industrial components lying in bulk on a conveyor belt (fig. 2-b and 2-e). In such a case, one should want to reuse the same strategy.



a) b) c)

d) e) f)

Fig. 2. input image and output image (or intermediate result) for the following applications :

a, d : cell classification in cytological image

b, e : sorting industrial components

c, f : extraction of clusters of cells in histological image

But one should also wish to solve a new problem dealing with a kind of image that has already been treated, or close to it. For instance, cytological and histological images of fig. 2, are somewhat analogous in the sense that they are both coming from the same biomedical material and the same acquisition device (microscope). The associated requests are however completely different : cell classification in the former versus extraction of clusters of cells in the latter (fig. 2-c and 2-f). But it turns out that the first two IP steps are identical (noise elimination and background elimination). When building the second plan, one should thus try to reuse these two steps of the first plan. To implement this kind of reuse, our approach is based on CBR techniques. CBR solves a new problem by retrieving and adapting solutions or parts of solutions of a

previously solved problem. It is based on a kind of reasoning that is close to human reasoning, and is thus well-suited for man/machine co-operation and enables us to focus on several categories of data about the plan, i.e. data about the request, the input images or the strategy to carry out.

3. Knowledge acquisition and modelling

3.1. The TMT 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, method selection, ...) and controlling the system's 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's behavioural rules with regards to users.

In our system, all three levels are uniformly represented thanks to the « TASK-METHOD-TOOL » (TMT) architecture [8].

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. 3-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 could be used. The method body can take two forms : either a decomposition into sub-tasks (fig. 3-c), described as a « THEN » tree - this is called a complex method, or the call to a run-time program through the medium of a tool (fig. 3-b) - this 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.

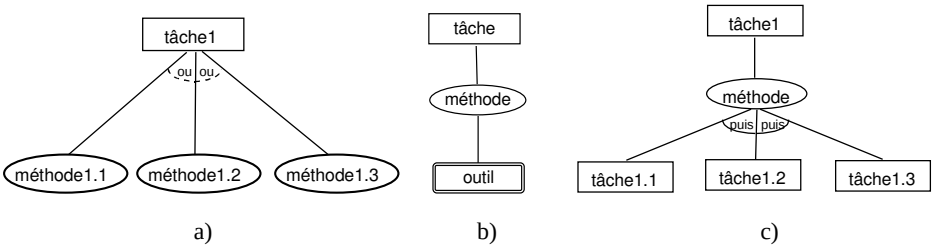


Fig. 3. Various possible links between TASKS, METHODS and TOOLS

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 (e.g. image thresholding, image smoothing, ...), but they can reify external Lisp or C functions likewise.

3.2. Interactive building of IP plans

Our knowledge-based system is provided with a graphical interface favouring man / machine co-operation for plan creation and execution.
When wishing to tune a new application, users can build their tasks and tools by filling in fields in adapted windows. Then, tasks and tools are linked by the 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. As it is possible to define all components in any order, users can adopt either a bottom-up building process (grouping sub-tasks together), or a top-down one (decomposing into sub-tasks), or even a mixed one.

To perform some processing on an image, users have to select a task in a menu; the corresponding plan is then instantiated when the image is given as input to the plan. The task is executed by proposing a choice of 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. Providing assistance to reuse

Owing to its closeness to human reasoning [12], Case-Based reasoning provides an attractive means to implement the reuse of IP plans, for it leads to a natural form of man / machine co-operation. When this kind of reasoning is used for planning purposes, it is called Case-Based planning, and finds applications in various domains such as building synthesis plans in organic chemistry with ReSYN/RàPC [11], providing assistance to automatic program synthesis with Déjà-Vu [16], or process planning with CAPLAN/CBC [17].

4.1 Case representation

In Case-Based planning, the representation of cases and the representation of plans must be closely related. In a system such as ReSYN/RàPC, where a plan corresponds to an ordered sequence of steps (a step being a state plus a transition), a case can be represented as a triplet < *set of steps , initial state , final state* >. This kind of representation is convenient when plans are not composed of too large a number of steps, or when the adaptation phase can be fully automated, which is not feasible in IP. So as to build a solution by combining plans, one has to resort, as in Déjà-Vu, to a representation enabling access to a plan at various abstraction levels. In our system, a case is defined as a pair < *IP problem, TMT plan solution to this problem* >. The solution plan can be accessed through its root task, with which are associated a set of criteria characterising this task and the problem it solves. These criteria are classified along two axes, whether they are dealing with the task's description or the application's context.

Criteria related to the task's description
These criteria are composed of the ***processing type*** or the ***processing step***, the actual ***definition*** of the problem and the associated task's ***abstraction level***.

According to the abstraction level of the current problem, either the ***processing type*** (image segmentation, image restoration, ...) or the ***processing step*** (e.g. in the case of segmentation : pre-processing, finding out of seeds, boundary localisation, ...) can be taken into account.

The ***problem definition*** is made up of a set of keywords that can be selected from keywords indexed in three lists : a list of verbs (the operations to perform, e.g. *detect, classify, binarise, ...*), a list of nouns (the objects on which the operation is performed, e.g. *boundaries, regions, labels, ...*) and a list of adjectives (to qualify the operation or the object, e.g. *local, large, partial, ...*).

Finally, ***abstraction levels***, which correspond to a horizontal division of the plan, are based on those defined in the automatic planner BORG [6] and are also closely related to the three abstraction levels mentioned by David Marr [13] in his study about visual perception. The first abstraction level is the *intentional* level. Tasks at this level answer the « what to do ? » question and deal with IP objectives. The second level is called the *functional* level. Tasks at this level answer the « how to do ? » question and

represent IP techniques, leaving aside technical constraints related to implementation. The third and last level is the *operational* level. Tasks at this level answer the « by means of what ? » question and represent a technical know-how in IP that can be implemented as algorithms. This level is close to the notion of IP operator, but without being linked to any specific library of operators.

Criteria related to the context of images

Among the criteria related to images, some of them describe image quality. In particular, they are dealing with ***noise type, noise amount***, ***contrast quality***. Such criteria are particularly important when it comes to choosing preliminary processing steps (noise reduction, contrast enhancement, …).

Other criteria are somewhat related to semantic image description and the kind of objects that are to be extracted. One can mention the ***presence or absence of image background***, the ***aspect*** of the objects to be detected (homogeneous grey level, light colour, textured region, thick boundaries, ...), their ***form*** (convex, concave, elongated, compact, ...), their ***relative size*** and ***relative position*** (right, left, under, above, at the centre of, ...) compared to the other objects of the scene and finally, ***inter-object relations*** (near, connected, included, ...).

Similarity calculation

Two kinds of approaches can be considered for the retrieval of similar cases, whether they are based on maximising similarity [3], or minimising adaptation effort [16][14]. In IP, as there is no suitable automatic evaluation method, only the second approach is conceivable. Moreover, all the previously mentioned criteria are not taken into account in every application. So, like ISAC [2], we need an iterative process in order to determine which are the important criteria for the current case. This process takes place in co-operation with the user that has to provide data to fill in criteria or to order them according to their importance with regards to the current case.

The order on criteria enables us to assign them an importance coefficient α_{Cr}, non-filled criteria being affected with a null value. Comparison functions $\varphi_{Cr}(S,T)$ are defined to compute similarity between a source case *S* and a target case *T* on each criterion *Cr*. These functions return a value between 0 and 1 and are defined according to the criterion type : proportion of common items for criteria defined as a list of terms, such as the problem definition ; distance calculation for numerical criteria, such as noise level, contrast quality, and abstraction levels. Comparison functions and importance coefficients are used in two similarity functions, $\Phi_t(S,T)$ and $\Phi_i(S,T)$, that are normalised by the sum of coefficients :

$$\Phi_t(S,T) = \frac{\left(\sum \alpha_{Cr} \times \varphi_{Cr}(S,T)\right)}{\sum \alpha_{Cr}} \forall Cr \in \{\text{criteria related to the task's description}\}$$

$$\Phi_t(S,T)=\frac{\left(\sum \alpha_{Cr}\times\varphi_{Cr}(S,T)\right)}{\sum \alpha_{Cr}}\forall Cr\in\{\text{criteria related to the task's description}\}$$

The use of these functions is described in the next paragraph. Take note that some criteria are directly filled in by the user, whereas others are either computed (e.g. contrast) or deduced from the image domain (e.g. noise).

4.2. Case retrieval of previous solutions

Some CBP systems base their search for a solution on the selection of strategies for the decomposition of a problem into sub-problems. Déjà-Vu, for instance, operates in several cycles : at each cycle, a search for a sub-solution to one of the sub-problems is done, this sub-solution being either a concrete solution or, in turn, a decomposition into sub-problems. We, for our part, like CAPLAN/CBC, are rather aiming at first selecting a whole plan, the description of which is close to the current problem, then adapting it by re-using parts of other plans.

In most systems, the selection of the nearest case from the current case is done in several steps : search for compatible cases, search for the more adaptable ones, then user consultation to validate the final choice in ISAC; prototype selection to reduce the search space, then selection of the case that best satisfies the problem in EADOCS [15]. As for us, we also operate in several steps, aiming first at reducing the search space by means of function Φ_t, then at selecting the nearest cases to the current case thanks to function Φ_i, and at last consulting the user for the final decision.

As was explained in the previous paragraph, the retrieval of a suitable case consists in finding out the root task of the associated plan. This search is done according to the following search/adaptation algorithm :

```
1- ask user for values of criteria related to the problem's description
2- determine the set T of the tasks matching the desired criteria by means of Φt
3- while cardinal of T is not small enough do
        - ask user for values of criteria related to the context of images
        - reduce set T by means of Φi
4- ask user to select a task among the set of remaining tasks
5- once a task is selected, the corresponding plan can be adapted by re-starting the search/adaptation algorithm on unsuitable sub-tasks, or user can manually build new parts of the plan.
```

Three thresholds are used in this algorithm, that must be user-defined : minimum value of Φ_t, maximum value of Φ_i, maximum number of tasks in set ***T***. The

iterative nature of step 3 enables us to ask for criteria values until a small enough list of cases is reached, in which the user can make his/her choice, while taking into account the importance order of criteria (the earlier a criterion is filled in, the higher its importance coefficient). This also allows us to only fill in the indispensable criteria, so as to leave enough weight to the intuitive aspect that characterises the user's final choice.

4.3. Adapting the case to the current problem

In CBP, plan adaptation can either be done through generation or transformation. Adaptation by generation is based on a planner-generator in order to solve remaining goals, as in CAPLAN/CBC, or to refine an abstract solution, as in PARIS [17]. Adaptation by transformation proceeds by substitution or instantiation of sub-solutions taken from other cases, as in EADOCS.

In our system, plan adaptation consists of modifying some sub-tasks or some tools. These modifications can be local (replacement of an operator by another one or modification of parameter values) or global (replacement of a sub-plan by another one). To achieve such modifications, the search/adaptation algorithm is applied locally and recursively. More precisely, once a task is selected at step 4, the user can modify the associated plan by re-starting the algorithm on unsuitable sub-tasks, or by creating new parts of plans with the help of TMT. This recursive aspect of the algorithm ensures plan adaptation at any level in the tree of tasks. Plan evaluation, as well as final validation allowing integration into the base of cases are done by the user after plan execution and result assessment.

5. Conclusions

In this paper, a computational system based on the TMT architecture and equipped with a graphical interface for user-friendly man / machine interaction has been described. We have also tried to show how Case-Based Reasoning could increase the assistance provided by our system. For this purpose, a search / adaptation algorithm and selection criteria have been proposed, in order to find out when an IP task is reusable. As it is possible to re-apply this algorithm locally and recursively, users can adapt their plan as accurately as desired. Several plans dedicated to various IP problems and dealing with images from different origins have been labelled by means of the selection criteria described in this paper. The good detection of similarities between these plans thanks to this labelling reinforces our confidence in the relevance of our criteria. The validation of the CBR module is presently being completed by effective implementation and testing of the search / adaptation algorithm.

References

1 Benjamins R. & Pierret-Golbreich C., Assumption of Problem-Solving Methods, Proceedings of the 6th KEML Workshop, Paris, 1996.

2 Bonzano A., Cunningham P. & Meckiff C., ISAC : A CBR System for Decision Support in Air Traffic Control, EWCBR'96, Lausanne, Switzerland, November1996.

3 Caulier P. & Houriez B., Apport de la modélisation des connaissances à partir de cas pour la capitalisation et la réutilisation de connaissances, JAVA'95, Grenoble, April 1995.

4 Chandrasekaran B. & Josephson J.R., The Ontology of Tasks and Methods, AAAI Fall Symposium, Stanford University, March 1997.

5 Clément V. & Thonnat M., A Knowledge-Based Approach to Integration of Image, CVGIP: Image Understanding, Vol. 57 (2), Academic Press, p164-184, 1993.

6 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 Co-operative Systems, Juan-les-Pins, France, p298-313, January 1995.

7 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, March 1997.

8 Ficet V., Construction interactive d'un modèle conceptuel d'applications de traitement d'images, RJC-IA, Nantes, France, p95-102, August 1996.

9 IRIS Explorer User's Guide, Silicon Graphics, Inc., Mountain View, California, n°007-1371-020, 1993.

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

11 Lieber J. & Napoli A., Planification à partir de cas et classification, JICAA'97, Roscoff, France, 1997.

12 Kolodner J.L., Improving Human Decision Making through Case-Based Decision Aiding, AI Magazine, vol. 12, pp52-68, 1991.

13 Marr D., *Vision : A computational investigation into the human representation and processing of visual information*, W.H. Freeman and Co, San Francisco, 1982.

14 Munoz-Avila H. & Hüllen J., Feature Weighting by Explaining Case-Based Problem Solving Episodes, Processing, Smith I. & Falling B. eds., EWCBR'96, Lausanne, Switzerland, November 1996.

15 Netten B. D. & Vingerhoeds R. A., Case Combination for Conceptual Design, EWCBR'96, Lausanne, Switzerland, November 1996.

16 Smyth B. & Keane M.T., Retrieval & Adaptation in Déjà-Vu, a Case-Based Reasoning System for Software Design, AAAI Fall Symposium'95, MIT Campus, Cambridge, Ma., November 1995.

17 Veloso M., Munoz-Avila H. & Bergmann R., Case-based planning : selected methods and system, AI Communication, vol.9, n.3, September 1996.

18 Willamowski J., Modélisation de tâches pour la résolution de problèmes en coopération système /utilisateur, Thèse de doctorat, Université Joseph Fourier, Grenoble, France, April 1994.