Knowledge-based Techniques for Image Processing and for Image Understanding

Monique Thonnat

1 Introduction

The objective of this chapter is to show how explicit expertise can help solving complex image processing problems. Two different kinds of problems are addressed : the use of an image processing library and the automation of image understanding.

A major problem image processing library users are faced with is how to build an application for a specific problem. For instance how to select the programs and how to set the tunable parameters to get pertinent information from the images at hand. This problem has been addressed by researchers in the field of artificial intelligence and image processing and is often called intelligent image processing, program supervision, software re-use or even software configuration.

A second difficult problem which has been addressed for years by researchers in the field of artificial intelligence and image processing is how to extract the semantics contained in an image or a set of images. This problem of image understanding can range from object recognition in the case of a static image to event and scenario recognition in the case of temporal sequence of images.

First in section 2 we briefly introduce artificial intelligence concepts and more precisely knowledge-based techniques. These techniques are then used in section 3 to build intelligent image processing systems and in section 4 to build intelligent image understanding systems.

2 Artificial Intelligence

It is very difficult to find a unique definition of Artificial Intelligence (A.I. in short). Among the various definitions available in the literature ([5],[6],[19], [4], [46]) we have selected the two following ones: "A.I. is an attempt to build computable models of cognitive process", or "in A.I. we affect to computers tasks which would be considered as intelligent if they were performed by human beings". In fact these definitions illustrate two main reasons to study A. I.. Researchers in cognitive psychology and cognitive science want to study the human brain; their interest is finding programs simulating human performances. So they are close to the first definition. Computers scientists and

engineers want to extend the capabilities of computers; their interest is in technological applications of Artificial Intelligence. Their motivation is closer to the second definition of A. I.. Let us briefly study the application domains of A.I. with respect to the results obtained so far. Games have been among the first application domains. Very good results have been obtained for difficult tasks as playing chess as the best systems are able to reach the level of experts; on the same way automatic theorem prover are able to solve scholar problems in group theory, topology, geometry or mechanics. For years the domain of medical diagnosis has been studied by A.I. scientists and experts systems were able to produce diagnosis and to propose treatments for particular diseases. On the contrary for domains very close to daily life, like natural language understanding, speech understanding and visual perception the results are very far from what a children can do easily (i.e. to speak and to see).

2.1 Knowledge-based systems

In this section we introduce the notions of knowledge-based systems and knowledge representation which will be useful in the following. Knowledge-based systems are the successors of the old expert systems based on production rules and inference engines. Dendral which has been developed by a team in Stanford in 1970 for mass spectrography in chemistry is considered as the ancestor of the expert systems. The first expert system, Mycin, has been created in 1974 for diagnosis of infectious blood diseases (for more detailed historical information see [5] and [46]). The main idea of expert systems is to simulate the behaviour of a specialist in a very precise domain. The objective is to cope the absence of a specialist either for problem solving or for teaching. Expert systems are software tools which use the knowledge provided by a specialist, which are able to handle symbolic and uncertain data and which use a reasoning strategy. These systems separate knowledge contents and deductive tools using this knowledge. They are not classical softwares as there is a clear distinction among the different roles in their building and their use. Three roles are distinguished: the end-user, the expert and the software engineer. More precisely the end-user fills the base of facts with data to be processed; the expert builds the knowledge base with the expertise of the domain; the software engineer builds a reasoning engine which is independent of the expert's domain but which implements a reasoning strategy. Knowledge-based systems on the contrary to expert systems are not restricted to production rule as knowledge representation scheme and the reasoning engine is not limited to an inference engine. Different types of reasoning have been experimented in the literature to solve various problems. Among them the main ones are classification, diagnosis, design, simulation and planning.

2.2 Knowledge representation

A knowledge representation is a set of syntactic and semantic conventions to describe a piece of knowledge. In order to build knowledge bases the experts describe the contents of their knowledge using knowledge representation schemes. A knowledge representation scheme must allow to explicit what is important and must be easy to handle. Two main knowledge representation schemes are presented in the following, production rules and frames. These schemes will be used in section 3 for representing the knowledge on the use of image processing library and in section 4 for representing the knowledge on image understanding.

Production rules Production rules are directly inspired from predicate logic and the inference capabilities of modus ponens. A production rule very generally expresses an "if-then" relashionship.

General syntax for production rules
Rule ruleX
comment : "a comment explaining the rule"
if
condition
and/or
condition
then
conclusion

which means that if a set of conditions are true then the conclusion holds. For instance in the domain of the bridge game, if we want to express that during bidding one can open with a specific distribution, we can use the production rule R1bridge:

Example of production rule
Rule R1bridge
comment : "bidding rule for opening"
if
balanced hand
and
16 to 18 head points
then
opening 1 no trump.

This rule means that if we have a balanced hand and between 16 and 18 head points then we can open at the level of 1 no trump.

Each piece of knowledge can be expressed with a different rule. A knowledge base can contain hundreds of production rules. It is very easy to add,

modify or remove a production rule in a knowledge base. This allows to extend or update a knowledge base. Production rules can be grouped in different sets to structure the knowledge and help the reasoning.

The power of expression of production rules relies on the kind of logic they are based. For rules based on propositional logic symbols represent all propositions or facts, everything is a constant. It was the case in the previous example for bidding rule R1bridge. A usual kind of production rules are rules based on 0+ order logic. The formalisms $(object, attribute, value)$ or $(attribute, value)$ are used; the values can be referenced. The rule R2 shows an example of 0+ order production rule, where the value of the temperature in the room is not directly given in the rule but referenced through the attribute room-temperature.

Example of 0+ order production rules
Rule R2
comment : "use the heater if the temperature is too low"
if
room-temperature $<$ 19
then
change heating-status to on.

For rules based on first-order logic or predicate logic quantifiers and variables can be used. The rule R3 shows an example of first-order production rule, where two variables X and Y and one quantifier, $\exists$, are used.

Example of first-order production rules
Rule R3
comment : " track mobile regions which are possible human beings"
if
$\exists$ mobile object X
and
shape of X = human
and
1.4 meters $<$ size of X $<$ 2 meters
then
track X.

In conclusion we can say that production rules are simple, in particular in the case of propositional logic; they are very modular and readable. They allow fast modifications of a knowledge base and explanations are easy to provide to a user. On the contrary they have different drawbacks: the knowledge is fragmented, the uniform formalism for expressing the knowledge leads to a lack of efficiency for problem solving.

Frames A Frame is a knowledge representation scheme which comes from cognitive research on human reasoning. The hypothesis is that human beings

refer when reasoning to prototypes already stored in memory and compare them to objects or events corresponding to new situations.

Minksy in 1975 has proposed Frames for previsions concerning structured objects. Schank in 1977 has proposed Scripts for previsions concerning sequences of events. The concept of Frames is a declarative knowledge representation scheme well adapted for structured object descriptions.

A Frame is a set of attributes (or properties). Each attribute has several slots which describe the characteristics of the attribute. The attributes are very dependent of the nature of object it represents. The slots are predefined general characteristics useful for any kind of attribute; classical slots are the type, the current value, a default value, a possible range of values. Frame1 is an example of frame with n attributes and 4 kind of slots.

	Frame1	
Attributes	Slots	*Slot values*
attribute 1	type	*type-value*
	value	*attribute-value*
	default	*default-value*
attribute 2	type	*type-value*
	value	*value*
	range-of-values	*range-value*
attribute 3	type	*type-value*
	value	*attribute-value*
	range-of-values	*range-value*
	default	*default-value*
....	...	
attribute n	type	*type-value*
	value	*attribute-value*
	range-of-value	*default-value*
	default	*default-value*

The general frame Frame1

A frame is useful to describe a general concept, as the concept of a chair. The frame Chair here-after describes a chair in terms of a number of legs, a back and a number of arms.

Frame:Chair		
number-of-legs	type	*integer*
	value	*unknown*
	default	*4*
back	type	*symbolic*
	value	*unknown*
	range-of-values	*(straight, curved)*
number-of-arms	type	*integer*
	value	*unknown*
	range-of-values	*(0, 2)*
	default	*0*

The frame Chair describing a general chair

Frames are also useful to describe particular objects or instances of a class of objects. The following example of frame, John's chair, describes a specific chair (i.e. John's chair); the attributes are the same as for the general frame Chair but attributes values are set to John's chair description. For instance the back is curved.

Frame:John's chair		
number-of-legs	type	*integer*
	value	*4*
	default	*4*
back	type	*symbolic*
	value	*curved*
	range-of-values	*(straight, curved)*
number-of-arms	type	*integer*
	value	*0*
	range-of-values	*(0, 2)*
	default	*0*

A frame describing a particular instance of chair

The properties which can be described with frames are: internal properties (size, age, color,etc...), structural properties (subparts), relations between objects (close-to, above, etc..), roles (father, server, etc...).

Frames can be organized in hierarchies, with inheritance of structure, values or daemons. Typical links between the frames are specialisation and instantiation links. The specialisation link **kind-of** enables to define subclasses. The instantiation link **is-a** enables to attach to a general class several instances.

Frames are mainly a declarative static representation. However a certain level of dynamicity exists because procedures can be attached to the attributes to guide the reasoning. In particular specific daemons can be defined

in slots. For instance very often a slot *if-needed* allows to attach a function to compute the value of an attribute if the value is unknown. Another possibilty is to define a slot *if-added* to use a function to check some coherency if the value of the attribute has been filled.

Frames have got a very large success due to appropriated languages as frames language (i.e. KRL). Their impact has been larger than knowledge representation domain. They have strongly influenced the design of he object oriented languages in computer science (i.e. smalltalk, C++).

Hybrid representation There is no unique universal representation scheme. Rules and Frames have their specificity and can be used in hybrid systems in different ways: cooperation where rules describe heuristics and frames represent the objects and their relations, frames can be referenced in rules, frames can activate rule bases (using daemons), rules can be frames which are grouped into classes.

2.3 Conclusion

The initial goal of Artificial Intelligence was very ambitious. After years of research in this domain, it appears that building a general problem solver (see GPS in the seventies) and aiming at representing the general knowledge we have on the world is an utopy. On the contrary Artificial Intelligence has provided efficient tools for modelling reasoning and for knowledge representation. We will describe in the following how the techniques can help solving specific problems when the knowledge is well formalized.

3 Intelligent Image Processing Systems

The field of image processing has produced a large number of powerful programs and many different program libraries have been developed. In such libraries the individual programs are integrated from a low-level point of view. But no support is provided to users who need to solve practical image processing problems. Every end-user cannot have a deep understanding of program semantics and syntax. Inexperienced ones may only have a basic understanding of the field of image processing and its terminology. On the other hand, programs implement more and more complex functionalities and their use is equally difficult. If it is too demanding for an end-user to catch the complexity of new programs, these programs will never be widely applied. In this section our goal is to present program supervision techniques to help users manage image processing techniques that are needed for their applications. From the analysis of the task of an image processing expert who processes data using a set of programs, knowledge models related to this task have been derived. Based on these models, knowledge-based systems for program supervision take in charge the management of the library use and

free the users of doing it manually. This aid can range from an advisory guide up to fully automatic program monitoring systems. The basic idea is to automate the choice and execution of programs from a library to accomplish a user's processing objective. This is done by encapsulating the knowledge of program use in a knowledge base and by emulating the strategy of an expert in the use of the programs.

3.1 Program Supervision

Program supervision is a research domain with an increasing number of work coming from many applicative and technical domains [56]. These research activities are often motivated by a particular application domain (as image processing, signal processing or scientific computing). Contrary to knowledge-based systems for image interpretation which have been studied for about 20 years, knowledge-based systems for image processing program supervision are more recent. In Japan, numerous teams, belonging both to the research and to the industrial sector, have spent an important effort on this problem (see [58], [47] and [35]). In Europe work has been developed either as research on general tools (as OCAPI [18]) or as industrial tools for a particular application (as the VIDIMUS Esprit project [10]). In the United States early work has been done by Johnston [32] and by Bailey [3]. Recently [26] and [15] have used planning techniques for composing image analysis processes.

Program Supervision aims at facilitating the (re)configuration of data processing programs. Such a configuration may involve the chaining of many different programs. It is to note that the program codes usually pre-exist (in a library, for example) and the goal is not to optimize the programs themselves, but their *use*. A program supervision knowledge-based system helps a non-specialist user to apply programs in different situations as shown in figure 1.

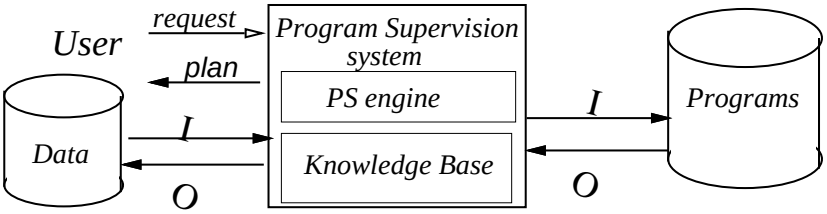


Fig. 1. A knowledge-based program supervision system helps a user (re)use a set of programs for solving a request on input data $\mathcal{I}$ to obtain output data $\mathcal{O}$, as the result of the execution of a plan of programs. It is composed of a program supervision engine and a knowledge base.

A knowledge-based system performing program supervision is composed of 1) a set of existing programs, 2)a knowledge base describing how to use

these programs and 3) a program supervision engine. The role of the program supervision *engine* is to exploit knowledge about programs in order to produce a plan of programs, that achieves the user's goal. It emulates the strategy of an expert in the use of programs. The final plan that produces satisfactory outputs is usually not straightforward, it often results from several trials and errors. The reasoning engine explores the different possibilities and computes the best one, with respect to expert criteria, available in the knowledge base. The reasoning of a program supervision system consists of different phases, that may be completely or only partly automated. The engine of a program supervision system can be decomposed into four phases: *planning* and *execution* of programs, *evaluation* of the results, and *repair*. Figure 2 shows the roles and interactions between the different phases.

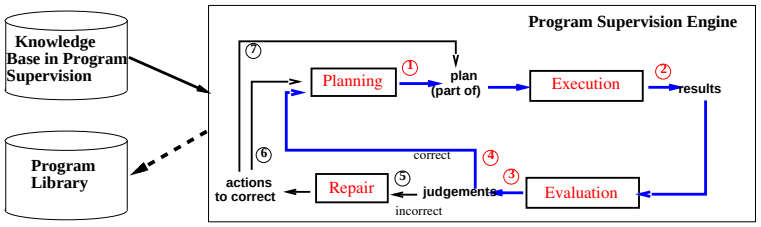


Fig. 2. The different phases of a program supervision system

The *knowledge base* contains *operators* which are representations of programs (with descriptions of their data and parameters) and of typical combinations of programs, as well as criteria to guide the reasoning process. The contents of the knowledge base should be sufficient for the engine to select the programs, to initialise their parameters, to manage non trivial data-flow and to combine the programs to produce a satisfactory plan of programs, depending on the input data and user's request. An exemple showing how an engine uses this kind of knowledge can be seen in figure 3; this figure presents a simplified version of the algorithm of the OCAPI program supervision engine ([18]).

In the following we describe the main concepts of a knowledge base in program supervision : Data, Goals, Requests, Operators and Criteria.

- **Data** contain all necessary information on the problem of the end-user;
- **Goals** express constraints on the expected final state;
- **Requests** are instantiations of goals on particular data, under particular constraints;
- **Operators** There are two types of operators: primitive and complex ones. A *primitive operator* represents a particular program and a *complex operator* represents a particular combination of programs. Program combinations correspond to decompositions into more concrete programs,

-1- global matching
while not all requests have been processed
-2- selection of the request to be processed
-3- classification of the operators (using **choice rules***)*
while the request is not satisfied
-4- selection of the best operator
-5- execution of the operator(using **initialisation or adjustement rules***)*
-6- assessment of the results (using **evaluation rules***)*

Fig. 3. The algorithm of a program supervision engine (OCAPI)

at different levels of abstraction, either by specialization (alternatives) or composition (sequences, parallel, loops, etc.).

1. **Primitive Operators** are the basic components that manipulate data. Their representation is composed of:
   - A functionality (information on "what is the operator for?");
   - A method or an action (e.g. calling syntax, simulation method);
   - Information on arguments (e.g. name, type of input and output arguments, signature) "what does the operator act on?";
   - Semantical information: characteristics (known by the expert), constraints, pre and post conditions (on the data);
   - Result evaluation criteria, argument initialisation and adjustment criteria.

   An example of a primitive operator *thres-hyst* is shown in figure 4. The functionality is thresholding, the corresponding program `hyster` works with a hysteresis method and two thresholds; it has an input image, two tunable parameters (two thresholds) and an output image; several preconditions on the input image coding format and noise are specified and the effect on the output is set.

Name	*thres-hyst*
Functionality	*thresholding*
Characteristics	*>-1-threshold, hysteresis-method*
Input arguments	*in : image*
Output arguments	*out : image*
Parameters	*upper-threshold : thres-hyst-threshold*
	lower-threshold : thres-hyst-threshold
Preconditions	*in.coding.format == inrimage*
	in.presentation.noise.kind == gaussian
Effects	*out.contents.segmented = value-based*
Calling syntax	`hyster` *in out* `-sh` *upper-threshold* `-sb` *lower-threshold*

Fig. 4. Example of a primitive operator

2. **Complex Operators** are skeletons of plans, provided by the expert. They describe the networks of known possible connections between operators (choice, sequence, entailment, repetition, etc), in order to achieve a given goal. Each complex operator may be decomposed into sub-elements. A complex operator may be represented as a graph, the leaves of which are associated with primitive operators. In addition to the operator information, complex operators contain information about data flow between their elements, together with tactical knowledge i.e. criteria to decide how to traverse the network.
   An example of a complex operator *pyr-level-match* is shown in figure 5. The functionality is pyramidal stereovision matching; it has a sequential decomposition into three substeps, first images-reduction, then primitives-extraction, finally stereo-matching; the input, output arguments are specified and the data flow is expressed.

Name	*pyr-level-match*
Functionality	*pyramidal-stereo-matching*
Input arguments	*right-image: promethee-image*
	left-image: promethee-image
Output arguments	*3D-image: promethee-image*
Subcomponents	*images-reduction, primitives-extraction,*
	stereo-matching
Control	`DO` *images-reduction*
	`THEN` *primitives-extraction*
	`THEN` *stereo-matching*
Data flow	*stereo-matching.out-image* $\rightarrow$ *3D-image*
	...

Fig. 5. Example of a complex operator

- **Criteria** Various criteria, implemented by rules, play an important role during the reasoning, e.g. to choose between different alternatives (*choice criteria*), to tune program execution (*initialisation criteria*), to diagnose the quality of the results (*evaluation criteria*) and to repair a bad execution (*adjustment* and *repair criteria*).
  1. **Choice rules** The role of a choice rule is to choose between different alternative operators having the same functionality. An example of two alternative operators which perform the same stereo-matching functionality but with different characteristics is shown in figure 6). An example of choice rule describing how to select between these operators based on their characteristics is shown in figure 7.
  2. **Evaluation rules** The role of an evaluation rule (also called assessment rule) is to diagnose the quality of the results. Two examples of these rules are shown in figure 8. The first rule computes an assessment concerning a too big filter size and decides that there is

Fonctionality :	stereo-matching
Operator-1:	o-ma-stereo-match
Characteristics:	high-quality
	low-adaptability
Operator-2:	o-meygret-stereo-match
Characteristics:	average-quality
	high-adaptability

Fig. 6. Example of alternative operators

If *context.user-constraints.quality-matching == high*
Then *use-operator-with-characteristic high-quality*

Fig. 7. Example of choice rule

a failure. The second rule displays results to an end-user in case of an interactive mode and proposes three possible qualifiers (correct, too-low, too-high).

If *xmax <= 255*
and *rayleighmax > 50*
Then *global-assess filter-size = too-big*
and *failure*

If *context.user-constraints.mode == interactive*
Then *assess-by-user thresholding (correct, too-low, too-high)*

Fig. 8. Example of two evaluation rules

3. **Initialisation rules** The role of an initialisation rule is to set the value of operator parameters before program execution. Two examples of these rules are shown in figure 9.

If *context.details == too-few*
Then *initialise thresh to 90*

If *context.noise == high*
Then *initialise window-size to 7*

Fig. 9. Example of initialisation rules (thresh and window-size are parameters)

4. **Repair rules**
The role of repair rules is to define a repair strategy after a failure decided by evaluation rules. For instance the repair rule shown in

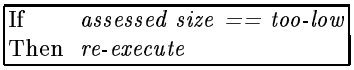


Fig. 10. Example of a repair rule

figure 10 implements the following strategy : if a bad assessment of the size has been decided by evaluation criteria, re-execute the same operator.

5. **Adjustment rules** The role of an adjustment rule is to propose a way to tune a parameter value to improve the quality of the results of an operator. Two adjustment rules are shown in figure 11. The first rule expresses that the parameter (window-size) needs to be tuned after a certain type of assessment (noise == too-high) and how this parameter must be must be changed (increase), the second rule expresses that after an ambiguous detection the parameter thresh must be decreased and that an option setting must be changed.

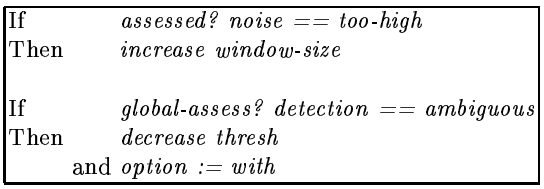


Fig. 11. Example of two adjustment rules (window-size and thresh are parameters)

3.2 Application of program supervision techniques

This section briefly shows examples of program supervision systems for image processing in very different application domains. Two examples are related to road obstacle detection and one example deals with medical imagery. In addition an example in astronomy for galaxy morphology description will be detailed in a separate section.

Stereovision-based Road Obstacle Detection The objective in this example is to use program supervision techniques for the detection of objects, such as cars, in urban scenes based on stereo data (see figures 12 and 13). The library of programmes is compounded of 24 modules (mainly a pyramidal stereovision algorithm based on contour chain points ([37]). The role of program supervision is to enable the re-use of these programmes for very different images. For that 54 numerical parameters must be initialised. In this example two kinds of problems are automatically detected and repaired: bad primitive extraction and bad matching. Two other kinds of thresholding problems are interactively detected and automatically repaired.

Fig. 12. A pair of stereo images in a urban scene

Fig. 13. Detected cars in a urban scene.

Real-time Road Obstacle Detection The second example takes place in a european Eureka project for driving assistance. The objective is to use program supervision techniques to perform real-time program configuration [38]. The programmes are organized into three perception modules:

- a **3-D obstacle detector using telemetry**
  (3 modes: *full-scanning, unique-measure , tracking*, and hardware constraints)
- an **obstacle detector based on mathematical morphology** (2 modes: *detection or focusing*),
- a **motion segmenter** (input data: switching between 2 cameras, and hardware constraints).

Each module has been developed by a different French vision research laboratory (LASMEA at Clermont, IRISA in Rennes and Mines Paris). The role of program supervision is to decide which data must be processed and how they must be processed (module launch or stop, tuning of different modes, hardware constaints, etc...).

Medical Imaging In this example the objective is to provide a clinician with better access to new medical image processing techniques [22]. The images are 3D MRI images of the brain (see figure 14). The programmes are 21 modules (built by Epidaure from INRIA Sophia). The role of program supervision is to work in interaction with the user leaving them the task of interpreting the results. The image processing decisions which have been automated are: *scheduling* 10 to 15 programmes w.r.t. image characteristics, *initialisation* of numerical values for 27 parameters and 7 *choices* between alternative methods.

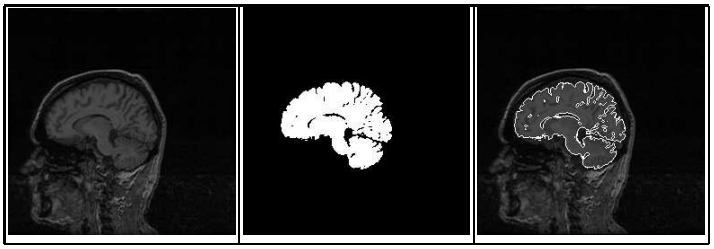


Fig. 14. Brain segmentation of 3D MRI images

3.3 Program supervision for galaxy image processing

We are interested in automating the data processing of images containing a galaxy. Our objective is to capitalize the knowledge in galaxy image processing. Another important motivation of this work is to provide a complete tool for automating the classification of galaxies. It is well known that image processing is a hard task, especially when the images to process are natural object images. This type of image has several characteristics : natural objects have very complex shapes and there are no simple geometric models to describe them. Thus, automating the treatment of such images requires a great amount of knowledge of image processing. Galaxies are typical natural objects. They present a great variability in their shapes as well as in their luminosity depending on many different factors (time of exposure, intrinsic luminosity of the galaxy, the distance between the galaxy and the telescope...) As our goal is to automatically classify the galaxies like experts ([60]), we

have to extract several numerical data describing, as precisely as possible, the galaxies. So, we need (but this is not sufficient) a system which is able to process images. This system has to be more intelligent than this ; it has to be able to adapt itself to different processing situations. One solution is to use a specialized program in order to process the images. But such a program is generally too rigid and is not able to adapt itself to different situations such as different qualities of the input images (e.g. presence of noise), different natures of images (intensity, density), different sizes, different acquisition ways (with CCD camera, with photography scans), and so on. Furthermore, a specialized program is hardly readable by anybody else than the author. In consequence, such a program becomes more and more difficult to extend or to update.

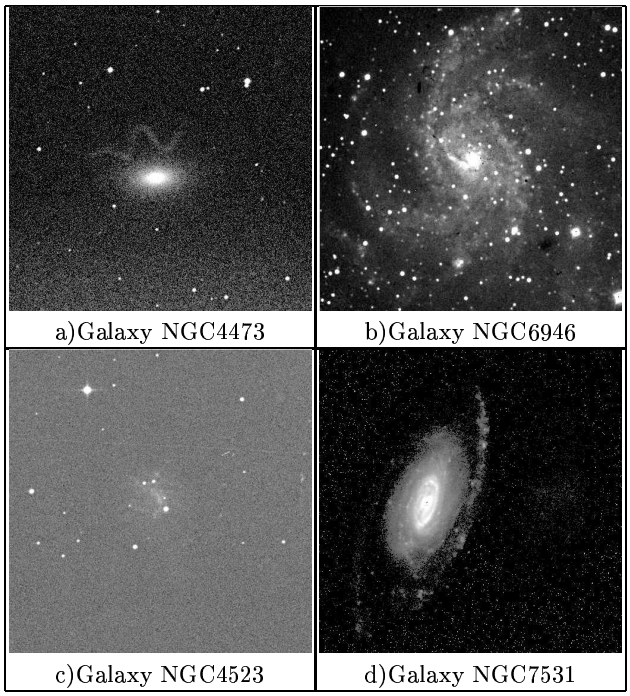


Fig. 15. Examples of possible variations in galaxy images

So we propose to use program supervision techniques dedicated to galaxy image processing ([51,50]) to automatically adapt the processing to variations in the images (see figure 15) and to provide optimal input to an automatic galaxy classification system.

We have developed PROGAL a knowledge-based system dedicated to supervision of a library of programs in astronomy. The library is a set of 37

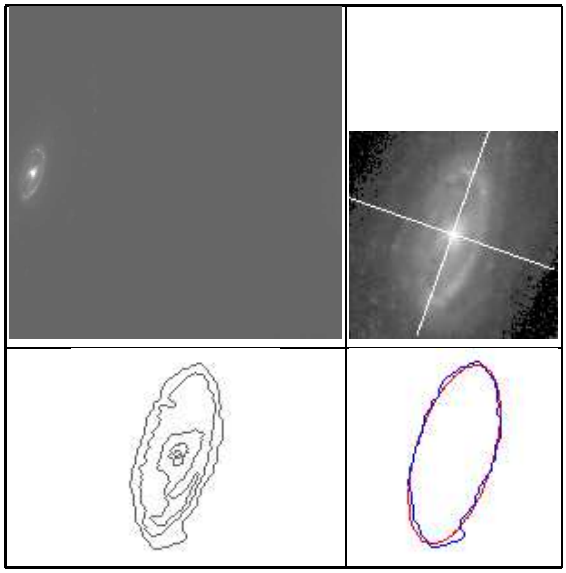


Fig. 16. Different steps of the processing of galaxy images

modular image processing programmes. The objective is to automate the detection of a galaxy in an astronomical image and to describe the shape of the galaxy (see figure 16) . The final result provided by these image processing programs is a morphological description of galaxies in terms of a set of numerical measures (see figure 17).

A first version of such a system [40] had been developed with the program supervision engine OCAPI [18]. Here, We present the knowledge base which has been developed with the program supervision engine PEGASE. It is a hierarchical skeleton based planner with a sophisticated repair mechanism. PEGASE is implemented with the LAMA platform [61]. For more details on the PEGASE program supervision engine see [62].

Knowledge base contents The last version of PROGAL is a large knowledge base. PROGAL contains 88 operators : 49 primitive operators corresponding to programmes and 39 complex operators expressing numerous abstraction levels. There are 30 complex operators with sequential decompositions and 9 complex operators with choice or alternative decompositions. PROGAL also contains 114 rules : 36 choice rules, 39 initialisation rules, 11 evaluation rules, 21 repair rules and 7 adjustments rules.

A common entity is shared by all these knowledge elements : the Context. This structured object, implemented by a frame (see figure 18) is used to express the possible variations in the contextual information for different galaxy images. This Context describes if the galaxy is a priori centered or

galaxy : *ngc6946*

area : *67886.7* ;
ellipticity : *0.35* ;
linear_err : *0.034* ;
profile : *1.96* ;
orientation : *86.46* ;
contour c1 : { centre_err : *0* ;
ellipse_err : *0.19* ;
compactness : *1.8* ;
angle : *-9.5* ;
excentricity : *0.22* };
contour c2 : { centre_err : *7* ;
ellipse_err : *0.37* ;
compactness : *8.1* ;
angle : *-4.4* ;
excentricity : *0.34* };
contour c3 : { centre_err : *3* ;
ellipse_err : *0.30* ;
compactness : *14.1* ;
angle : *50.4* ;
excentricity : *0.40* };
contour c4 : { centre_err : *12* ;
ellipse_err : *0.25* ;
compactness : *25.7* ;
angle : *55.4* ;
excentricity : *0.31* };
contour c5 : { centre_err : *20* ;
ellipse_err : *0.48* ;
compactness : *38.6* ;
angle : *84.2* ;
excentricity : *0.54* }.

Fig. 17. The output of the processing

not, if the image is noisy or not, if the image has been calibrated or not (density or intensity images), the minimal size of the objects in the image and finally if other objects as stars are a priori known in this region of the sky.

Context		
Attributes	Values	Comments
galaxy-position	*centered*	position of the object in the image
	uncentered	
min-size	[1, 1000000]	minimal size of the object
noise	*with*	noisy image or not
	without	
image-type	*intensity*	image type
	density	
stars	*in-front-of-the-galaxy*	eventual presence of other objects
	absent everywhere	
	outside-the-galaxy	

Fig. 18. Description of the contextual information

In the following we describe this knowledge base through two main aspects: planning knowledge and repair knowledge.

Planning knowledge Planning knowledge is mainly described with operators (primitive and complex) as well as choice and initialisation rules.

The global processing for galaxy description has eight sub-steps :

1. Creation and initialisation of the file containing the numerical parameters.
2. Isolation of the object of interest, the galaxy itself. This step is the most difficult one.
3. Determination of an object-centered coordinate space.
4. Global parameter computing. Once the galaxy is well isolated, parameters are extracted concerning global properties of the galaxy such as *orientation, barycenter, surface, length along the main direction...* (see Fig. 17).
5. Contour construction. In order to describe the galaxy in a multi-scale way, the processing builds five iso-intensity contours. The construction itself is a complex task which is decomposed in sub-tasks :
   - extraction of an *average profile* of the galaxy along its main axis
   - with this *average profile*, computing of five different thresholds allowing to build five different contours. This step depends on the context : if the input image is an **intensity** one, the thresholds are corrected.
   - thresholding the input image with the five computed thresholds
   - construction of five contour chains
   - an **optional step** allows the user to construct viewable files of the constructed contours. This step can be important in order to verify that the processing has been well done or not.
6. Contour parameter computing. For each of the five extracted contours, several parameters are extracted : orientation, surface, perimeter, length along the main and the minor axis, barycenter, distance between the contour and its approximation by an ellipse.
7. Visualization. Then, optionally, the visualization of the superimposed contours with the ellipses is possible by construction of viewable files (see Fig. 16).
8. Storing of the numerical parameters in a file

This decomposition is expressed in the knowledge base by a complex operator **c-galaxy-description** with a sequential decomposition into eight sub-steps as shown in figure 19.

Among these sub-steps three sub-steps (the first, the third and the last) are simple and are performed by a unique programme; so these three steps are respectively represented in the knowledge base by the primitive operators **p-initialisation**, **p-determination-of-axes** and **p-parameter-storing**.

The other five sub-steps are not achieved directly by one programme, so they are respectively represented in the knowledge base by five complex operators with sequential decompositions : **c-object-isolation**, **c-global-parameters**, **c-contours-construction**, **c-5contours-parameters** and **c-5ellipses-visualization**.

We now detail the planning knowledge for the most difficult part : *the isolation of the object of interest.* It is a very important step on which depends

Complex Operator

Name:	c-galaxy-description
Comment:	"image processing before classification"
Functionality:	galaxy-description
Input Data:	ie
Output Data:	gparam
Body:	
	p-initialisation -
	c-object-isolation -
	p-determination-of-axes -
	c-global-parameters -
	c-contours-construction -
	c-5contours-parameters -
	c-5ellipses-visualization -
	p-parameter-storing;

Fig. 19. The complex operator c-galaxy-description with a decomposition into 8 operators expressed in the Body attribute. The "-" sign between operators stands for a sequential separator "then".

the quality of the final result. The system must be able to take into account several conditions and must be able to evaluate the intermediate results of the processing and if necessary to adjust the treatment. This step is represented in the knowledge base by a complex operator o-object-isolation as shown in figure 20. It consists in two main steps which are localization of the center of the galaxy and effective isolation of the galaxy. The second step of effective isolation of the galaxy is done by calculating the limits of the galaxy followed by subtracting the background of the image.

Complex Operator

Name:	c-object-isolation
Comment:	"detection and extraction of the object"
Functionality:	object-isolation
Input Data:	ie
Output Data:	iisolee
Body:	
	c-object-location - c-effective-isol ;

Fig. 20. The complex operator c-object-isolation

The first substep (localization of the center of the galaxy) is the most difficult part. This step is described by a complex operator **c-object-location** with 5 substeps (see figure 21).

First, in order to reduce time processing, a **p-preprocessing** operator is applied depending on the size of the input image. An *optional criterion* is invoked to decide whether the image must be sampled or not. If the size of the image is bigger than a certain threshold (**contextual information**), the image is sampled.

```
Complex Operator
        Name:           c-object-location
        Comment:        " research of the object in an image "
        Functionality:  object-location
        Input Data:     ie
        Output Data:    is
        Body
                        p-preprocessing -
                        p-thresh-his-
                        c-galaxy-area-location -
                        c-max-conv -
                        p-center ;
```

Fig. 21. Complex operator c-object-location with a sequential decomposition into 5 operators

The third sub-step is **c-galaxy-area-location** as shown in figure 22.

```
Complex Operator
        Name:           c-galaxy-area-location
        Comment:        " galaxy location with two alternatives "
        Functionality:  galaxy-area-location
        Input Data:     ie
        Output Data:    is
        Body
                        c-uncentered-image | p-centered-image ;
```

Fig. 22. Complex operator c-galaxy-area-location with a decomposition into two alternative operators. The "|" sign stands for the alternative separator "or".

Depending on the context of utilization, *choice criteria* (see rules r-gc and r-guc in figure 23) allow the system to process the image in two possible ways.

- if the **galaxy is centered** in the image, a sub-image is extracted around the center of the image; a primitive operator **p-centered-image** represents this step in the knowledge base.
- if **we have no a priori information on the position** of the galaxy, the system looks for the most important object present in the image

```
Choice Rule
        Name : r-gc
        Let ?c a Context
        If              ?c.galaxy-position == 'centered
        Then            use-operator p-centered-image
Choice Rule
        Name : r-guc
        Let ?c a Context
        If              ?c.galaxy-position == 'uncentered
        Then            use-operator c-uncentered-image
```

Fig. 23. Choice rules to select between operators for galaxy area location

(biggest extended source). This is **the most difficult part**, a complex operator **c-uncentered-image** represents this step in the knowledge base (see figure 24). This complex operator has 3 substeps **among which c-detection** (see figure 25).

```
Complex Operator
        Name:           c-uncentered-image
        Comment:        " research of the biggest object in an image "
        Functionality:  galaxy-area-location
        Input Data:     ie
        Output Data:    is
        Body
                        c-detection - p-extraction - p-mu ;
```

Fig. 24. Complex operator c-uncentered-image with a sequential decomposition into 3 operators

We will se in next section on repair knowledge 3.3 how we can check the quality of the galaxy detection.

Repair knowledge Repair knowledge is mainly described with evaluation, repair and adjustment rules. In this section we will present how the knowledge for error detection and recovery is expressed. For the complete processing there are mainly three types of errors which can be recovered. As is often the case in image processing, the detection of an error is not possible immediately after the execution of the "guilty" operator. In Fig. 26 we see a synthetic view of the complete processing. The first case of error recovery occurs when the size of the detected object is similar to the size of a star. The second case occurs when the galaxy has not been correctly isolated. This case is easy to

Complex Operator	
Name:	c-detection
Comment:	" detection of the brightest region"
Functionality:	galaxy-detection
Input Data:	ie
	sm
Output Data:	is
	taille
	...
Body:	
	p-thresh-muls - p-muls - c-morphlis - c-contour-chain - p-bbox;
Repair rules:	rule-5

Fig. 25. The complex operator c-detection

repair (enlargement of the estimated size of the galaxy) but the error can only be detected at the end of the processing.

Finally, the last case occurs when the five iso-contours overlap.

We will detail how the knowledge is expressed for the first case (bad galaxy detection).

Operator p-bbox The operator **p-bbox** (see figure 27) computes the bounding box of the detected object. Several evaluation criteria : rule-1 (see figure 28), rule-2 (see figure 29) and rule-3 (see figure 30) attached to this operator compare the size of this bounding box (an output of this operator) with the normal size of the stars in this field (a value provided in the context). If the evaluation is ambiguous, the repair knowledge sends this information to the complex operator **c-detection** using the transmission function **send-up** (see rule rule-4 in figure 31).

Operator c-detection If an error is sent to the operator **c-detection** (already shown in figure 25 in Planning Knowledge section), PEGASE uses the repair rule base attached to the operator **c-detection** to repair it. In this case, if the error is *size-ambiguous* (sent by **p-bbox**), the error recovery is simply to send it to one of its suboperators **p-muls** using the function **send-down** (see rule rule-5 in figure 32).

Operator p-muls The operator **p-muls** (see figure 33) contains a repair rule and an adjustment rule. When the error *size-ambiguous*, the result assessment performed after evaluation of the results of **p-bbox**, is received by **p-muls**, the error strategy expressed in the repair rule rule-6 (see figure34) is to re-execute this operator. The function **re-execute** triggers the adjustment rule base attached to this operator to compute a new value for the threshold smuls. The adjustment method (see rule rule-7 in figure 35) is an adjustment

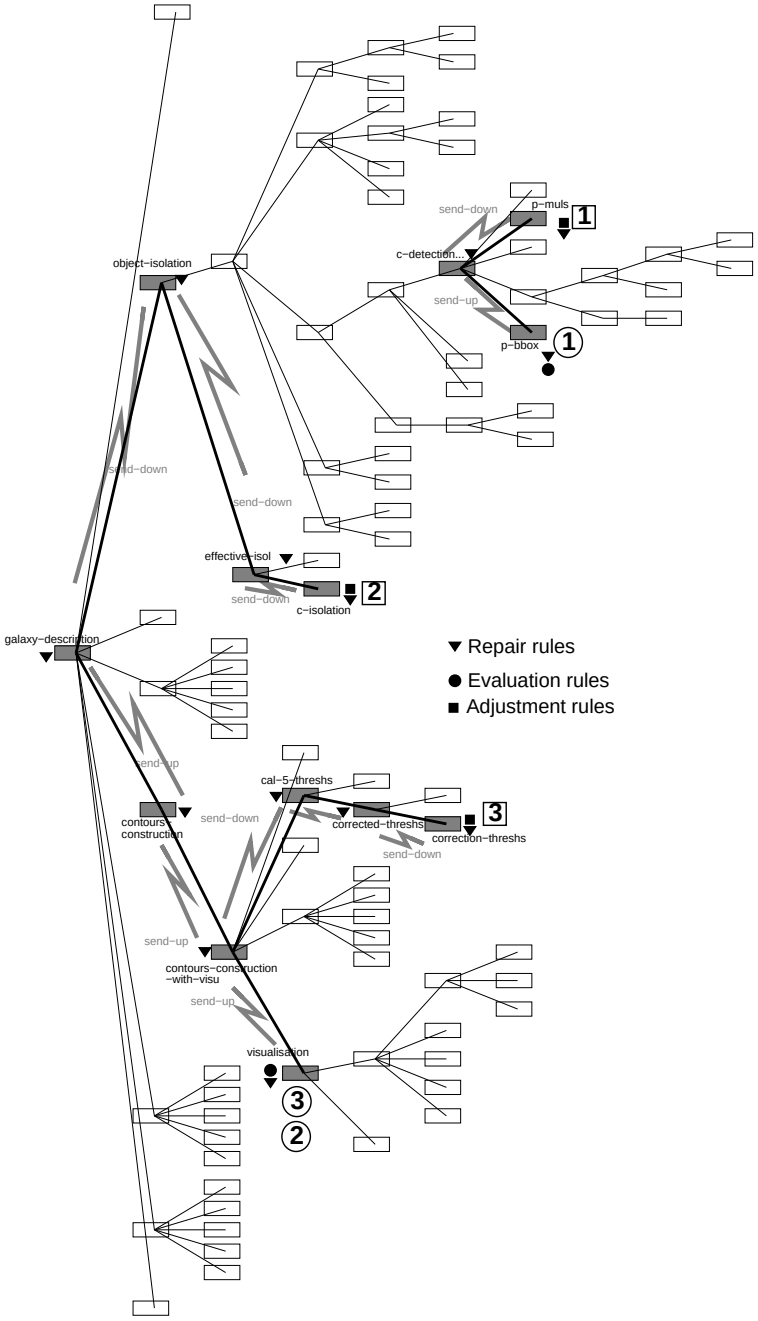


Fig. 26. A tree showing the decomposition of the complex operator chaine-astro into 7 sub-steps. The steps being decomposed into other sub-steps. The leaf of the tree are concrete programs (or primitive operators). Evaluation, repair and adjustment rules are located in the tree.

Primitive Operator	
Name:	p-bbox
Comment:	" computing the bounding box of a region"
Functionality:	bounding-box
Input Data:	ie
	sm
Output Data:	ix
	iy
	x
	y
	size
	...
Evaluation rules:	rule-1, rule-2, rule-3
Repair rules:	rule-4

Fig. 27. The primitive operator p-bbox

Evaluation Rule	
Name : rule-1	
Comment :	"If the object size is really larger than the size of a star, the detection is good"
Let	*?context* a **Context**
If	size $>$ *?context*.star-size +(*?context*.star-size / 2)
Then	**assess-operator** good **continue**

Fig. 28. Evaluation rule attached to operator p-bbox deciding positive assessment

Evaluation Rule	
Name : rule-2	
Comment :	"If the object size is slightly larger than the size of a star, the detection is limit
Let	*?context* a **Context**
If	size $<$ *?context*.star-size + (*?context*.star-size / 2), size $>$ *?context*.star-size
Then	**assess-operator** limit **continue**

Fig. 29. Evaluation rule attached to operator p-bbox deciding that detection is limit

Evaluation Rule	
Name : rule-3	
Comment :	"If the object size is smaller or equal to the size of a star, the detection is ambiguous "
Let	*?context* a **Context**
If	size $<$ *?context*.star.size
Then	**assess-operator** ambiguous **repair**

Fig. 30. Evaluation rule attached to operator p-bbox deciding that detection is ambiguous

Repair Rule	
Name : rule-4	
Comment :	"If the detection is ambiguous, send-up"
If	**assess-operator?** p-bbox ambiguous
Then	**send-up** size-ambiguous

Fig. 31. Repair rule attached to operator p-bbox deciding to send up a negative assessment

Repair Rule	
Name : rule-5	
Comment :	"If the detection is ambiguous, send-down p-muls"
If	**assess-operator?** c-detection size-ambiguous
Then	**send-down** p-muls size-ambiguous

Fig. 32. Repair rule attached to operator c-detection deciding to send down a negative assessment

per percentage with a step value of 0.05 %. As the goal is to increase the size of the detected object in the binary image, the threshold has to be decreased.

Primitive Operator	
Name:	p-muls
Comment:	"binarization of an image"
Functionality:	thresholding
Input Data:	ie
Parameters:	smuls
Output Data:	is
	...
Repair rules:	rule-6
Adjustment rules:	rule-7

Fig. 33. The primitive operator p-muls

Repair Rule	
Name : rule-6	
Comment :	"If the error is size-ambiguous, re-execute"
If	**assess-operator?** p-muls size-ambiguous
Then	**re-execute**

Fig. 34. Repair rule attached to operator p-muls deciding to re-excute the operator p-muls

Adjustment Rule	
Name : rule-7	
Comment :	"Decrease threshold smuls of five per cent"
If	**assess-operator?** p-muls size-ambiguous
Then	**adjustment-method** smuls **percent-float**,
	adjustment-step smuls .05,
	decrease smuls

Fig. 35. Adjustment rule attached to operator p-muls deciding to modify the value of parameter smuls

The complete treatment can be viewed as a hierarchical processing tree (see Fig. 26).

Conclusion We have presented PROGAL, a program supervision system which allows the expert to express his/her knowledge in a natural way. A main advantage of a knowledge based system is that it allows to capitalize knowledge : new processing methods can be added or the knowledge can easily be updated.

Furthermore, this system allows to make dynamic processing in the sense that it adapts itself to different situations that can occur during the processing of galaxy images. Thus, this system is a flexible tool able to process different kinds of images taken in different conditions.

3.4 Conclusion

We have presented knowledge-based techniques for building intelligent image processing systems : program supervison techniques. These techniques make it possible both to capitalize the knowledge of the use of a set of programmes in a formalized knowledge base and to build flexible and automatic systems with dedicated reasoning mechanisms. Program supervision systems can be developed when several conditions are verified: first, the existence of a set of modular programs; second, the need of a complex and flexible use

of these programs (different input parameters values, different selections of programs,...); third, the existence of a set of explicit criteria to decide what to do (knowledge of typical decompositions, evaluation criteria, repair strategies,...).

4 Image Understanding

In this section we study two different image understanding problems: first, we are interested in the problem of single complex object recognition on static images then we consider the problem of scenario recognition on video sequences.

4.1 Object Recognition

Introduction Our goal is to find the class of a structured object using predefined taxonomy. We propose to perform the numerical description of the objects with vision algorithms and the classification with a knowledge-base system dedicated to object classification. The computation of the numerical description of the objects can be controlled by program supervision techniques as explained in the previous section. In this section we are interested in the interpretation of the numerical description in order to class the object. The objects we consider are complex and structured objects. We make the hypothesis that the objects are isolated non overlapping objects. One class usually has several appearances in the image. For instance the object is a 3D object but the input data are extracted from a unique 2D image. Another frequent case is a non-rigid object (an animal or a human being) with varying distance, size, position or postures. The second hypothesis we make is that the classes are known and that symbolic and numerical description of prototypical objects belonging to each class is possible.

We have applied the same object recognition method for different domains: in astronomy we have studied the classification of galaxies according to their morphology [50]. The objects are luminous 3D objects with no clear boundaries, various orientations, distances, sizes. The input is the numerical description of a galaxy extracted from a unique 2D image with a galaxy plus stars (see section 3 for examples). The classes are function of morphological and photometrical models. In biology we have studied the recognition of zooplanktons [53]. The objects are 3D living organisms more or less transparent with various orientations, positions, states of maturations and articulations (non rigid). The input image is a unique 2D image of a zooplankton in aquous environment. The classes correspond to biological taxonomy.

In the following we first present the classification engine CLASSIC then we detail an example for planktonic foraminifera classification ([64] and [33]).

Classification engine The role of a knowledge-based system for object recognition is to interpret the data obtained by the image analysis algorithms and to find the class (or classes) that the object belongs to. This role is different than the role of program supervision engines like OCAPI or PEGASE described in section 3 which are used to control the image analysis algorithms. We build object recognition systems using a common engine *CLASSIC*. This engine is designed for classification and diagnosis problems. It was initially built for galaxy classification ([27,54]), and has been successfully used for other applications as the classification of zooplanktons ([53]), the diagnosis of antenna failures ([41]).

Knowledge representation

CLASSIC organizes knowledge through two kinds of knowledge representation schemes: prototypes and rules. The prototypes are used to describe in an explicit way models of various classes of the objects to be classified. These prototypes are implemented by frames with attributes and predefined slots. The values of the attributes are symbolic, numerical or even frames. The prototypes are organized in a tree (prototype trees), which reflects the hierarchy of the classes. In addition to that, deductive rules are used to perform data abstraction. These rules are implemented by production rules.

Imprecise and uncertain knowledge are represented using results from possibility theory and fuzzy set theory. An imprecise fact is characterized by a possibility distribution which corresponds to the characteristic function of a fuzzy set. A uncertain fact is characterized by a confidence factor (called a possibility measure) and a doubt factor (which is the certainty of the negation).

Reasoning

The reasoning in CLASSIC is accomplished by repeatedly passing through three phases: data abstraction, matching with the prototype, and refinement of the classification:

- Data abstraction phase allows the passage from measures obtained by the image analysis algorithms to qualitative textual descriptions used by experts and to other measures necessary for automatic classification. This phase is performed mainly by the production rules.
- During the matching phase, the object to be classified is compared to a prototype. The comparison is performed in two stages:
  - (1) Calculate the compatibility coefficient and the incompatibility coefficient of the prototype, on the basis of the combination of the compatibility coefficient and the incompatibility coefficient of each slots of the prototype.
  - (2) Accept or reject the prototype by comparing these two coefficients to the compatibility threshold and the incompatibility threshold.
- The role of the refinement phase is to find a more precise classification for the object based on its current status of descriptions.

During the classification process, the object to be classified is compared to each node in the prototype tree, from the root to the leaves, in order to find the class which corresponds the best to the object. Each class in the hierarchy is associated with a rule base. This rule base is invoked when the object is compatible with that class in order to complete its description. When no more rule provides any new information, the comparison goes on to the next node, and so on.

Classification of planktonic foraminifera We have used the classification engine CLASSIC for automatic classification of planktonic foraminifera.

Foraminifera

Foraminifera are single-celled micro-organisms, enclosed within a mineralized shell (test). They live either on the sea floor (benthic foraminifera) or amongst the marine plankton (planktonic foraminifera) [9]. Their size is mostly less than $1mm$. Being the most important group of the microfossils (about 2000 genera and 30000 species), the foraminifera are presently the best known and most comprehensively studied of all the calcareous microfossils. They constitute invaluable indicators in the age determination as well as the depositional environment of sedimentary strata.

In oil exploration, the identification of foraminifera is an important task. Their identities can specify the likelihood and the quality of oil to be found. However it is impossible to have microfossil experts available wherever their knowledge is needed. It may happen that oil companies spend a million dollars a day to keep rigs operating in waiting for the expertise results from the research center [48]. Moreover, the identification of foraminifera has always been a tedious and time-consuming task for micropaleontologists.

Therefore, it is desirable that some "intelligent" computer systems could be designed to accelerate microfossil identification. Several attempts have been made at developing such computer systems [11,21,43,48]. In particular, the system Vides [48] developed by British Petroleum aims at speeding up microfossils classification for oil drilling companies. However, all these systems are interactive so that human intervention is needed during the identification process. Especially, the descriptions of the specimens to be identified have to be provided by the user.

Automatic Classification of Planktonic Foraminifera

At the current stage of our work, we focus on the identification of some species of the group Globotruncanids, which is representative of the planktonic foraminifera with trochospiral coiling of the late Cretaceous time (cf. [13,44]). In particular, we have tested the following species: *Globotruncanita calcarata* (Cushman, 1927), *Globotruncanita stuarti* (De Lapparent, 1918), *Rosita contusa* (Cushman, 1926), *Rosita fornicata* (Plummer, 1931), *Abathomphalus intermedius* (Bolli, 1951), *Abathomphalus mayaroensis* (Bolli, 1951), *Rugoglobigerina rotundata* (Bronnimann, 1951), *Gansserina gansseri* (Bolli, 1951). Although they belong to the same type (trochospiral) of chamber

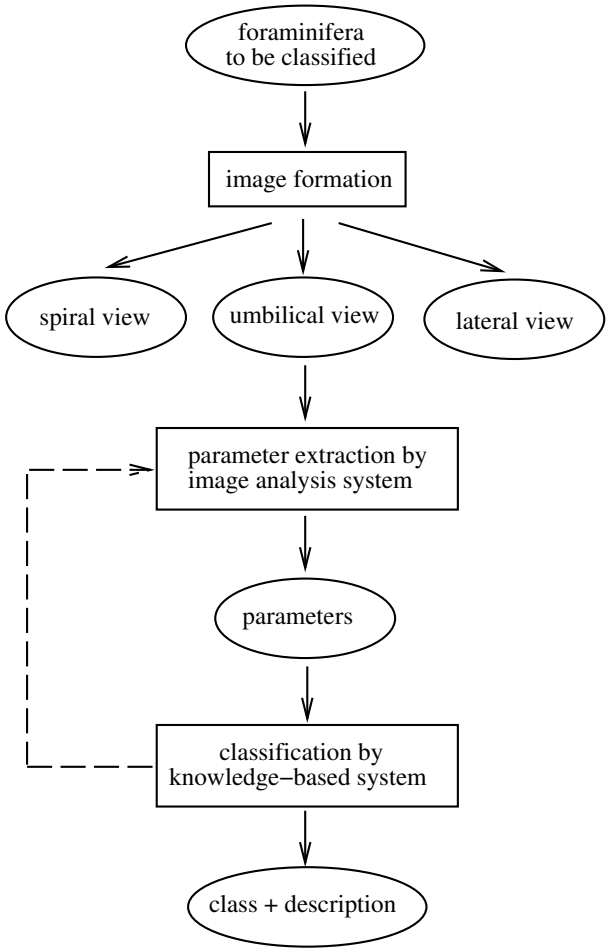


Fig. 36. Schema of automatic classification of planktonic foraminifera.

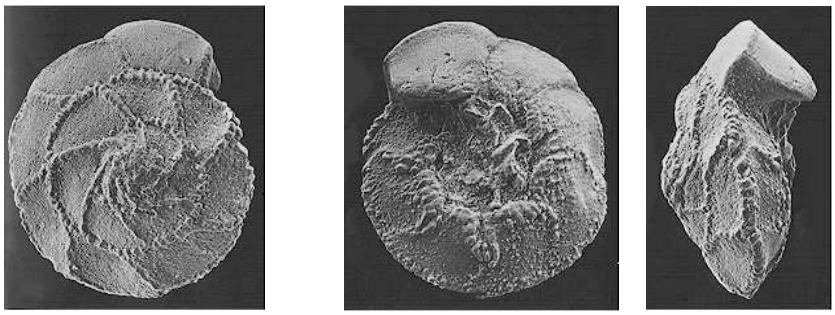
arrangement, these species have been chosen for their wide range of morphological variations.

Figure 36 illustrates the structure of our system. It is composed of two subsystems: an image analysis system for shape description and a knowledge-based system for classification. The dotted line indicates possible guidance of the image analysis process by the knowledge-based system.

The identification process is made automatic owing to the image analysis system. It is a hard task to construct such an automatic system. On one hand, foraminifera are complex objects with many morphological variations, and often broken or covered with sediments. Description of their shapes is by no means easy, sometimes difficult even for a human. On the other hand, the description terms as well as the classification criteria used by micropaleon-

tologists are given in natural language; whereas the automatic identification processes are necessarily based on quantitative measures. The transformation of the qualitative textual descriptions to quantitative measures is nontrivial.

We have been able to overcome some of the difficulties by proposing various geometric parameters to characterize the morphological features of the foraminifera, and by developing a set of image analysis algorithms to obtain these parameters. We also have designed a set of rules in the knowledge-based system in order to relate the description parameters we proposed to the traditional terminology.



(a) (b) (c)

Fig. 37. A specimen to be identified (×100). (a) spiral view. (b) umbilical view. (c) profile view.

In our system, the identification of a foraminiferon is based on the shape descriptions of its three characteristic views: the spiral view, the umbilical view, and the lateral view (cf. Figure 37). Images of these three views are processed by the image analysis system which produces a set of measures describing the morphological features of the specimen. The knowledge-based system is then activated and classifies the specimen into an appropriate class using these measures.

The image analysis system allows us to obtain most of the descriptive parameters that are used by the micropaleontologists for foraminifera identification. However, some of the features, in particular the umbilical structures, are very difficult to describe automatically by computer vision system. In fact, their description is sometimes nontrivial even for experts. Automatic description of these features remains unsolved at present.

Owing to the fact that our knowledge-based system does not use sequential reasoning as in the traditional taxonomic way, the classification process can be carried out even with some values "unknown", though the classifica-

tion precision might be reduced. In fact, the traditional taxonomic method uses fixed decision trees, whereas our system makes decisions by matching the description of the object to be classified with the prototype description tree (see next section). Moreover, our knowledge-based system also has an interactive mode to allow users to enter feature values, when necessary, during the classification process. Details about our image analysis system can be found in [34].

Knowledge Base Contents

The knowledge base, consisting of a prototype base and a rule base, is built up according to the description terms and the classification criteria used by the experts in the field. As the computed parameters describing the shape of the foraminifera are mostly different from the terms used by the experts, we need rules to establish links between these parameters and the description terms used in the microfossil field.

During the construction of the system we had to describe the characteristics of each known object using both the terminology used by the experts and the parameters suitable for computer extraction. All the prototypes are organized in a hierarchical tree. Production rules are used to derive new data from the input data with the aim of completing the description of the objects.

Prototype Base

The prototypes are frame-like objects. Both symbolic and numerical values are used to describe the characteristics of the classes.

Figure 38 illustrates the prototype tree of group Globotruncanids. It directly reflects the hierarchy of the classes. The nodes of intermediate levels and the leaves correspond respectively to the genera and the species of the group.

The organization of prototypes into a hierarchical tree provides a convenient way both for knowledge structuring and for reasoning control. Indeed, due to the inheritance of descriptions from father nodes to their sons, redundancy is eliminated, and modularity is enhanced. The hierarchical organization of the knowledge also defines natural reasoning steps and permits efficient control.

The prototype Globotruncanita is shown below:

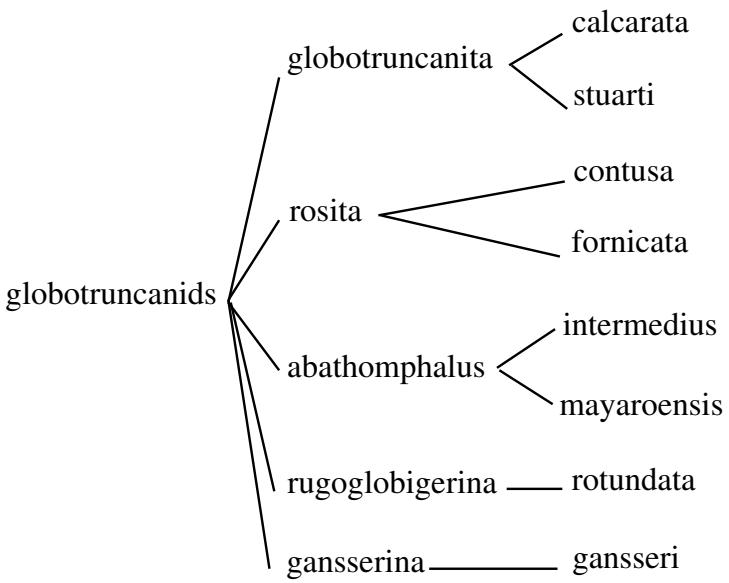


Fig. 38. The hierarchy prototype tree of group Globotruncanids.

Prototype	**Globotruncanita:**
superclass:	*globotruncanides*
contour:	*{circular spinal}*
chamber-nb:	*[5, 9]*
spi-chamber-form:	*{trapezoidal}*
spi-chamber-ornement:	*{perforated, rugose}*
spi-suture-insert:	*{oblique slightly-oblique perpendicular}*
umb-chamber-ornement:	*{perforated pustulous}*
primary-aperture:	*{umbilical}*
accessory:	*{portici tegilla}*
p-spi-form:	*{convex slightly-convex flat}*
p-umb-form:	*{very-convex convex}*
keel-nb:	*[1, 1]*
keel-band:	*{absent}*

The prototype Stuarti is shown below:

Prototype	**stuarti**
superclass:	globotruncanita
contour:	circular
chamber-nb:	[7, 9]
spi-suture-width:	thick
spi-suture-depression:	non-depressed
spi-suture-insert:	slightly-oblique perpendicular
umb-chamber-ornement:	perforated
accessory-aperture:	present
p-spi-form:	convex
p-umb-form:	convex
p-symmetry:	symmetric

Rule Base Production rules are used to complete the description of the objects to be classified. Particularly, they are used to relate the parameters obtained by our image analysis algorithms to the description terms used by the experts. They will also be used, in the future development, to invoke image analysis procedures depending on the context, as in [53].

Rules are used in a dynamic mode, either to transform numerical data to symbolic ones, like:

rule	"spi-chamber-form-trapezoidal "
if	spi-chamber-compa <= thresh.spi-chamber-compa
	spi-chamber-a-c <= thresh.spi-chamber-a-c
then	spi-chamber-form = trapezoidal
comment	For a chamber on the spiral side, if both its compactness and the ratio of its arc to its chord are small, then its form is trapezoidal.

or to increase the abstract level of symbolic data, like:

rule	"p-symmetry"
if	p-spi-form is known
	p-umb-form is known
	p-spi-form = p-v-form
then	p-symmetry = symmetric
comment	Viewed from the profile of a specimen, if the form of its spiral side is the same as that of its umbilical side, then the object has a symmetric profile.

In our rule base, fuzzy predicates like $\sim=$, $\sim<$, *close to*, $far\ from$, are also used to manipulate imprecise data. The assignment of appropriate rules to each node of the prototype tree is performed automatically by the system engine.

Fact Base

Facts represent the descriptive information about the object to be classified. For our application, the initial fact base contains the data obtained by

our image analysis algorithms. Deduced data are dynamically added during the reasoning. We refer the reader to the next section for examples of initial data and of deduced ones.

surface:	*area of the spiral/umbilical sides*
diameter:	*diameter of the spiral/umbilical sides*
perimeter:	*perimeter of the spiral/umbilical sides*
peak-nb:	*number of peaks on the distance curve*
peak-form:	*shape of the peaks on the distance curve*
ch-vertices:	*number of vertices on the convex hull*
ch-points:	*total number of points on the convex hull*
chamber-nb:	*number of chambers on the last whorl*
spi-chamber-surface:	*surface of a chamber (spiral side)*
spi-chamber-perimeter:	*perimeter of a chamber (spiral side)*
spi-chamber-chord:	*length of a chord of a chamber (spiral side)*
spi-chamber-arc:	*length of the external arc of a chamber (spiral side)*
spi-chamber-height:	*height of a chamber (spiral side)*
spi-chamber-ornement:	*ornement of a chamber (spiral side)*
spi-suture-width:	*width of the suture (spiral side)*
spi-suture-depression:	*depression of the suture (spiral side)*
spi-suture-angle:	*junction angle of the suture and the external contour*
umb-chamber-ornement:	*ornement of a chamber (umbilical side)*
maj-axis:	*length of the major axis of the lateral side*
min-axis:	*length of the minor axis of the lateral side*
trocho-spi:	*width of the spiral side*

Working Session The classification of foraminifera is performed as follows: Given images of the three views of the specimen (see Figure 37), the image analysis algorithms are applied to obtain initial descriptive parameters of the object. The knowledge-based system takes these results as input data and searches the hierarchy tree of the prototypes to find the most compatible class (or classes). Related rules are invoked during the reasoning process to complete the descriptions of the object.

Just before activating the classification, the control system verifies whether all the required parameters have a value. If not, the system will first let the operator specify the missing values. Otherwise, the system will consider these values"unknown".

The working session record is shown below:

Initial description of	p31-3-2
surface:	31369
diameter:	216
perimeter:	598
peak-nb:	1
peak-form:	concave
ch-vertices:	60
ch-point:	642
chamber-nb:	7.5
spi-chamber-surface:	2828
spi-chamber-perimeter:	252
spi-chamber-chord:	76
spi-chamber-arc:	80
spi-chamber-height:	5
spi-chamber-ornement:	perforated
spi-sutu-width:	thick
spi-sutu-depression:	non-depressed
spi-sutu-angle:	72
umb-chamber-ornement:	perforated
maj-axis:	208
min-axis:	102
trocho-spi:	64

The system checks that the input parameters are compatible with the descriptions of the prototype Globotruncanids and validates it.

The system scans the base of production rules attached to prototype Globotruncanids and activates those for which the premises are verified:
Rule spi-sutu-r-angle
(determining that sutures on the spiral side join the spiral suture almost at right angle)
Rule p-umb-form-convex
(determining that the profile has a convex umbilical side)
Rule p-spi-form-convex
(determining that the profile has a convex spiral side)
Rule spi-chamber-form-trapezoidal
(determining that chambers on the spiral side are trapezoidal)
Rule cont-circu
(determining that the outline is circular)
Rule p-symmetry
(determining that the profile is symmetrical)

According to the new values added in the fact base by the rules, the system chooses among the subclasses of Globotruncanids those which are compatible with the specimen p31-3-2. So the prototype Globotruncanita is validated.

Then, the system validates the prototype Stuarti.

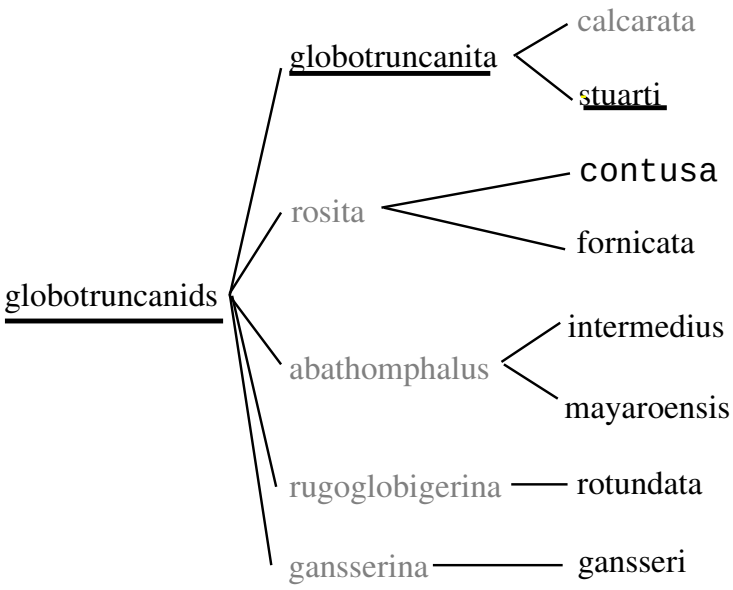


Fig. 39. The result of the classification of a specimen. The underlined nodes correspond to validated nodes. The grey nodes correspond to incompatible nodes. The other nodes are nonexploited nodes.

Final description of	p31-3-2
class:	stuarti
compactness:	11.39992
ch-v-over-p:	.09345794
spi-chamber-compa:	22.45544
spi-chamber-a-c:	1.052632
spi-chamber-h-over-c:	.06578948
half-maj-axis:	104
trocho-umb:	38
trocho-spi-over-hm:	.6153846
trocho-umb-over-hm:	.3653846
thresholds:	thresholds
contour:	circular
spi-chamber-form:	trapezoidal
spi-suture-insert:	slightly-oblique
p-spi-form:	convex
p-umb-form:	convex
p-symmetry:	symmetric

Figure 39 displays the classification result of the specimen shown in Figure 37. The deep-grey nodes correspond to validated nodes. The light-grey nodes correspond to incompatible nodes. Others correspond to the nonexploited nodes. The species to be identified is thus classed as an organism of group Globotruncanids, genus Globotruncanita, and species Stuarti.

Conclusion We have presented a knowledge-based technique for object recognition. First an engine dedicated to classification reasoning has been presented. Then a complex application in micropaleontoly for planktonic foraminifera identification has been detailed. An overview of the method and especially the knowledge-based system have been discussed. This system has been applied to the automatic identification of several important species of the group Globotruncanids of late Cretaceous time. Automating such a task has at least two main advantages: (1) The process of identification is much faster than the interactive systems; (2) An interactive identification is subjective and may yield in different results according to operators; whereas automatic identification uses unified objective criteria and therefore is more consistent. Owing to the fact that in this object recognition approach the knowledge base is separated from the control structure, modifications in the knowledge base can be carried out easily.

4.2 Scenario Recognition

The second image understanding problem we consider in this section is scenario recognition on video sequences. We propose a framework based on two kinds of a priori knowledge : predefined scenarios and 3D scene model. This approach has been applied on video sequences of the AVS-PV visual surveillance european project.

Introduction Our goal is to detect mobile objects (specially people) and to analyze their behavior. This work is based on three hypotheses: first we consider a static camera, second we use a unique monocular camera and third we deal with real-time constraints. The first hypothesis (static camera) is often verified for current visual surveillance networks and allows us to simplify the low-level detection of mobile objects w.r.t. a fixed environment. The second hypothesis (unique monocular camera) is verified in almost all current visual surveillance networks. The third hypothesis (real-time constraints) is very interesting as it implies that the solutions should be obtained with a short computing time, i.e. shorter than the time frequency between two consecutive images of the sequence. But it also implies a fully automated system. After a presentation of related work, we briefly present the current low-level image processing techniques used for mobile object detection and tracking. Then two kinds of a priori knowledge we use are described. First we describe the

role of a priori 3D scene model and different ways of representing this information. Second we adress the problem of high-level description of mobile object behavior using generic observable events and application-dependent scenarios. Finally, results obtained on different visual surveillance applications in the european Esprit project AVS-PV are shown and discussed.

Related work Cohen, Bremond Medioni and Nevatia (University of South California), in DARPA's VSAM focus on event recognition involving vehicles and humans ([36] and [20]). The particularity of this work is that videos are filmed by non-fixed cameras. They used models of maps of the environment to place aerial images in an *a priori* known map and a property net to compute events and states involved in predefined automata describing situations. Herzog (VITRA) proposes a system able to dynamically describe scenes with humans. The originality of his work is the application environment: a soccer stadium ([1] and [28]) and the inference method based on time interval logic, to describe temporal sequence of events, which are computed and typed separately. Intille and Bobick (MIT Media lab), in a similar environment, focus on analysis of American football scenes. Their aim is the recognition of particular strategies in complex players' interactions ([29] and [30]). The main point is that those activities are not just human behaviors but human group behavior. Shah (University of Central Florida) is interested by dynamic description of human behaviors in office environments ([2] and [23]). Even if the problem is the recognition of long duration activities, the authors insist on the importance of the recognition of 'key instants' which are the conditions of changing states in an automaton representing the global behavior. The "Key instants" are generated when certain conditions are realised.

Tessier (ONERA), in the PERCEPTION project, proposes an original method to describe behavior. Petri nets are used to represent dynamic evolutions of a car park scene with humans and vehicles ([49] and [14]). Buxton and Gong (University of Sussex) made an important contribution to the domain with the VIEWS project ([12]). The system was able to deal with humans and vehicles on streets or in car parks. A high level representation based on Bayesian networks was computed. This work points out the necessity to deal with uncertainty and to use contextual information to enhance detection and tracking results. In the same vein, Ivanov and Grimson (MIT) work on detection of human and vehicle behaviors in a car park. The interest of this research is in the event combination method ([31]). A behavior is represented by a set of rules based on a stochastic context-free grammar, which allows certain combinations of simple constant predicates. The general scheme of our approach is based on the use of predefined scenarios [17] and scene model [16]. In this section we propose a method based on trees to declare events [57] and on temporal logics to declare application dependent scenarios.

Perception

In this section we will shortly present the perception component we have used. A more detailed description can be found in [45]. The perception methods are standard ones which verify the real-time constraint hypothesis. Their role is to incrementally provide a history of the persons who have been detected in the scene. It is composed of four main sub-parts: motion detection, person detection, person tracking and smoothing. Each sub-part contains alternative methods which are manually selected and parametrized in a configuration phase.

Motion detection

Motion detection is basically a thresholding of the difference of the current image with respect to a reference image. Before the difference is computed we filter the current image with a 3×3 gaussian filter to reduce noise. Then for each pixel we compute the absolute difference between its intensity (grey or colour) and the intensity of the corresponding pixel in the reference image. If this difference is greater than a certain threshold α, the pixel is marked as mobile and otherwise it's marked as stationary. We then update the reference image I_r with information from the current image I_c according to the following equation:

$$I_r = (1 - \beta)\, I_r + \beta I_c$$

We see that for $\beta = 1$ we detect motion as the difference between subsequent images in the sequence and that for $\beta = 0$ motion is detected with respect to a fixed background image. α and β are parameters of the motion detector.

People detection The goal is to detect which mobile regions (blobs) correspond to a person. We use a model of a person with 8 parameters: the position of the center of gravity $(px_{img},\ py_{img})$, the height h_{img} and the width l_{img} in the $2D$ image, the $3D$ position $(px_{3D},\ py_{3D})$ on the ground plane of the scene, the $3D$ width l_{3D} and the $3D$ size h_{3D}. The bounding box of a person is defined by the image measures $(px_{img},\ py_{img})$, h_{img} et l_{img}. The people detection algorithm analyzes the set of blobs. Both $2D$ image criteria and $3D$ scene criteria are used. The first ones are based on the $2D$ distance between blobs in the image. The goal is to merge the closest blobs in the image. The second ones are based on constraints on the $3D$ height and width. The $3D$ measures are obtained by linear projection of the image plane.

People tracking

The goal of this step is to update the set of trajectories. For that purpose, the persons who have been detected in the current image must be matched with those detected in the previous ones. This matching can be defined as a function from the set P_{t-1} of persons detected a time $t-1$ into the set P_t of the persons detected a time t. We use 3 alternative methods: a method based on the amount of overlap in the $2D$ image, a method based on the proximity of the persons in the $3D$ scene and a restrictive method based on the proximity of the persons in the $3D$ scene. The first method (based on the amount of overlap in the $2D$ image) states that two persons detected at

two consecutive times are the same real person if the percentage of overlap of their bounding box is greater than a threshold. The second method matches a person at time t with a person at time $t-1$ if their $3D$ distance is below a threshold. The third method is similar to the second one, but the function must be either an injection or a surjection.

Smoothing

The first goal of this step is to correct errors made in the previous perception steps on the different $3D$ measures of a person: $(px_{3D},\ py_{3D})$ the position on the ground plane, h_{img} the height and l_{img} the width. The second goal is to estimate $(vx_{3D},\ vy_{3D})$ the instantaneous speed of the persons. Three smoothing methods are used. The first method uses a standard Kalman filter. The state vector is defined by $(px_{3D},\ py_{3D},\ vx_{3D},\ vy_{3D})$. The linear dynamic model is based on the hypothesis of a constant speed. The second and third methods are respectively median and mean filtering with temporal window size 3, 5 or 7 . $(vx_{3D},\ vy_{3D})$ is initialised by computing $v(t) = \frac{p(t)-p(t-1)}{\delta t}$ then each of the four values px_{3D}, py_{3D}, vx_{3D} and vy_{3D} are filtered.

3D scene model As our goal is to provide a framework which can be adapted to specific conditions we propose to define two kinds of a priori informations : 3D scene model (see this section) and predefined scenarios (see section 5). 3D scene model is a priori information which contains a description of the static environment observed by a camera. For each camera looking at a particular environment, a security operator must provide, in a configuration phase, pertinent information according to the formalism we propose. For more details on the role of scene model in video understanding see [16]. The 3D scene model contains in addition to geometric information some semantic information. Its structure is made of a set physical objects, a set of interesting areas and a calibration matrix for the transformation from the 2D image plane to 3D coordinates.

The geometry of the interesting areas is described by a list of polygons defined in planes which may have any orientation. The geometry of each physical object, or piece of equipment, is a generalized cylinder defined by its height and its polygonal basis.

The semantic information of each piece of equipment and of each interesting area is made of six attributes : four with symbolic values and two with numerical values. The four attributes with symbolic values are : the type (equipment or area), the function (i.e. table, seat, corridor, etc...) the name (i.e. seat3, corridor2, etc...) and characteristics (i.e. yellow, fragile, etc...). The two attributes with numerical values which are very useful for scenario recognition are: the normal distance and the normal time of usage of an equipment.

Figures 40 and 41 show two examples of environments we have modeled; for each example the left image shows the view observed from the camera and the right image shows a 3D view of the same environment based on the

geometrical information contained in the model. In the first example, a coffee room, the 3D scene model contains: the calibration matrix, the description of nine pieces of equipment (three seats, one table, one coffee machine, one elevator, one dustbin, one door, and one heater) and the description of three areas (a seat area, an entrance, and a corridor). The second example is a real scene of a metro station in Nuremberg which has been selected in the european project AVS-PV. It is an entrance of a metro statio with eight equipments and two areas. The equipment are six turnstiles and two ticket vending machines. The areas are an entrance and a corridor.

Figure 42 shows an example of the complete description of a ticket vending machine for a metro station in Nuremberg.

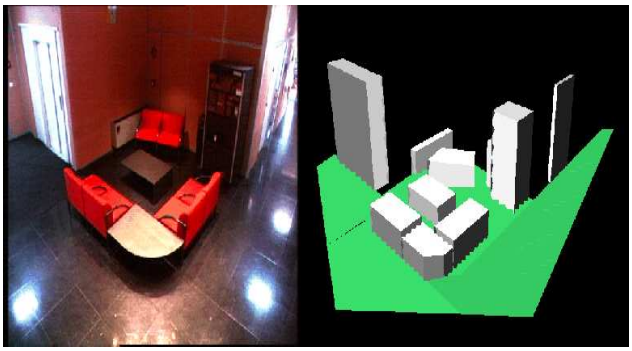


Fig. 40. Left: image of a coffee room. Right: $3D$ scene model.

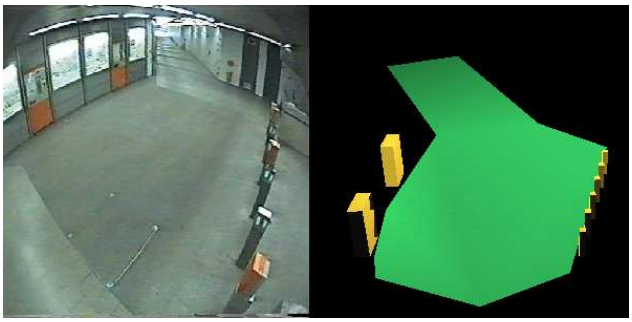


Fig. 41. Left: image of a metro station. Right: $3D$ scene model.

The contents of the 3D scene model has not the objective of being exhaustive and very detailed. It has to contain all the needed information to help the interpretation of video sequences. As this information is filled by a human

name =	ticket vending machine 1
type =	equipment
function =	ticket vending machine
characteritics =	fragile
proximity =	100 cm
normal time =	30 s
polygon =	([0, 415, 0],[0, 520, 0,],
	[-50, 520, 0], [-50, 415, 0])
height =	180 cm

Fig. 42. Example of description of a ticket vending machine

operator for each camera its contents must be limited to useful information. Although the proposed geometric model and the semantic information can be enriched in the future.

Interpretation In this section we adress the problem of high-level description of mobile object behavior using generic observable events and application-dependent scenarios. The recognition process of temporal concepts can be reduced to the recognition of atemporal ones: object states. An event is a spatio-temporal property which represents a significative change in the state of objects in the scene. Typical events usally are "to enter", "to start running", "to stand up" or "to leave". The algorithm for event recognition is the following : an event is recognized if for a given object state, the value of this state is significantly different between an image (corresponding to time t_0) and another image (corresponding to time t_n, with $t_n = t_0 + d_{rec}$). The time interval between I_0 and I_n is called recognition delay d_{rec}.

For instance, if , at time t_0, a person is far from a coffee machine, and close to that coffee machine at time t_n, then the event "the person moves close to the coffee machine" is recognized.

The problem of event recognition can be reduced to the atemporal problem of finding a set of states describing the scene with enough accuracy. In other words, solving the problem of event recognition is reduced to solving the problem of symbolic description of the scene. So, if for each image we have a symbolic description of the scene, it is sufficient to compare these descriptions to know the changes which had happened, i.e. the events which have occured. This scene description must include a translation from numerical to symbolic values and must be generic enough to be applied to different environments and different applicative domains.

State model

The objective is to define state models which can be extended and parametrized. A state of objects in the scene is defined by an n-ary tree which represents the way this state is computed (see Fig. 43 an abstract example of such a tree). Four types of nodes are distinguished: object nodes, descriptor

nodes, operator nodes and classifier nodes (see below for their definition). The leaves of this tree are the objects involved in that state. Father nodes of the leaves are numerical descriptors of these objects. All intermediate nodes are operator nodes. The root node is always a classifier node which computes the symbolic value of the object state. The minimal tree structure is reduced to 3 nodes, 1 object leaf node 1 descriptor intermediate node and 1 classifier root node. The number of branches of the tree and the length of the branches are free.

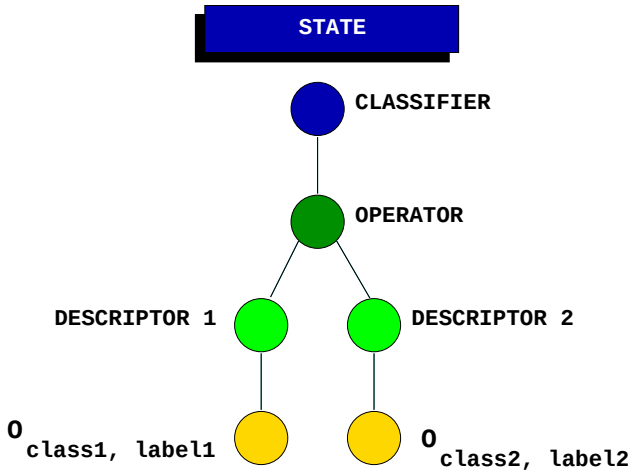


Fig. 43. Example of state modeling. Objects are in light grey, descriptors are in grey, the operator is in dark grey, and the classifier in black.

Objects. Objects are the objects of the scene at time t, i.e. an element of O, the set of the objects $o_{i,j}$ where i is the class of the object and j its label. For instance the object $o_{person,1}$ is a mobile object which has been recognized as being a person and whose label is 1. $o_{equipment,door2}$ is an object belonging to the class equipment labeled as door2.

Descriptors. Descriptors are functions defined from O to R^p to access an object measure. For instance, the size, the position, the shape, the trajectory, the orientation or the volume are possible descriptors. This notion ensures the anchoring of the model in the numerical results of the perception component.

Operators.Operators are functions defined from $(R^{p_1} \times \ldots \times R^{p_n})$ to R^q in order to operate on the measures. Examples of operators are the distance, the norm, the classical arithmetic or logic ones.

Classifiers. Classifiers are functions defined from R^p to S, the set of authorized symbols of the state. For instance *close* and far can be possible symbols for a state. These classifiers ensure the transformation from numbers

to symbols by defining a numerical domain of definition for each symbolic value.

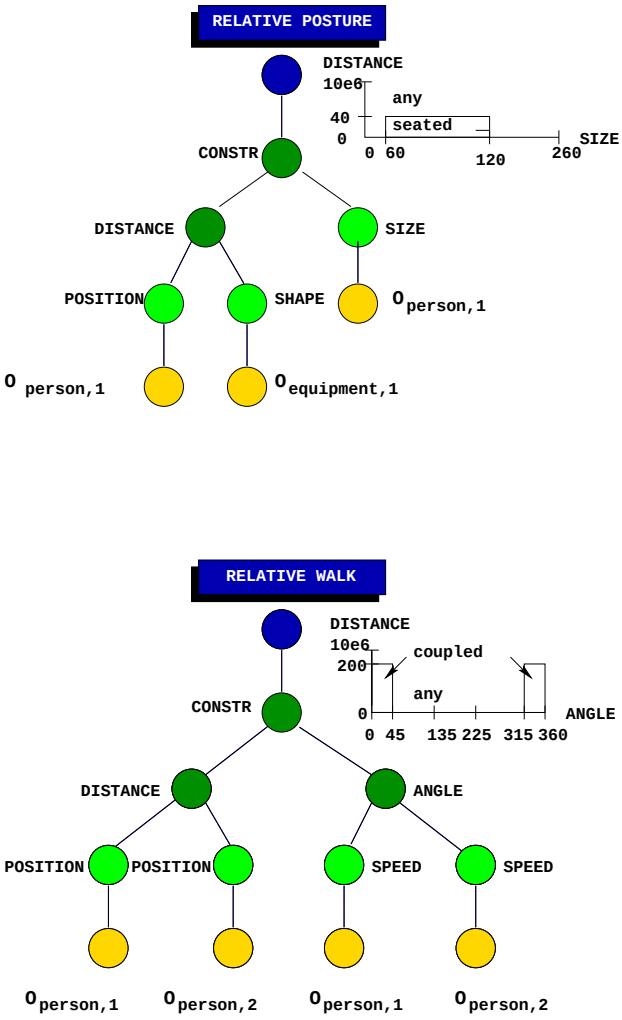


Fig. 44. Two instances of the model of state. Objects are in light grey, descriptors are in grey, the operator is in dark grey, and the classifier in black.

We have used this model of state to define a first set of states (see two examples on figure 44). For that we have defined three classes of objects, four descriptors, four operators and eight classifiers.

The 3 classes of objects are *person*, *area*, and *equipment*. Persons are the mobile objects of the scene which have been detected by the perception

component. Persons are described by a vector (px_{3D}, py_{3D}) representing the location of the person on the ground, a vector (vx_{3D}, vy_{3D}) representing the speed vector of the person and the size h_{3D} of that person. Areas and equipments are those which are defined in the 3D scene model (see Section 4). An area is a static object representing a subpart of the ground of the scene with a polygon. An equipment represents any volumic object of the environment for which we know the polygonal basis and the height h.

The 4 descriptors are: *position*, *size*, *speed* and *shape*. More precisely: $position(o_{i,j})$ applied to a person gives access to (px_{3D}, py_{3D}), $size(o_{i,j})$ applied to a person or to an equipment enables us to recover its size, $speed(o_{i,j})$ applied to a person returns the speed vector (vx_{3D}, vy_{3D}) and $shape(o_{i,j})$ applied to an equipment or an area returns its associated polygon.

The 4 operators are: *distance* the euclidean distance, *norm* the norm of a vector, *angle* the angle between two vectors in degrees and *constr* an operator which constructs a $2D$ vector with its scalar components.

We have defined 8 classifiers which compute 8 states:
posture, *direction*, *velocity*, *location*, *proximity*, *relative location*, *relative posture* and *relative walk*. For instance we have defined (see figure 44) the state *relative walk*($o_{person,i}$, $o_{person,j}$) by measuring the angle between the speed vectors of $o_{person,i}$ and $o_{person,j}$ and the distance between these two persons. If the speed vectors have a similar orientation (an angle below 45 degrees or greater than 315 degrees) and if the distance is small (below 200cm) then these persons are considered as having a *coupled* relative walk.

Event recognition

Event recognition is now straight forward: for each image frame the object states are computed with the current objects detected in the scene at that time. If for a detected object and for a state model, there is a change in its symbolic value a new event is created at that time.

The 8 predefined state models enable us to define 18 types of event.
Posture changes create the events $o_{person,i}$ **falls down** or **crouches down** or **stands up**.
Direction changes create the events $o_{person,i}$ **goes right side** or **goes left side**, or **goes away** or **arrives**. Velocity changes create the events $o_{person,i}$ **stops** or **walks** or **starts runing**.
Location changes create the events $o_{person,i}$ **leaves** or **enters** $o_{area,j}$.
Proximity changes create the events $o_{person,i}$ **moves close to** or **moves away from** $o_{equipment,j}$.
Relative location changes create the events $o_{person,i}$ **moves close to** or **moves away from** $o_{person,j}$.
Relative posture changes create the event $o_{person,i}$ **sits on** $o_{equipment,j}$.
Relative walk changes create the event $o_{person,i}$ and $o_{person,j}$ **walk together**.

Scenario recognition The final problem is to incrementally recognize predefined scenarios representing behaviors. A scenario is an interdependent set of events. To recognize a scenario implies to recognize all the events which

compose it and to verify their dependencies. The constraints can be temporal, spatial, logical or algebraic.

We will now give details of the scenario model we use. A scenario $s_{i,t}$, where i is the scenario identifier and t the current time of recognition, is composed of four parts: Events, Constraints, Conditions, and Success. Examples of scenario are shown in next section (see Fig. 45, 46, 50 and 51).

Events. They are the events $\{e_1, ..., e_i, ..., e_n\}$ requested by the scenario. Each event e_i is associated with the variable t_i which represents the time when e_i occured. There are two categories of events in this part: positive events and negative events. Positive events must occur for the total recognition and negative events must not occur during scenario recognition.

Constraints. They are temporal constraints $\{c_1, ..., c_i, ..., c_m\}$. Those constraints are described as first degree linear inequations on $t_1, ..., t_i, ..., t_n$.

Conditions. They are non-temporal constraints $\{k_1, ..., k_i, ..., k_p\}$ on the objects involved in the events. It forces an event object attribute to a predefined value. This attribute can be symbolic (name, function, etc...) or numerical (height, size, velocity, etc...).

Success. They are keywords $\{f_1, ..., f_i, ..., f_q\}$, which indicate the kind of feedback associated with the scenario. This part is used when the scenario is totally instantiated. There are two kinds of feedback: external and internal. External feedback is used to trigger an alarm to the security operators and internal feedback is used to generate an event to signify that the scenario has been totally recognized.

A scenario can be totally recognized, when all the events are recognized and all the constraints are verified; it can be partially recognized, when a subset S of all events are recognized and the constraints involving events of S are verified; when no event of a scenario are recognized, this scenario is called a blank scenario. The principle of the scenario recognition algorithm [17] consists of two points: as previously described, we generate, image after image, interesting events which happened in the scene, then with those events we instantiate in parallee predefined scenario models. It means that scenario recognition corresponds to updating a set of partially recognized scenarios. This scenario recognition method is an extension of the work on chronicles explained in [24].

In short, given a set of scenarios $\{s_{1,t-1}, ..., s_{i,t-1}, s_{i+1,0}, .., s_{k,0}\}$ composed of partially recognized at $t-1$ scenarios and blank scenarios and a set of events $\{e_{1,t}, ..., e_{n,t}\}$ recognized at t, the principle of scenario recognition is based on two points.

For each $s_j \in \{s_{1,t-1}, ..., s_{i,t-1}, s_{i+1,0}, .., s_{k,0}\}$, for each event of s_j if the event matches an event $e_{1,t}, ..., e_{n,t}$ and verifies the temporal constraints $c_1, ..., c_i, ..., c_m$ and the non-temporal constraints $k_1, ..., k_i, ..., k_p$, we create $s_{j,t}$. It results a new set $\{s_{1,t}, ..., s_{l,t}\}$ of scenarios. In this context, an event of s_j matches $e_{1,t}$ means that $e_{1,t}$ is an instance of this event.

We remove invalid scenarios $s_{i,t}$ from $\{s_{1,t}, ..., s_{l,t}\}$, if:
-one negative event e_k of $s_{i,t}$ has been instantiated,
-one of the $c_1, ..., c_i, ..., c_m$ implies that $s_{i,t}$ will not be instantiated.

Results of metro station applications In this part, we will describe the results obtained on real visual surveillance applications for metro stations. The videos come from the CCTV networks of the metro operators partners of the AVS-PV european project. We have formalized the expertise of three security engineers in a knowledge base containing currently 15 scenarios.

Metro Station in Brussels. In the following, we detail how two scenarios described in figures 45 and 46 are recognized. These scenarios belong to the knowledge base built for AVS-PV european project and videos have been recorded in a STIB Metro Station in Brussels. The camera observes the platform of a metro station. The aims of those two scenarios are: to prevent vandalism against equipment and to ensure the safety of passengers.

Scenario
Name = "forbidden access to area",
Events = $(t_1,\ enters(p_1 : Person,\ a_1 : Area))$,
not$(t_2,\ leaves(p_1 : Person,\ a_1 : Area))$,
Constraints = $t_1 \leq t_2,\ t_2 \leq t_1 + 1.0$,
Conditions = function(a_1, "forbidden_access"),
Success = alarm(p_1, "has entered area",a_1)

Fig. 45. AVS-PV scenario model:"forbidden access to area"

Scenario
Name = "graffiti on wall",
Events =
$(t_1, moves\ close\ to\ (p_1 : Person,\ e_1 : Equipment))$,
not$(t_2,\ moves\ away\ from\ (p_1 : Person,\ e_1 : Equipment))$,
Constraints = $t_1 \leq t_2$,
$t_2 \leq t_1$ + normal_presence_time(e_1),
Conditions = function(e_1, "wall"),
Success = alarm(p_1, "doing graffiti onto",e_1)

Fig. 46. AVS-PV scenario model: "graffiti on wall"

At t_1, a detected person (named Person 1 in the folllowing) goes *inside* the tracks area. The event "person 1 *enters* tracks area is triggered. This area is labelled as a forbbiden area, so the first event of "forbidden access to area" scenario is recognized. Several frames later, at t_2 (see Fig. 47), Person 1 is still *inside* the "tracks" area, so the event "Person 1 *exits* the "tracks" area"

has not been triggered. The non-occurence of the event matches the second event (negative event) of the scenario "forbidden access to area". An alarm is sent to the security operator.

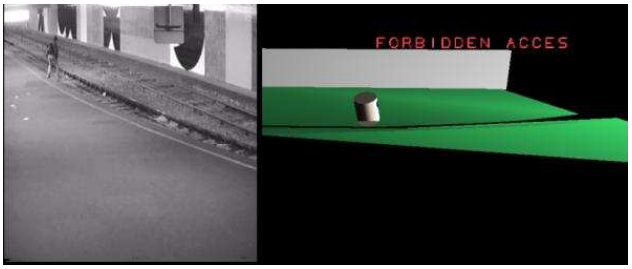


Fig. 47. Left: Metro platform in Brussels at t_2. Right: 3D position of the detected person (represented by a cylinder) w.r.t the 3D scene model. An alarm 'forbidden access to area" is sent to the security operator.

Further at t_3 (see Fig. 48), Person 1 is *close to* the equipment "wall", so the event "person 1 *moves close to* equipment wall" is triggered. This event instantiates the first event of the scenario "graffiti on wall".

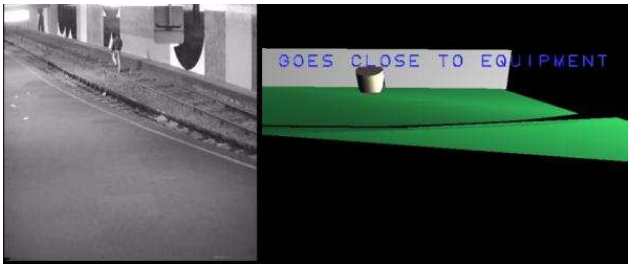


Fig. 48. Left: Metro platform in Brussels at t_3. Right: an event "person 1 *moves close to* equipment wall" is detected.

Further at t_4 (see Fig.49), Person 1 is still *close to* the equipment "wall", so the event "person 1 *moves off* equipment wall" has not been triggered. The non-occurence of the event matches the second event (negative event) of the scenario "graffiti on wall". An alarm is sent to the security operator.

Metro Station in Nuremberg. In the following, we detail how two other scenarios described in figures 50 and 51 are recognized. These scenarios also belong to the knowledge base built for AVS-PV european project and videos have been taken in a VAG Metro Station in Nuremberg (Germany). In this example, the camera observes the entrance of a metro station. The aim of

Fig. 49. Left: Metro platform in Brussels at t_4. Right: an alarm 'graffiti on wall" is sent to the security operator.

those two scenarios is to prevent vandalism against ticket vending machines. These machines have been defined in the 3D scene model (see Section 42) as fragile equipment.

Scenario
Name = "Presence period near fragile equipment",
Events =
$(t_1, moves\ close\ to\ (p_1 : Person,\ e_1 : Equipment))$,
not$(t_2,\ moves\ away\ from\ (p_1 : Person,\ e_1 : Equipment))$,
$(t_3,\ stops(p_1 : Person))$,
Constraints =
$t_1 \leq t_2,\ t_1 \leq t_3$,
$t_2 \leq t_1 +$ normal_presence_time(e_1),
Conditions = function$(e_1,$ "fragile"),
Success =
alarm("Presence period near equipment",e_1),
loopback(t_2, presence_period_near_fragile, e_1, p_1)

Fig. 50. AVS-PV scenario model: "Presence period near fragile equipment"

At t_1, a detected person (named Person 1 in the folllowing) is far from an equipment labeled as "fragile". Further at t_2 (see Fig. 52), person 1 is *close* to equipment labeled as "fragile", so the event "person 1 *moves close to* an equipment" is triggered. The first event of scenario "Period near fragile equipment" is instanciated.

Further at t_3 Person 1 is still *close* to the machine, so the negative event of scenario "Period near fragile equipment" is instanciated. Secondly, the fact that Person 1 stops triggers the event "Person 1 *stops*". The three events of the scenario "Period near fragile equipment" are recognized and an alarm is sent to the security operator. The complete recognition of this scenario triggers the specific loopback event: $presence\_near\_fragile$" equipment. This specific event matches the first event of the scenario "Repeated Presence period near

Scenario

Name = "Repeated period near fragile equipment",
Events =
(t_1, presence_period_near_fragile, e_1, p_1),
(t_2, $moves\ close\ to$ ($p_1 : Person$, $e_1 : Equipment$)),
not(t_3, $moves\ away\ from$ ($p_1 : Person$, $e_1 : Equipment$)),
(t_4, $stop(p_1 : Person)$)
Constraints =
$t_1 \leq t_2$, $t_2 \leq t_3$, $t_2 \leq t_4$,
$t_3 \leq t_2 +$ normal_presence_time(e_1),
Conditions = function(e_1, "fragile"),
Success = alarm("Vandalism on ",e_1)

Fig. 51. AVS-PV scenario model: "Repeated Presence period near fragile equipment"

Fig. 52. Left: Metro entrance in Nuremberg at t_2. Right: an event "person 1 *moves close to* an equipment" is detected.

fragile equipment". Then at t_4 (see Fig. 53) an other event is detected because the Person 1 moves away in the direction of the corridor to check is anybody is arriving.

At t_5 (see Fig. 54), the event "Person 1 *moves close to* equipment" is triggered. This equipment is the same equipment that the one at t_2. The scenario "Repeated Presence period near fragile equipment" is now totally recognized. An alarm is sent to the security operator.

The results of these applications were considered very satifactory by the metro operators. The richness of the formalism for scenario description allows us to specify a set of constraints (temporal as well as atemporal) which reduce false alarms. The formalism we have proposed for scenario description has enabled us to represent the expertise for these applications. The knowledge modeling is still difficult. The main reason is that we need to manage the gap between vague security concepts (such as "abnormal behavior") to rigorous scenario models.

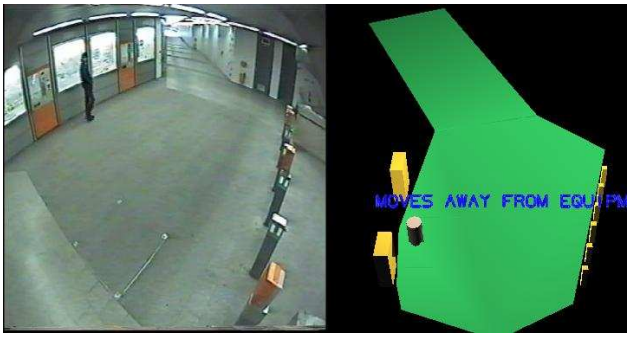


Fig. 53. Left: Metro entrance in Nuremberg at t_4. Right: an event 'person 1 *moves away from* an equipment" is detected

Fig. 54. Left: Metro entrance in Nuremberg at t_5. Right: a "Vandalism' 'alarm is sent to the security operator.

Conclusion In this section we have shown that high-level video understanding can be performed based on images taken from a single static camera and with simple perception methods working almost in real-time. This has been possible by using two sets of a priori information: first, 3D scene model describing the $3D$ geometry of the observed scene and semantic information on the static objects and interesting areas, second, general knowledge of predefined scenarios valid for an application domain. We have proposed a formalism to represent these two types of a priori information and explained how to use them for video understanding. We have also proposed a formalism for event recognition based on object state models. This formalism is independent of a particular application domain and includes a transformation from the perception data to the scenario models. The current video understanding framework we propose has several limitations. One type of problem is the imprecision and uncertainty in the detection and location of mobile objects; most of these low-level detection errors are due either to reflections, shadows

or occlusions. A solution to cope with these problems is to relax our second hypothesis and not to restrict ourselves to the use of a single camera. Another more general problem is that as every vision system, this framework needs, for each perception method and for each interpretation method, to set the values of numerical parameters in a configuration phase. One solution to solve this problem is to use learning techniques to find the best parameter values for an application if they exist.

5 Conclusion

In this chapter we have seen how knowledge-base techniques can help solving complex image processing problems. First we have addressed the problem of using an image processing library and proposed program supervision techniques to encapsulate the knowledge of planning and repair of programs. Then we have shown how these techniques have been applied in astronomy for the automatic description of images containing a galaxy. Second we have addressed the problem of image understanding. Two different subproblems have been considered: the automatic recognition of a complex natural object and the recognition of scenario in video sequences. The recognition of complex natural object has been modeled as a classification problem with the hypothesis that a hierarchy of predefined classes are available. This method has been described through an example in micropaleontology for foraminefera identification. Two kinds of a priori knowledge have been modeled: 3D scene model describing the static environment and predefined scenarios describing possible interesting events. The recognition of scenario in video sequences has been detailed in the context of visualsurveillance applications.

Acknowledgements

The author would like to thank the members of the ORION team who have been involved in this research work, in particular Sabine Moisan and Véronique Clément on program supervision, Shan Liu-Yu on object recognition , François Brémond, Nicolas Chleq and Nathanael Rota on scenario recognition.

References

1. E. André, G. Herzog, and T. Rist. On the simultaneous interpretation of real world image sequences and their natural language description: The system soccer. In *8th European Conference of Artificial Intelligence*, pages 449 – 454, Munich, 1988.
2. D. Ayers and M. Shah. Monitoring human behavior in an office environment. In *Computer Society Workshop on Interpretation of Visual Motion*, 1998.

3. D. G. Bailey. Research on Computer-assisted Generation of Image Processing Algorithms. In *Workshop on Computer Vision - Special Hardware and Industria l Applications*, Tokyo, October 1988. IAPR.
4. A. Barr, P.R. Cohen and E.A. Feigenbaum editors. *The Hanbook of Artificial Intelligence*, volume 4 Addison-Wesley, Reading, Massachusetts, 1989.
5. A. Barr and E.A. Feigenbaum editors. *The Hanbook of Artificial Intelligence*, volume 1 HeurisTech Press and William Kaufman, Standford., 1981.
6. A. Barr and E.A. Feigenbaum editors. *The Hanbook of Artificial Intelligence*, volume 2 HeurisTech Press and William Kaufman, Standford., 1982.
7. A. Bellon, J-P. Dérutin, F. Heitz, and Y. Ricquebourg. Real-Time Collision Avoidance at Road Crossing On Board the Prometheus Prolab2 Vehicle. In *Intelligent Vehicles'94 Symposium*, pages 56–61, Paris, (France), October 1994.
8. S. Beucher and M. Bilodeau. Road Segmentation and Obstacle Dectection by a Fast Watershed Transformation. In *Intelligent Vehicles'94 Symposium*, pages 296–301, Paris, (France), October 1994.
9. A. Boersma, "Introduction to marine micropaleontology", in *Foraminifera*, B. U. Haq and A. Boersma, ed., Elsevier, New-York, pp. 19–78, 1978.
10. British-Aerospace. VIDIMUS Esprit Project Annual Report. Technical report, Sowerby Research Centre, Bristol, England, 1991.
11. D. R. Brough, I. F. Alexander, "The fossil expert system", *Expert Systems*, Vo. 3, No. 2, pp.76–83, 1986.
12. H. Buxton and S. Gong. Visual surveillance in dynamic and uncertain world. *Artificial Intelligence Journal*, 78:431 – 459, 1995.
13. M. Caron, *Cretaceous planktonic foraminifera*, Cambridge University Press, Eds. H. M. Bolli et al., pp. 17–85, 1985.
14. C. Castel, L. Chaudron, and C. Tessier. What is going on ? a high level interpretation of sequences of images. In *4th European Conference on Computer Vision, Workshop on Conceptual Descriptions from Images*, Cambridge UK, April 1996.
15. S. A. Chien and H. B. Mortensen. Automating image processing for scientific data analysis of a large image database. *IEEE Transactions on Pattern Analysis and Machine Intelligence* , 18(8):854–859, August 1996.
16. N. Chleq, F. Bremond, and M. Thonnat. *Advanced Video-Based Surveillance Systems*, chapter Image Understanding for Prevention of Vandalism in Metro Station, pages 106 –117. Kluwer Academic Publishers, 1999.
17. N. Chleq and M. Thonnat. Realtime image sequence interpretation for video-surveillance. In IEEE, editor, *International Conference on Image Processing*, pages 801 – 804, Lausanne, Switzerland, 1996.
18. V. Clément and M. Thonnat. Integration of image processing procedures: OCAPI, a knowledge based approach. *Computer Vision Graphics and Image Processing: Image Understanding*, 57(2), 1993.
19. P.R. Cohen and E.A. Feigenbaum editors. *The Hanbook of Aritificial Intelligence*, volume 3 HeurisTech Press and William Kaufman, Standford., 1982
20. I. Cohen and G. Medioni. Detecting and tracking moving objects for video surveillance. In *Computer Vision and Pattern Recognition*, Fort Collins, Colorado, June 1999.
21. M. A. Conrad, D. S. Beightol, "Expert systems identify fossils and manage large paleontological databases", *Geobyte*, Vol. 3, No. 1, pp. 42-46, 1988.

22. M. Crubézy, F. Aubry, S. Moisan, V. Chameroy, M. Thonnat, and R. Di Paola. Managing Complex Processing of Medical Image Sequences by Program Supervision Techniques. In *SPIE Medical Imaging'97*, volume 3035, pages 614–625, February 1997.
23. S. Dettmer, A. Seetharamaiah, L. Wang, and M. Shah. Model-based approch for recognizing human activities from video sequences. In *Workshop on Motion of Non-Rigid and Articulated Objects*, June 1998.
24. C. Dousson, P. Gaborit and M. Ghallab. Situation recognition:representation and algorithms. In *Proc. 13th IJCAI*, Chambery, p.166-172, 1993.
25. B. Draper, A. Hanson, and E. Riseman. Knowledge-directed vision: Control, learning, and integration. *IEEE Signals and Symbols*, 84(11):1625–1637, 1996.
26. L. Gong and C. A. Kulikowski. Composition of Image Analysis Processes Through Object-Centered Hierarchical Planning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(10), October 1995.
27. C. Granger, *Reconnaissance d'objets par mise en correspondance en vision par ordinateur*, Ph. D. thesis, l'Université de Nice - Sophia Antipolis, France, 1985.
28. G. Herzog. Utilizing interval-based event representation for incrementatal high-level scene analysis. In *4th International Workshop on Semantics of Time, Space, and Movement and Spatio-Temporal Reasonning*, Chateau de Bonas, France, 1992.
29. S. Intille and A. Bobick. Closed world tracking. In *5th International Conference on Computer Vision*, Cambridge, 1995.
30. S. Intille and A. F. Bobick. Visual recognition of multi-agent action using binary temporal relations. In *Computer Vision and Pattern Recognition*, Fort Collins, Colorado, June 1999.
31. Y. Ivanov, C. Stauffer, A. Bobick, and W. Grimson. Video surveillance of interactions. In *2nd International Workshop on Visual Surveillance*, pages 82 – 89, Fort Collins, Colorado, June 1999.
32. M.D. Johnston. An Expert System Approach to Astronomical Data Analysis. In *Proceedings, Goddard Conf. on Space Applications of Artificial Intelligence and Robotics*, pages 1–17, 1987.
33. S. Liu, M. Thonnat and M. Berthod. Automatic Classification of Planktonic Foraminifera by a Kno wledge-based System. In *Proceedings of The Tenth Conference on Artificial Intelligence for Applications, CAIA*,IEEE Computer Society Press San Antonio, Texas, pp358-364, March, 1994.
34. S. Liu-Yu, *Reconnaissance de formes par vision par ordinateur: application à l'identification des foraminifères planctoniques*, Ph. D. thesis, l'Université de Nice - Sophia Antipolis, France, June 1992.
35. T. Matsuyama. Expert systems for image processing: Knowledge based composition of image analysis processes. *Computer Vision Graphics Image Processes*, 48:22–49, 1989.
36. G. Medioni, I. Cohen, F. Brémond, S. Hongeng, and R. Nevatia. Event detection and analysis from video streams. In *DARPA Image Understanding Workshop*, Monterey, November 1998.
37. A. Meygret, M. Thonnat, and M. Berthod. A pyramidal stereovision algorithm based on cont our chain points. In *European Conference on Computer Vision*, Antibes, April 1990 .
38. S. Moisan, C. Shekhar, and M. Thonnat. Real-Time Perception Program supervision for Vehicle Driving Assistance. In Okyay Kaynak, Mehmed Ozkan, Nurdan Bekiroglu, and Ilker Tunay, editors, *ICRAM'95 International Conference*

on recent Advances in Mechatronics, pages 173–179, Istanbul, Turkey, August 1995.

39. S.H.Nawab and V.Lesser. Integrated processing and understanding of signals. In A.V.Oppenheim and S.H.Nawab, editors, *Symbolic and Knowledge-based Signal Processing*, pages 251–285. Prentice Hall, 1992.
40. J.C. Ossola and M. Thonnat. Cooperation of knowledge based systems for galaxy classification. In *8th International Conference on Image Analysis and Processing (ICIAP'95)*, September 1995.
41. S. Pouget, M. Thonnat, V. Clement,A. de Fombelle, "Automatic diagnosis of a technical device by classification, the expert system DANTE : an application to antenna failures", *6th International conference on reliability and maintenability*, Strasbourg, October 1988.
42. Groupe PROART. Rapport de fin de contrat de recherche PROMETHEUS-PROART. Technical report, Projet PROMETHEUS, December 1994.
43. W. R. Riedel, "IDENTIFY: A Prolog program to help identify fossils", *Computer & Geosciences*, Vol. 15, No. 5, pp. 809–823, 1989.
44. F. Robaszynski, M. Caron, J. M. Gonzalez Donoso, A. H. Wonders, "Atlas of Late Cretaceous Globotruncanids", *Revue de Micropaléontologie*, Vol. 26, No. 3–4, Maison de la Géologie, Paris, 1984.
45. N. Rota, R. Stahr and M. Thonnat. .Tracking for Visual Surveillance in VSIS. In *PETS2000*, Grenoble, March 2000.
46. S. Russel and P. Norvig. Artificial Intelligence: a modern approach, Prentice Hall International Editions, 1995
47. H. Sato, Y. Kitamura, and H. Tamura. A Knowledge-based Approach to Vision Algorithm Design for Industria l Parts Feeder. In *Proceedings, IAPR Workshop on Computer Vision, Special hardware and industrial applications*, pages 413–416, Tokyo, 1988.
48. P. A. Swaby, "Integrating Artificial Intelligence and Graphics in a Tool for Microfossil Identification for Use in the Petroleum Industry", *Second Annual Conference on Innovative Applications of Artificial Intelligence*, pp. 203 – 218, Washington, 1st – 3rd May, 1990.
49. C. Tessier. Reconnaissance de scènes dynamiques à partir de données issues de capteurs: le projet perception. Technical report, Onera-Cert, 2 avenue Edouard-Belin BP 4025 31055 Toulouse Cedex France, Août 1997.
50. M. Thonnat. *The world of galaxies*, chapter Toward an automatic classification of galaxies, pages 53–74. Springer-Verlag, 1989.
51. M. Thonnat, V. Clément, and J. C. Ossola. Automatic galaxy description. *Astrophysical Letters and Communication*, 31(1-6):65–72, 1995.
52. M. Thonnat, V. Clement, and J. van den Elst. Supervision of perception tasks for autonomous systems: the OCAPI approach. *Journal of Information Science and Technology*, 3(2):140–163, Jan 1994. Also in Rapport de Recherche 2000, 1993, INRIA Sophia Antipolis.
53. M. Thonnat and M-H. Gandelin, "An expert system for the automatic classification and description of zooplanktons from monocular images", ICPR9, pp. 114–118, Roma, Italy, November 1988.
54. M. Thonnat, C. Granger, M. Berthod, "Design of an expert system for object classification through an application to the classification of galaxies", *Proc. of CVPR'85*, pp. 206–208, 1985.

55. M. Thonnat and S. Moisan. Knowledge-based systems for program supervision. In *First international workshop on Knowledge-Based systems for the (re)Use of Programs libraries KBUP'95*, Sophia Antipolis, France, 1995. INRIA.
56. M. Thonnat and S. Moisan. Knowledge-Based Systems for Program Supervision. In *Proceedings of KBUP'95*, pages 3–8, Sophia Antipolis, France, November 1995. INRIA.
57. M. Thonnat and N. Rota. Image understanding for visual surveillance applications. In *Third International Workshop on Cooperative Distributed Vision*, Kyoto, Japan, November 1999.
58. T. Toriu, H. Iwase, and M Yoshida. An Expert System for Image Processing. *FUJITSU Sci.Tech.Journal*, 23.2:111–118, 1987.
59. L. Trassoudaine, J. Alizon, F. Collange, and J. Gallice. Visual Tracking by a Multisensorial Approach. In *First IFAC International Workshop on Intelligent Autonomous Vehicles*, pages 111–116, Southampton, (UK), April 1993.
60. G. de Vaucouleurs. Classification and morphology of external galaxies. *Handbuch der Physik*, 53:275, March 1957.
61. R. Vincent, S. Moisan and M. Thonnat. Une bibliothèque pour des moteurs de pilotage de programmes. Research Report 3011, I.N.R.I.A., 1996. http://www.inria.fr/orion/.
62. R. Vincent and M. Thonnat. Planning, Executing, Controlling and Replanning for IP program library. In *8th Artificial intelligence and Soft computing (ASC'97)*, Banff, July 1997.
63. R. Vincent, M. Thonnat, and J.C. Ossola. Program supervision for automatic galaxy classification. In *Proc. of the International Conference on Imaging Science, Systems, and Technology CISST'97*, Las Vegas, USA, June 1997.
64. , S. Yu, P. Saint-Marc, M. Thonnat and M. Berthod. Feasibility study of automatic identification of planktic foraminifera by c omputer vision. In *Journal of Foramineferal Research*, 26, (2),April 1996.