PERGAMON

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Pattern Recognition 37 (2004) 631–645

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

An immune oriented multi-agent system for biological image processing

V. Rodin*, A. Benzinou, A. Guillaud, P. Ballet, F. Harrouet, J. Tisseau, J. Le Bihan

Technopôle Brest-Iroise, École Nationale d'Ingénieurs de Brest, CS 73862, 29238 Brest Cedex 3, France

Received 16 September 2003; accepted 23 September 2003

Abstract

In this article, we present a parallel image processing system based on the concept of reactive agents. Our system lies in the oRis language, which allows to describe finely and simply the agents' behaviors to detect image features. We also present a method of segmentation using a multi-agent system, and two biological applications made with oRis. The stopping of this multi-agent system is implemented through a technique issued from immunology: the apoptosis.

Keywords: Artificial immune systems; Multi-agent language; Multi-agent system; Parallel image processing; Spots detection; Striae detection

1. Introduction

In this article, we present a parallel image processing system based on a multi-agent language. Our system is a multi-agent system made up of reactive agents. An agent can move and has its own position in its environment, that is to say the image. An agent actually picks up a small part of the environment and also requires a shared memory. The notion of shared memory is very important because it allows the interaction among agents and the coordination of their actions. The shared memory contains the already detected features (edges, region, etc.). In order to model our multi-agent system for the detection of image features we use the formalism Basic Representation of Interactive Components (BRIC) [1]. Fig. 1a presents our system which is made up of three component parts: the agents, the scheduler and the environment. Sensors allow each agent to receive outer information (see Fig. 1b). According to the data given by the sensors, its internal memory and its internal state, the agent makes its decisions and applies them through its actuators (Go to and, eventually, write a feature in the shared memory). These internal states are usually represented by a finite state machine. This means that, in our multi-agent system, each agent has a very simple behavior which allows it to take a decision (find out an edge, a region, etc.) according to its position in the image and to the information enclosed in it. The behaviors of these agents are very simply described in oRis [2], an agent-oriented object language which is in the center of our parallel image processing system.

In this paper, we also present a method of segmentation using a multi-agent system, and two biological applications made with oRis. The first application allows the detection of concentric striae of biological "objects" (age-rings of tree, fish otolith growth rings, etc.). In this application, we use a set of agents which follow either the light rings or the dark rings and act on the image. Their actions aim at the reinforcement of the rings by stressing the contrasts thus allowing a reliable detection of these rings, even if they are discontinuous (see Fig. 2a). The second application concerns the 2D gel electrophoresis domain which is used for the isolation of proteins for further characterization by mass spectroscopy (see Fig. 2b). In this application, we developed agents able to detect spots in the images even if the spots are saturated or very low contrasted. These agents are very similar to the agents designed for the first application. In these two applications, we try to get informations which

* Corresponding author. Tel.: +33-298056626; fax: +33-298056629.
E-mail address: rodin@enib.fr (V. Rodin).

0031-3203/$30.00 © 2003 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2003.09.014

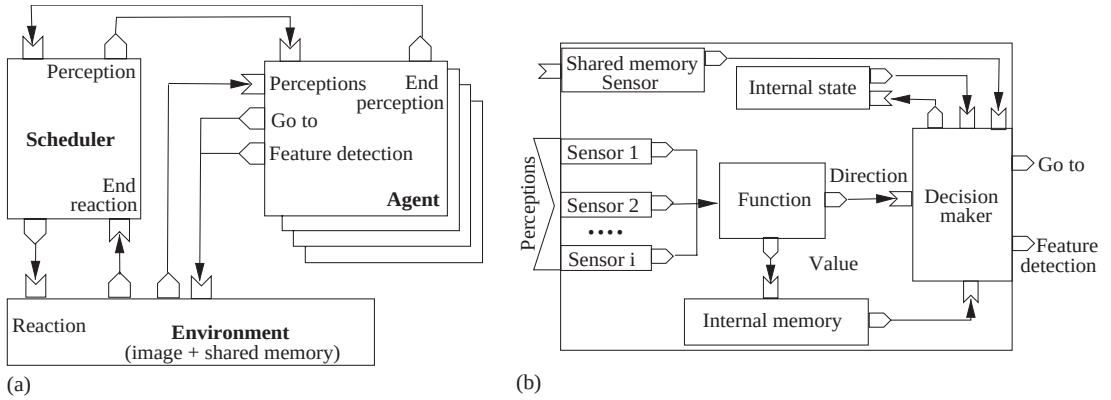


Fig. 1. BRIC models: (a) image processing multi-agent system, (b) agents.

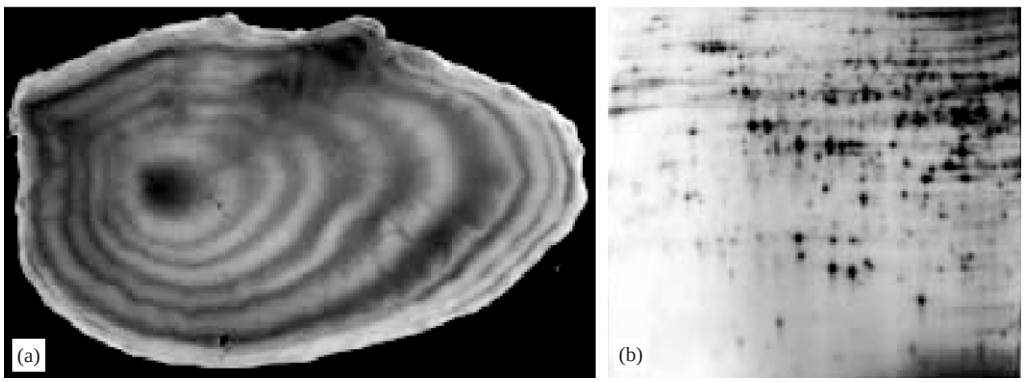


Fig. 2. (a) A 8 year old plaice otolith; (b) A 2D gel electrophoresis (root of pine).

can be assimilated with roof edges in images. We present an approach to detect this type of edges using a multi-agent system.

Edges are generally considered as discontinuities in the gray levels of an image, and are called step edges. Nevertheless one can find other types of edges in images, like roof edges or ridge edges. Canny [3] has defined a way to evaluate a good edge detector by using some criteria: the detector must give a good localization of the edge and a low number of bad detected points. Ziou [4] has adapted the optimal filters defined by Canny [3] for roof and ridge detection and extended them to the 2D case. He applied it on synthetic images featuring lines in an image containing gaussian noise. The results show an important over-detection of the edges. Haralick [5] has proposed to identify a ridge or a valley using all second partial derivative of a cubic polynomial centered at the origin of a mask. The zero crossing of the first derivative is searched in the direction which extremizes the second directional derivative of the polynomial. The method needs an important amount of computations and thresholds and the coefficients of the polynomial have to be chosen. Smith [6] proposes a method to detect skeletons on line images which could be used to detect ridges, with an application to handwritten characters recognition. The image has to be first binarized, which can be easy for this type of application. Nevertheless this step is risky in several cases. For example, in the case of otoliths, a low threshold would erase the contrast between the last rings that are light, and a high threshold would erase the first rings that are darker. Another type of methods has recently been set for detecting features in images, which are based on multi-agent systems [7,8]. The agents used achieve quite simple actions, but by sharing their results, their work can result in a more complex treatment. Liu [8] used a multi-agent system to detect homogeneous regions and applied it to brain scan images. Each agent can achieve tests on pixels around it in a circular neighborhood, as computing the variance or the mean gray level. If it finds that its neighborhood satisfies the conditions to be a region, the central pixel will be marked and new agents will be generated in this neighborhood. This method is well adapted for brain scan images, because regions characteristics are regular for tumors, sane parts, etc.

In this paper we first describe oRis our multi-agent environment (Section 2). Then, we present two image processing applications developed with oRis (Section 3). We also analyze the influence of our agents' parameters (Section 4). Finally, we detail the stopping of our image processing multi-agent system which is implemented through an immunologic regulation concept (Section 5).

2. oRis language and simulator

The oRis environment not only proposes an agent-oriented language which allows to describe the structure and functioning of any multi-agent system, but also a simulator to dynamically simulate and modify the modeled system [2].

2.1. oRis language

The oRis language allows the structured programming in a way very close to the C/C++ language. Thus, the writing of a program in structured programming consists in defining the functions and classes. The structured programming's main code is represented by a block of instructions named `'execute'`. This block is quite similar to the function `'main()'` of the C/C++ language since it represents the program's entry point. The oRis language adds to the object-oriented programming the notion of active object. It allows to access to the agent-oriented programming. The most important point which is brought is the instance autonomy. It means that the instances can be equipped with an autonomous behavior they perpetually execute as soon as they are created; these instances are qualified as agent. To write an agent-oriented application comes to instantiate agents. The block `'execute'` is only used to initialize the application by creating instances; then the instances, and not a main program, make the application evolve. Therefore, the end of the block `'execute'` does not represent the end of the application anymore. This is the great difference compared with the function `'main()'` of C/C++ languages. In oRis, the distinction object/agent is possible thanks to the instance's method `'main()'`. An instance equipped with this method is qualified as agent. Otherwise, it is an object, which means that we have to explicitly invoke the instance's methods to make it work. Actually, oRis is more than a language. It is also a simulator in charge of animating every agents. This is the reason why we usually talk about simulation for an oRis application. The simulator automatically and perpetually calls each agent's method `'main()'`. Effectively, this method represents the behavior's entry point of the concerned agent. Fig. 3 represents a simple agent-oriented program. This program contains the different points previously mentioned as well as the obtained display. It is to be

```
class Example              // Declaration of the class 'Example'
{
 int age;                  // An 'Example' contains an integer 'age'
 void main(void);          // An 'Example' has a method 'main()'
};                         // ---> it is an agent

void Example::main(void)    // Method 'main()' definition
{
 println("The age of ", this , " is ", ++age);
}

execute                     // Code to execute (initialization)
{
 println("---- begin ----");
 for(int i=0; i<4; i++) new Example; // Instantiate four 'Example's
 println("---- end ----");
}
```

The result of the agent–oriented program execution is as follows:

```
---- begin ----
---- end ----
The age of Example.2 is 1
The age of Example.3 is 1
The age of Example.4 is 1
The age of Example.1 is 1
The age of Example.1 is 2
The age of Example.3 is 2
The age of Example.2 is 2
The age of Example.4 is 2
...
```

Remark : agents are chosen in a random order.

Fig. 3. A simple agent-oriented program in oRis.

noticed that the block ‘execute’ only contains instantiations and that the execution of the application happens after the exit from this block and indefinitely continues.

Since the agents have their autonomy and their own existence, it is sometimes useful to specify the particularities of behavior on a particular method and not on the entire class. Thus, thanks to its granularity on the instance level, oRis allows to obtain an agent different from the other agents of its class. This allows an agent to redefine its behavior or other agents' behaviors if it thinks that this behavior is better for the execution of a precise task. This behavior redefinition could be done using the oRis dynamic properties. By dynamic properties, we consider the possibility to modify or complete the code of a simulation when this one is in process and this without interrupting its progress. All the language is available during the introduction of new codes. Thus, it is possible to redefine methods of existing classes, redefine methods of existing instances, define new classes, add methods to instances, add attributes to instances and execute any instruction, at any time and without stopping. The code, dynamically introduced may be composed by the agents themselves. The instance granularity really allows to consider agents which basis behavior changes with time. This concept of instance granularity will be very useful thereafter when the agents used to find information in images are able to differentiate themselves from each other (see Section 5). Fig. 4 represents two examples of redefinitions of methods on the instance level. This code implies that the simulation described Fig. 3 always works. It has to be noticed that the second redefinition is made by an agent which modifies its own ‘main()’.

The interest of a multi-agent architecture lies in the collaboration between agents. Therefore, these agents have several ways of communication: synchronous, asynchronous and broadcasting. The synchronous communication corresponds to simple calls of methods as in C++. This kind of communication is perfectly well adapted to passive objects or active objects with a small autonomy. On the other hand, a higher level communication requires that the objects (or agents) have the ability to exchange messages on an asynchronous way, with the management of a message box, in order to increase the flexibility and liability of the service demands, propositions, negotiations, etc. In oRis, the asynchronous communication consists in placing a message in an addressee's message box so as this latest treats the message whenever it wants and the way it wants. To develop advanced agents, it is also necessary to have a way of propagating and collecting information in the environment. The broadcasting mechanism seems to be well adapted to this kind of requirement. Effectively, it gives the transmitter the possibility to send information in the environment without caring about the concerned objects. Reciprocally, the agents can pay attention to the information without caring about the circumstances of the broadcasting. Thus, it is enough for the different agents to be able to broadcast and receive the same kind of information, to have the possibility to communicate and to co-evolve. This point is very interesting since it allows an agent to warn the other agents that, for example, a precise region of the image was not treated yet, or that it is better to use a precise threshold for the image processing, etc.

2.2. oRis simulator

Although being a language, oRis is also a simulator in charge of the agents management which behaviors are

```
void Example.2::main(void) // => 'main()' method redefinition of
{                          // the instance Example.2
 println("First redefinition :");
 println("=> ", this , " : ", ++age , " cycles");
 string secondProgram  = "";
 secondProgram  = "void Example.2::main(void)";
 secondProgram += "{";
 secondProgram += " println(\"Second redefinition :\");";
 secondProgram += " println(\"==> \", this , \" : age=\",++age);";
 secondProgram += "}";
 parse(secondProgram);
}
```

After introducing this code in the simulation, the result becomes as follows:

```
The age of Example.4 is 3
The age of Example.3 is 3
First redefinition :
=> Example.2 : 3 cycles
The age of Example.1 is 3
The age of Example.3 is 4
The age of Example.4 is 4
The age of Example.1 is 4
```



```
Second redefinition :
==> Example.2 : age=4
The age of Example.1 is 5
Second redefinition :
==> Example.2 : age=5
The age of Example.4 is 5
The age of Example.3 is 5
...
```

Fig. 4. Two examples of redefinition of methods on the instance level.

described by the oRis language. Multi-agent simulation requires several entities to be executed in parallel. Therefore, we must make sure that the activation procedure of these autonomous agents does not lead to a bias that would result in a global state for which the execution platform would be responsible: it is imperative that only the algorithmic procedures described in agent behaviors explain the model's global state. Controlling the scheduling procedure for agent behaviors is therefore very important. In oRis, an agent has a `'main()'` method that represents the entrance point of the its behavior. When an instance equipped with a `'main()'` method is created, this method is immediately ready to be executed. When the end of the method is reached, it is automatically relaunched to the beginning. Another way to start a new parallel process for active objects is to split the execution flow by using the `start` primitive (see Fig. 5). This procedure, which generates several execution flows from one flow, is mainly used to assign several activities to one agent. It is very conceivable then that the main behavior of an agent (its `'main()'` method) will generate other supplementary activities. It seems reasonable that an agent could, for example, move around while communicating with others.

The oRis scheduler maintains several execution flows. Fig. 6 shows the data structures used to manage these parallel processes. Each execution flow is represented by a data structure which, in addition to the activity's identifier, contains a context stack and a temporary values stack. The context stack is used to embody function and method nested calls. The temporary data stack is shared by all of the execution flow contexts and makes it possible to stack

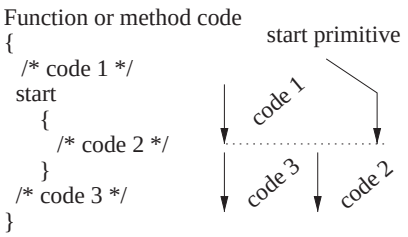


Fig. 5. Splitting and execution flow in oRis.

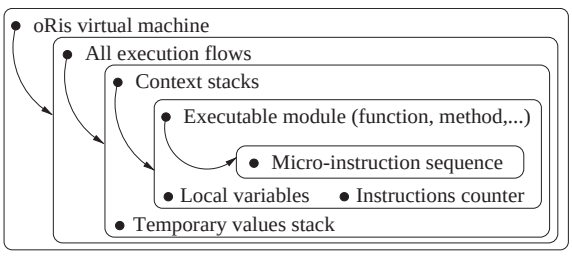


Fig. 6. Structure of the oRis virtual machine.

the parameters of an executable module before its call, and to retrieve the stacked result when the called module is terminated. The executable modules are represented by a sequence of micro-instructions. Since these are atomic, activities can only be switched between different flows through the execution of two micro-instructions. The programmer can specify in milliseconds when a switch must take place. In order to ensure that time is shared equally between different activities, we use the notion of execution cycle, which adds the following property to the system: each execution flow goes around one single time per cycle. If new activities appear during the cycle; they are executed in the next cycle to make sure that cycles end. During a cycle, we suggest two activation modes: a fixed order and a random one.

The oRis simulator is written in C++. Then, this implies a deep coupling between oRis and C++. The programmer can specify a connection between an oRis class and a C++ class. The call of native oRis methods initiates the associated C++ methods. Thanks to the oRis/C++ coupling, it is possible to increase the performances of the system by making native methods faster than the equivalent methods defined in oRis Moreover, thanks to the oRis/C++ coupling, we have set up a group of classes for the management and visualization of images which allow to easily achieve image processing applications.

3. Multi-agent image processing applications

In the following part of this paper, we present two applications achieved with oRis. The first application allows to detect concentric striae located on different biological "objects" (age-rings of tree, fish otolith growth rings, etc.) even if they are discontinuous. The second application concerns the 2D gel electrophoresis domain. In this application, we developed agents able to detect spots in the images even if the spots are saturated or very low contrasted. These agents are very similar to the agents designed for the first application.

3.1. A multi-agent system for the detection of concentric striae

One of the major problems encountered during detection of concentric striae in biological images is the lack of continuity perception. We propose an approach to this continuity perception based on a multi-agent system dedicated to concentric striae detection. Each agent can move around on its environment which consists of an image made up of light and dark rings set out concentrically. In this section, we present more particularly an application for fish otolith growth rings detection (see Fig. 2a). An otolith is usually a less than 1 cm long calcified structure. Because of the alternate apparition of opaque rings (winter rings) and translucent rings (summer rings) during its accretionnary growth, its magnification using a microscope with transmitted light

enables to show alternate light and dark concentric rings, which center is called the nucleus. Thus, the profile of such images from the nucleus to the edge of the otolith presents an alternation of peaks and valleys, which can be assimilated with roof edges. The automatic segmentation of this type of images by traditional techniques can be made difficult because, a variation of gray level can appear inside the otolith growth rings creating textures. The purpose of growth rings identification is to acquire data on age and growth of fish population. Such data are needed in a great number of biological and ecological studies to improve stock management. Up to now, this analysis limited to a ring count, for age estimation, is routinely achieved by a human reader, but this task depends on subjectivity of the reader. Ring continuity is a major concept on which human readers base their ring detection.

Our multi-agent system for ring detection is made up of a set of agents named darkening agents and lightening agents. These agents follow either the light rings (lightening agents) or the dark rings (darkening agents) and act on the image. Their actions aim to reinforce the rings by stressing the contrasts allowing thus a reliable detection of these rings, even if they are discontinuous. In order to model our multi-agent system, we use the formalism BRIC (see Fig. 7a). Each agent can pick up a small part of its environment, thus allowing it to make a choice depending on what its sensors receive, but depending also on its internal state. A BRIC representation of an agent is shown Fig. 7b. An agent has three actuators. The first one allows to modify its environment by increasing or decreasing the brightness of a pixel. The other two actuators allow the agent to have rotating movements and go forward. An agent has also three sensors allowing it to obtain information about the environment (see Fig. 8a). The sensors are made up of unit sensors returning the value of a pixel. The three sensors of an agent are a unit sensor ($Sensor_1$) allowing the agent to know if it is located on an already detected stria, and two square-shaped sensors ($Sensor_2$ and $Sensor_3$) made up of unit sensors. These two sensors return the mean of their unit sensors. They are located in front of the agent and distant one from the other. At each step it makes, the agent computes the mean gray levels V_2 and V_3 for its two sensors ($Sensor_2$ and $Sensor_3$) and also computes their difference Δ. Those two square-shaped sensors are used to determine the movements of the agent and, therefore, to detect the rings. A darkening agent tries to move where the values returned by the sensors are minimal. A lightening agent does the opposite. The greater the difference Δ between the two sensors, the more the agent will deviate. From Fig. 8a, we can see the agents' parameters. These parameters are $(x, y) \in R^2$, the position of the agent in the environment; $\theta \in [0, 2\pi]$, the angle between $Sensor_2$ and $Sensor_3$; $L \in N$, the distance between $Sensor_2$ (resp. $Sensor_3$) from the center of the agent; and finally l, the side length of a square-shaped sensor. The sensors allow the agents to receive information from their environment, and depending on their internal state and the data thus obtained, each of them takes decisions, applying them with the help of its actuators. Those internal state are represented by a finite state machine (see Fig. 8b). States and transitions of an agent are detailed hereafter.

- State *initialization*: the agent is placed at random on the image.
- State *nose*: the agent is a (detecting) nose, i.e. it tries to settle on a ring (the color of the ring may be black or white: it depends if it is a lightening or a darkening agent). It turns into a *marker* if it makes a 2π self-rotation.
- State *marker*: the agent reinforces the ring on which it is located. If it is a lightening agent, it increases brightness of the pixels, and if it is a darkening agent, it decreases brightness of the pixels. As soon as the agent has rotated once (natural rotation of 2π), it turns into a *recorder*.
- State *recorder*: the agent memorizes its path until it has rotated once more again and is back to its initial position of memorization. The agent then turns into a *killer*.
- State *killer*: the agent sets a polygon which correspond to the path it has just memorized into the shared memory. Like this, it indicates to the other agents that this ring has been already detected. That kills all other agents located on the same ring. We can say here that the polygon is a poison for the agents. That avoids validation of the same ring several times. It then turns into *validator*.

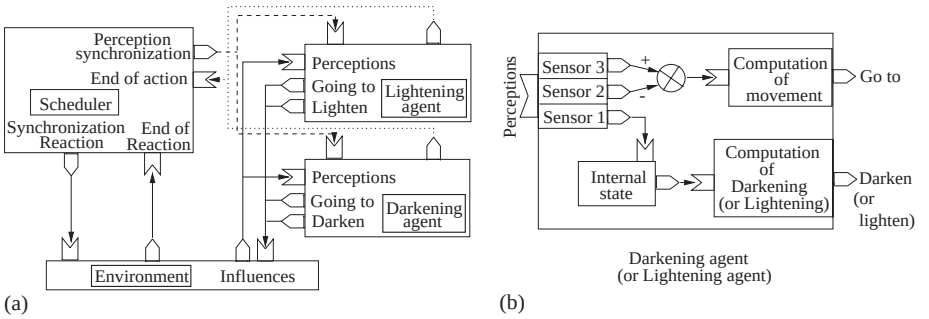


Fig. 7. (a) Our multi-agent system in BRIC, (b) an agent in BRIC.

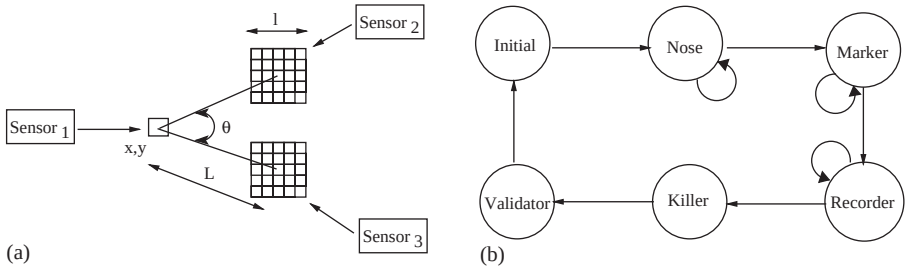
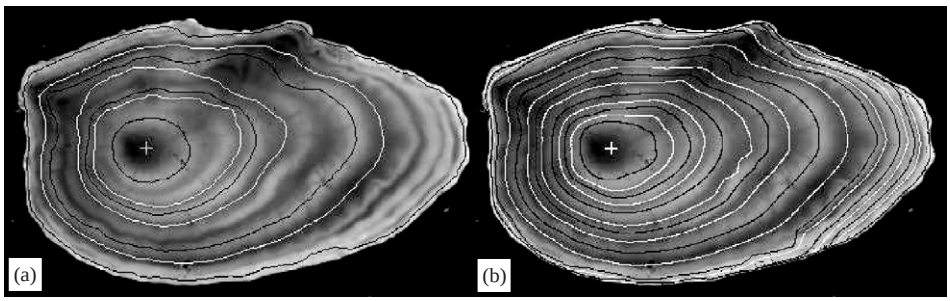


Fig. 8. (a) Agent's sensors, (b) finite state machine describing an agent's behavior.



simple agents high level agents

Fig. 9. Examples of results obtained with simple agents and high-level agents.

- State *validator*: the agent draws on the image a polygon corresponding to the path it has just memorized. The polygon is black or white according to the type of the agent (and then, of course, the type of ring). The agent is then replaced at random in the image (the environment) (state *initialization*).

At each state, if an agent is on a poison or if its age exceed the maximum age, it is killed, replaced at random in the image and turned into a *nose*.

Though its designing cost remains very low, this type of multi-agent system is very efficient (see Fig. 9a). In order to improve the results we also propose to use "high-level" informations such as the shape of the external otolith edge [9]. In this case, a decision can be taken by an agent to recognize whether its path is correct or not and can really improve the results (see Fig. 9b). Fig. 9, the agents' parameters used were $L = 8$, $\theta = \pi/4$ and $l = 1$. The influence of these parameters on quality detection will be studied further in this article. Before this study, let us see a second application dedicated to spot detection on 2D gel electrophoresis images.

3.2. A multi-agent system for the detection of spots

2D gel electrophoresis is a method for the separation and identification of proteins in a sample by displacement in 2 dimensions oriented at right angles to one another. 2D gel electrophoresis is generally used as a component of proteomics and is the step used for the isolation of proteins for further characterization by mass spectroscopy. The isolation is performed firstly by isolelectric focusing (IEF). Isolelectric focusing allows to separate proteins by their charge (pI). Secondly, the other dimension is obtained using the molecular weight through the Sodium Dodecyl Sulphate—PolyAcrylamide Gel Electrophoresis (SDS-PAGE) method. Fig. 2b presents a 2D electrophoresis image. The 2D gel electrophoresis images are far from being perfect. The spots can have very different round shapes: circular, elliptic, etc. Moreover, the images contain trails or spots of water.

The agents used for this application are very similar to the agents designed for striae detection (see Fig. 10) because we have to detect round shapes. An agent always has two frontal sensors ($Sensor_2$ and $Sensor_3$). Moreover, now it has two lateral sensors ($Sensor_4$ and $Sensor_5$). The purpose of these two new sensors are to indicate to the agent its darkest side. This avoids to the agent to turn around a white spot—a spot of water. In such a case, the agent must be reinitialized (transition from *Marker* state to *Initial* state). The agents can turn only in one direction (positive or negative). When an agent is created, this direction is fixed at random. Indeed, the agents must detect spots with the round shapes. This enables them to better resist to the noise, such as the rectilinear shapes superimposed on the spots—the trails. Fig. 11 presents a 2D gel electrophoresis image and the detected spots.

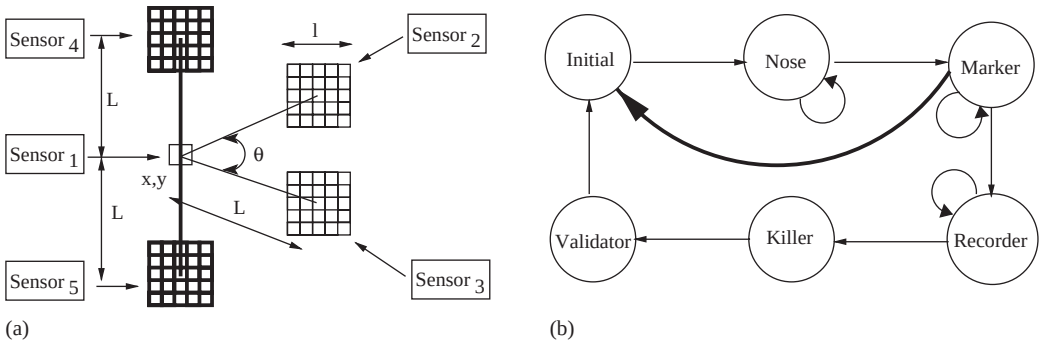


Fig. 10. (a) Agent's sensors, (b) finite state machine describing an agent's behavior.

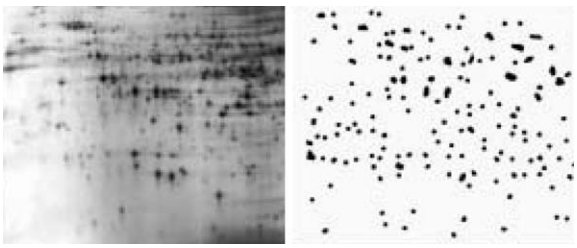


Fig. 11. An image and the detected spots; spots are filled for a better visibility.

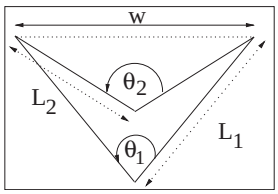


Fig. 12. Two different combinations of (L, θ) for the same w.

4. Parametrization method

In this section, we study the influence of the parameters used for growth ring detection on fish otoliths. First, let us recall our agents' behaviors and their parameters. At each step it makes, an agent computes the gray levels V_2 and V_3 for its two sensors ($Sensor_2$ and $Sensor_3$ on Fig. 8a) and also computes their difference Δ. The greater the difference between the two sensors, the more the agent will deviate. As the two sensors $Sensor_2$ and $Sensor_3$ are square and not oriented, it can be notified that we could first preprocess the image by computing a mean of the image and then use agents with only one pixel wide sensors. The preprocessing could also be different from a mean treatment, for example a gaussian filter. For the rest of the section we will deal with lightening agents, that search for maximums of the image, as the behavior of darkening agents is symmetrical. We will also consider the size of the sensors as one square pixel ($l=1$), as the preprocessing of the image can be treated separately. Then, the different parameters that we need to tune are L, the distance between the central position of the agent and the sensors, and θ, the angle separating the orientation of the two square-shaped sensors. To test different behaviors of the agents with varying parameters, we have created synthetic images containing roof edges. The size of the image is 128*128 pixels, the image features a roof edge which top is located at the abscissa $x=64$. The slope and the width of the roof are variable. Gaussian noise has been added to these images. As the position of the edge is known, we have used it to compare the detection achieved by the agents and the real edge when varying the parameters. We have isolated two parameters that are necessary to tune. These parameters are not completely independent. As a matter of fact, if we assume that the distance w between the two sensors has to be optimized according to the width of the edge, this distance can be obtained with different combinations of L and θ, as seen in Fig. 12. The distance w can then be easily computed using Eq. (1). Moreover, if we need a width w, for a given θ, we can compute the right L with Eq. (2).

$$w = 2L_1 \sin\left(\frac{\theta_1}{2}\right) = 2L_2 \sin\left(\frac{\theta_2}{2}\right) \tag{1}$$

$$L = \frac{w}{2\sin\left(\frac{\theta}{2}\right)} \tag{2}$$

4.1. Parameter θ

If the angle θ is too acute, as we work with numeric images, the distance between $Sensor_2$ and $Sensor_3$ could reach zero, which must not happen if we want to make a comparison between the two sensors. If they are located at the same place, the difference Δ will be zero, so the agent will never modify its direction. For example if we have $L=2$, and $\theta=\pi/13$, the computation of w will give $w=0.48$ which is inferior to 0.5, therefore the numeric distance between the two sensors will be zero. If the angle is too obtuse, like π, the agent's direction will be fuzzy and it will oscillate around

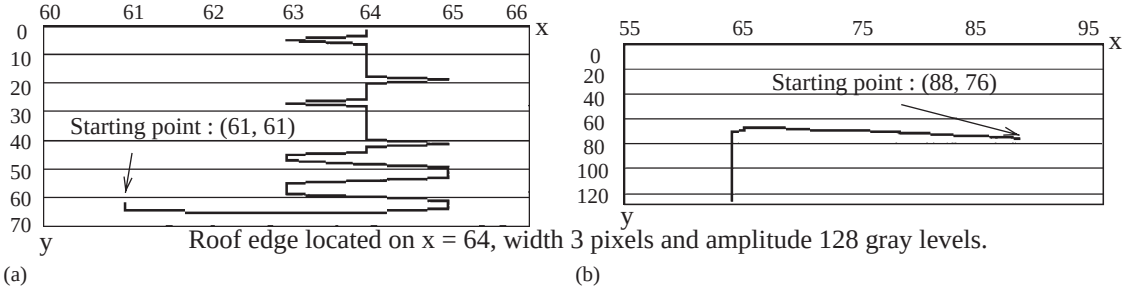


Fig. 13. Path of an agent. (a) $\theta = \pi,\ L = 2$; (b) $\theta = \pi/4,\ L = 2$.

the position of the edge (see Fig. 13a). We have to find a compromise between these two extreme conditions. In the case $\theta = \pi/4$ the direction of the agent will be regular for a straight roof edge (see Fig. 13b). So we will fix $\theta = \pi/4$ and analyze the influence of L on the detection using the Canny criteria.

4.2. Use of the Canny criteria

In order to evaluate the agents taken as edge detectors, we have created a synthetic image containing a perfect edge, to which we have added gaussian noise (see Fig. 14a). We have then compared the edge detected by the agents with the real position of the edge. The Canny criteria allow to compare an ideal edge detection and the edge detection achieved with a particular method. In order to create the image of the detection achieved by the agents, theirs paths are recorded during the processing. Each agent records its path in a common image. The initial gray level of this image is zero; the gray level of each pixel is incremented (arbitrarily of 2 gray levels) each time an agent goes on it. As the agents are first placed at random on the image, some pixels will be marked in the background, that are not edge points, even if the edge is completely detected. Nevertheless the non edge pixels will be less intense than real edge pixels, as the agents are guided by the edge and try to follow it. The paths of the agents on the image shown Fig. 14a are presented in Fig. 14b. After an automatic threshold, the binary image obtained can then be compare with the read edge position using the Canny criteria (see Eq. (3), (4) and (5)). We have added another criterion to express the quality detection before binarizing the paths (see Eq. (6)). We will now try to optimize the Canny criteria by choosing the best value for L when $\theta = \pi/4$.

$$c_1 = 100\frac{number\_of\_detected\_pixels\_that\_are\_not\_edge\_points}{number\_of\_non\_edge\_pixels} \tag{3}$$

$$c_2 = 100\frac{number\_of\_edge\_pixels\_non\_detected}{number\_of\_edge\_pixels} \tag{4}$$

$$c_3 = mean\_distance\_between\_the\_position\_of\_the\_edge\_detected\_and\_the\_real\_position\_of\_the\_edge \tag{5}$$

$$c_4 = mean\_gray\_level\_of\_the\_pixels\_located\_on\_the\_edge - mean\_gray\_level\_of\_the\_pixels\_non\_located\_on\_edge\_points \tag{6}$$

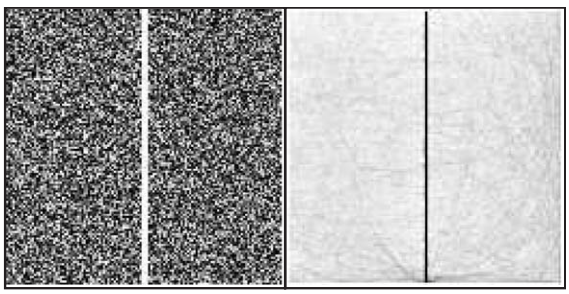


Fig. 14. (a) Roof edge (width 3 pixels, amplitude 60 gray levels) image with gaussian noise of standard deviation 2 gray levels (after an histogram equalization for a better visibility); (b) Paths of the agents on this image with $L = 2$ (inverted gray levels).

4.3. Parameter L

As L increases, the distance between the two sensors of the agents increases too. Thus it is preferable to keep L little in order to have a more precise position of the agent's sensors on each side of the edge. Choosing $L = 1$ would induce problems to discern the two sensors, because of rounding off. If the roof edge image contains low noise, the agents

Table 1
Criteria obtained with different values of L on very noisy image

L	c_1	c_2	c_3	c_4
2	1.27	11.36	0.2	115
3	0.76	9.09	0.18	141
4	0.21	36.36	0.25	101
5	0.29	18.18	0.12	121
6	0.72	0	0	207
7	1.46	0	0	186
8	1.39	0	0	181

can detect any type of roof edge (any amplitude, any width) using $L = 2$. Nevertheless when the signal-to-noise ratio (SNR) is very weak, increasing L can improve the detection. We have taken for example a type of roof edge that has been found in an otolith image. The slope of the roof is 1.52 and the width is 73 pixels. We have then created a synthetic image with this roof and added gaussian noise of standard deviation 2, in order to simulate the type of noise on otolith images. The width is the distance between the two bases of the roof, the slope is the gray level difference between two adjacent pixels on the roof. By increasing L, we have obtained better criteria (see Table 1). Increasing L allows the agents to have a more general view on the noisy roof, and their paths are clearer (see Fig. 15).

4.4. Robustness of the method and application to otolith images

The method has also been tested on an image described in Ziou [4]. This image's size is 150*150 pixels, it features horizontal and vertical lines of slope 50. Noise of standard deviation 10 (SNR = 7 dB) has been added in the upper right corner of the image, and noise of standard deviation 5 (SNR = 10 dB) has been added in the lower left triangle of the image. For reasonable SNR (10 times the logarithm of the roof slope divided by the standard deviation of the noise), we can obtain a good detection when choosing $L = 2$ (see Fig. 16). We applied the same treatment to fish otolith images (512*512 pixels). The results are shown Fig. 17. On fish otolith images, these results show that the best striae localization is obtained for $L = 8$. Let us recall that, the other agents' parameters to be taken into account are l, fixed to 1, and θ, for which the best value is $\pi/4$. Fig. 18 represents a set of results obtained using our striae detection method with these parameters' values.

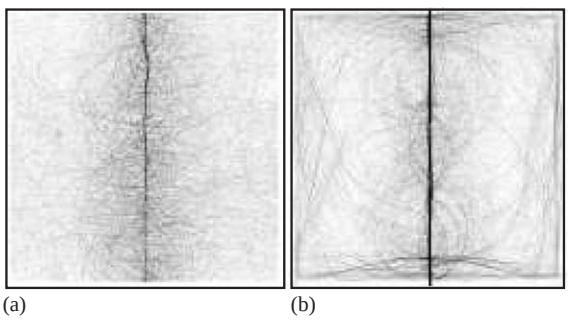


Fig. 15. Path of the agents. (a) $L = 2$; (b) $L = 6$.

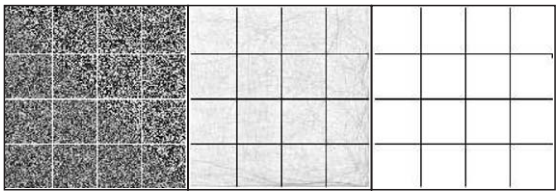


Fig. 16. (a) Noisy image after an histogram equalization; (b) paths of the agents, $L = 2$, inverted gray levels; (c) paths of the agents after an automatic threshold.

Our multi-agent detection system has been tested on a sample of 119 plaice otolith images, which age had been estimated by human experts. The number of dark and light rings on images gives the number of seasons the fish has lived, and knowing the season where the fish has been caught, the experts can estimate the fish age in years. In Table 2, the percentage of good age estimation obtained with our striae detection method is compared with methods previously developed. Our method gives better results than those obtained with a mono-dimensional method applied on otolith images [10]. This method consists in searching intensity extremes on an image profile starting from the nucleus to the otolith edge. Therefore, the structures continuity is not taken into account. Our method also presents better results than the deformable template method [11]. This method uses the shape of the external otolith edge reduced by an homothetic transform centered on the nucleus, in order to search the position of the growth rings. This method can only give an approximate shape to rings. Otherwise the graph method [12] gives similar results with those of our method. This graph method requires a polar transformation of the image using the nucleus as the centre. Peaks (resp. valley), corresponding to

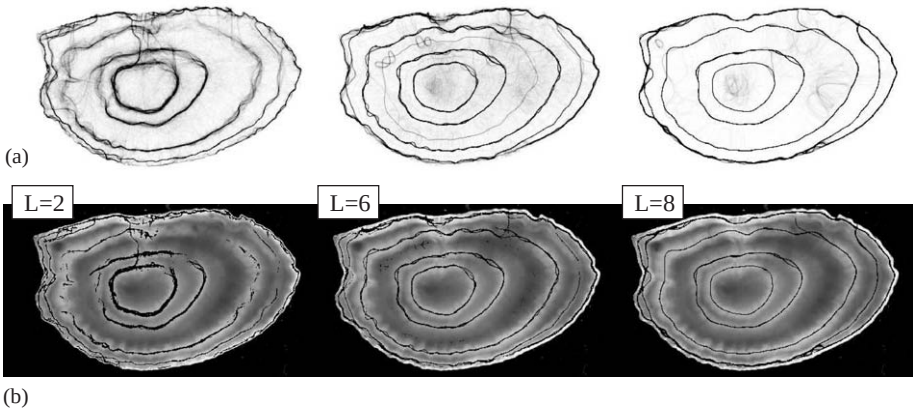


Fig. 17. (a) Paths of the agents for different values of L; (b) paths after threshold.

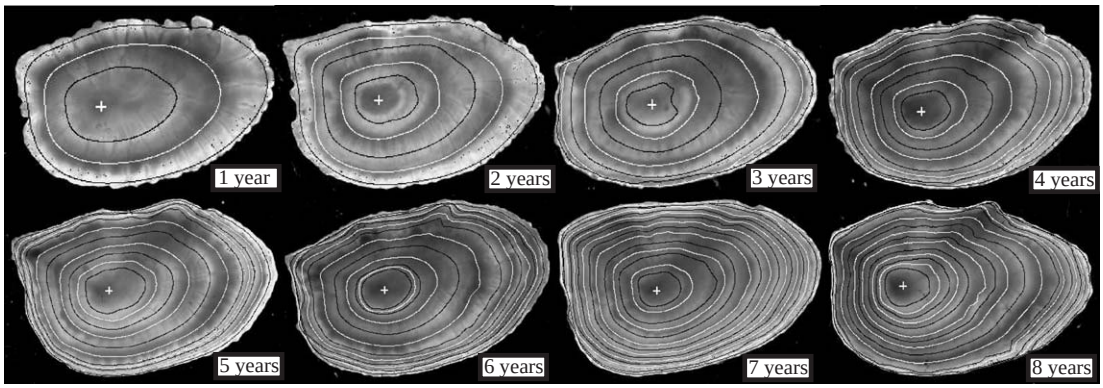


Fig. 18. Results obtained on otoliths of fishes aged from 1 to 8 years old.

Table 2
Percentage of good age estimation obtained with different methods

Age groups	1D	Templates (%)	Graphs (%)	Agents (%)
1–3 years	50%	100	90	87
1–5 years	50%	80	80	82
5–8 years		20	60	68

light (resp. dark) rings, are then extracted using morphological transforms. Objects are then labelled and closest objects are connected to reconstruct rings.

To conclude, our striae detection method allows to perceive continuity of contours in textured, noisy and low contrast images. All the previous methods needed operator intervention to give the nucleus position whereas the pointing is not necessary in our system. The agents are able to detect local edges in the image, while perceiving their continuity by the way they move.

5. Stopping the system by auto-regulation

The stopping of a multi-agent system is one of the most important problems of such a system. In order to solve this problem we propose here the use of immune mechanisms for the regulation of reactive multi-agent systems. More precisely, the aim of this section is to determine how we can take benefit from immune phenomenon to auto-regulate agent populations. This regulation can be made while integrating cell and molecule behaviors into agent's behaviors.

Let us quote for example the mitosis, apoptosis or differentiation that are essential mechanisms during an immune response. The work to do or the problem to be solved are seen as foreign substances, that is antigenic bodies. The agents represent immuno-qualified cells having for goal the antigen inhibition. This process must be efficient, that means it must finish the work ($\neq$ hypo-immune response) and just the work to do ($\neq$ allergy). Each agent inherit from one or several cell behaviors. Those behaviors are extracted from immune cells which have well defined roles. The first consists in detecting the antigen (the work to do), the second in giving alarm on a large scale, the third in increasing the capacity and the precision of the response and the fourth in eliminating the antigen. Our agents use these roles to mime an immune response. The immune system was chosen to model different aspects of multi-agent systems because it is compound with autonomous entities, able to cooperate, having behaviors, receptors and means of action. Therefore, a cell is very close to the agent concept. The immune system is also able to distinguish "self" and "non-self". Like this, it can detect the work to do among 10^{16} different patterns. Thus, this system is flexible and adaptative, what gets an unquestionable advantage in environments with strong variability (like for aerial images [13]). This number of possible shapes is very important, but it can be reduce for the need of simulation [14]. Another reason to choose the immune system is that it is quasi-optimal in the power of the answer to eliminate the antigen, which would allow a quasi-optimal use of the computer resources during multi-agent processes: the notion of regulation [15].

5.1. Immune mechanisms for multi-agent systems regulation

We approach in this part the use of a certain number of immune mechanisms for the development of self-regulated multi-agent systems. Thus, we describe several types of immune phenomena implied into self-regulation. We see the negative and positive selections allowing to avoid the presence of cells (or agents) useless or disturbing the system. Then we see the phenomena of activation, differentiation, proliferation and programmed cellular death (apoptosis). The latter are the basis of the mechanisms for automatic regulation during an immune response. The cooperation between the T cells and the B cells within the immune system also allows a limitation of the risks of drift (underprocessing of the antigen or image detection unexpected) while allowing an adaptation located in time and space. But we do not develop this mechanism in this paper. Finally, we propose an architecture of multi-agent systems based on the immune principles quoted above.

Negative and positive selections are key mechanisms of the immune system. Thanks to these selections, the immune system is able to distinguish "self" and "non-self". Our approach concerns the optimization of agent populations using immune regulation. That is why, we do not only see these selections as a mean of learning, but also as a regulation principle. In fact, the selections eliminate entities that useless or over active. More an agent is reactive, more the machine resources needed are reduced. It consists for an agent that is not well-adapted (too reactive or without reactivity), to destroy itself before being activated by the system. The difficulty is to determine the two thresholds of selection which are today empirically defined into our systems.

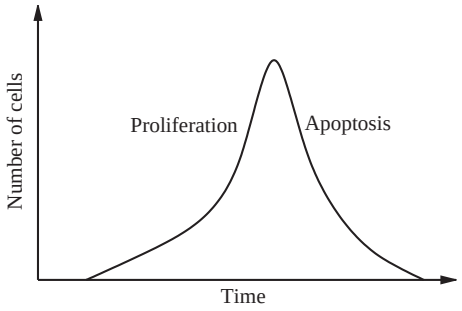


Fig. 19. The two main stages of an immune response.

Thanks to activation, differentiation (maturation), proliferation and apoptosis mechanisms, the immune system is able to specifically increase or reduce its potential against one or several antigens. More precisely, the immune system increases the number of cells directed against an antigen when this antigen is in the body and reduces it when it is eliminated (see Fig. 19).

The activation is the first step into an immune response. During the activation, a cell changes its morphology and its role. So, we observe structural modifications and behavioral modifications. For example, the structural changes can improve the cell mobility, the sensitivity to chemical messengers and/or can change the life duration. The new behavior implies new aims. For instance, a macrophage having phagocitized a foreign substance becomes able to present this antigen to T4 cell. The activation depends on the internal state of cells and on the local environment situation. Here, we restrict our activation study to the T and B cells. A cell (B or T) must receive two types of signals to become activated. The first one is an antigenic signal and the second one is a proliferating signal (interleukines). There are two ways concerning the antigenic signal: a direct activation thanks to endocytosis (B cell and antigen) and a cell mediated activation (Antigen Presenting Cell and T4 cells). The direct activation is fast but can implies an over-reaction (somatic hypermutation like allergy). The cell-mediated activation moderates the direct activation and secures the response. For the auto-regulation, the activation is essential: only activated agents can proliferate. Inversely, an agent that received a partial activation decrease its life duration (that means the agent is no more useful or is not adapted to the problem). To summary, we can say that there are three types of states for an agent. Firstly, the agent is activated, so it

get new properties, new behaviors and increase its life duration (*maturity* threshold). Secondly, the agent is incompletely activated and then it reduces its life duration (*maturity* threshold). And thirdly, the agent is non-activated, it just waits for an hypothetic activation.

The differentiation (maturation) is implied into cell specialization. Like this, the immune system improves its efficiency against an antigen. This mechanism is linked to external signal (into the local environment of cells) or to the age of cells (maturation). The differentiation generates, like activation, structural and behavioral changes for the cells. We treat the differentiation of the agents via the dynamic properties of our oRis language which allow instance granularity. The behavior of a differentiated agent realize a qualitative evolution. It increases the agent's efficiency to solve the problem. This augmentation has repercussions for all the system that becomes more accurate.

The proliferation increases the quantitative (number of cells) and qualitative (improvement of the affinity with the antigen) immune capacity to inhibit the antigen. This phenomenon is called mitosis. The proliferation corresponds to the creation of new agents. The new created agents are structurally and behaviorally close to their creators but not exactly the same to allow the adaptation of the system.

The apoptosis corresponds to the programmed cellular death. This mechanism occurs when a cell is not adapted to the antigen elimination. Thus, useless cells are destroyed. The agents that are able to kill themselves take into account their internal state and the stimuli perceived by their receptors. This phenomenon essentially takes place when the activation is incomplete. The apoptosis behavior is describe by a logical circuit (see Fig. 20).

We can group the different mechanisms described above into a system that reproduces a part of the immune system (see Fig. 21). In such a system, during stage 1, we have first of all the creation of the agents. Then, stage 2, the created agents undergo two selections. The positive selection concerns the agents able to deal with a given problem. The negative selection applies to the agents having a too important reactivity to treat the problem. This second selection makes it possible to avoid the system treating information in a wrong way. That corresponds for the immune system

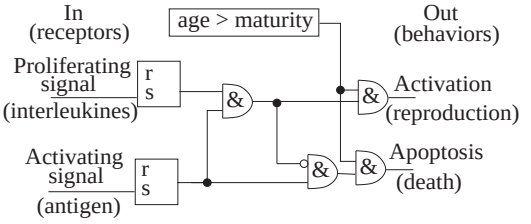


Fig. 20. Logical circuit for activation and apoptosis behaviors. The *maturity* threshold is fitted during the time according to the usefulness of the agent in the system (negative and positive selection).

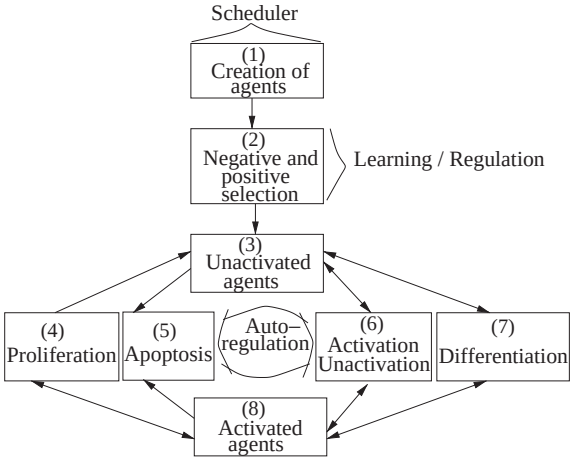


Fig. 21. General structure of our immune oriented multi-agent system.

to the destruction of cell implied in the autoimmune diseases. Stages 3–8 concern the regulation phase of our system. We used these immune oriented multi-agent system to biological image processing and more precisely on striae detection.

5.2. Application to image processing

Thus, we found interesting to extend this regulation concept to the multi-agent systems which, a priori, do not have any relationship with immunology. Then, it was necessary to extend the notion of antigens to the notion of work to do and the notion of interleukin to the stimulant concept. Thus, an agent which executes some work may inherit from the B-cell properties to completely act in the regulation of the system it belongs. Thus, our image processing agents inherit from the B-cell properties previously described. Now they are able to reproduce or destroy themselves. With this new model a processing for the detection of concentric striae starts with the creation of two agents (a lightening agent and a darkening agent) placed at random in the image to treat (the environment). Stimulants are also placed at random in the environment. When a stria is detected by an agent, this agent generates stimulants. These stimulants will be placed at random in the image. Thanks to the stimulants, the agents will multiply and achieve the processing. When the stimulants disappear, the agents die. Fig. 22 presents the evolution of the number of agents with time. This approach gives interesting results because the system is able to stop by itself. Its duration is short in terms of logical time (internal clock of the simulator) but is longer when talking about physical time (external clock of the simulator). Moreover, the number of agents is not optimal. Effectively, the maximum number of agents depends on the number of stimulants initially placed on the image. In order to optimize the number of agents, it would be necessary to use agents specialized in the detection

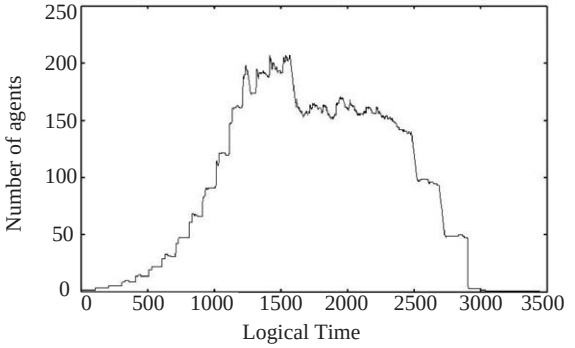


Fig. 22. Evolution of the number of agents with time.

of the work to achieve, as the macrophages and TCD4 B-cells do during an immune answer. These agents would place stimulants on the striae which were not already detected. This would stimulate the agents which directly work on the image.

The use of immune regulation into multi-agent systems is interesting when the number of agents is relatively important. Concerning the detection on otolith images, the results obtained are very similar to ones obtained with the non-regulated system. The immune-oriented multi-agent system allows us to attenuate the number of agent influence: if this number is too little the system increases it, if not, agents use the apoptosis behavior. Like this, the system adapts itself according to its environment. We have also shown that immune-oriented multi-agent system permits to create image segmentation systems without any global controller nor central decisional system. The immune responses offer to computer scientists many regulation principles that can be included into multi-agent systems to optimize the number of agents [16].

6. Conclusion

In this paper, we have presented a parallel image processing system based on the concept of reactive agents. This means that, in our system, each agent has a very simple behavior which allows it to take a decision (find out an edge, a region, etc.) according to its position in the image and to the information enclosed in it. Our system lies in the oRis language, which allows to describe very finely and simply the agents' behaviors. We have also described a multi-agent system dedicated to roof edge detection. The system proposed here is composed of several agents whose individual task is to detect local extremes on a grayscale image. For this aim the agents are provided with sensors on the gray levels of the image. By computing the mean gray level of two or more sensors placed around of it, the agent, will decide to turn in a particular direction according to its behavior. We have analyzed the influence of different parameters ruling the agents' behavior. We have found relations between the edge's structure and the optimal parameters. The agents we have dealt with have only low-level information about the image, so that they can be adapted easily to different type of images (fish otolith, 2D gel electrophoresis, etc.). The stopping of the multi-agent systems is one of the most important problems of such a system. In order to solve this problem, we have studied the stopping of our multi-agent system by auto-regulation through the use of immune mechanisms. We can then talk about immune oriented multi-agent system.

Acknowledgements

Thanks to IFREMER Brest (France), LASAA, for otolith images and INRA Bordeaux (France), for 2D gel electrophoresis images.

References

[1] J. Ferber, Multi-Agent System: An Introduction to Distributed Artificial Intelligence, Addison-Wesley, Reading, MA, 1999.
[2] F. Harrouet, P. Reignier, J. Tisseau, Multiagent systems and virtual reality for interactive prototyping, Proceedings of ISAS'99, Orlando, USA, Vol. 3, 1999, pp. 50–57.
[3] J.F. Canny, A computational approach to edge detection, IEEE Trans. Pattern Anal. Mach. Intell. 8 (1986) 679–698.
[4] D. Ziou, Line detection using an optimal IIR filter, Pattern Recognition 24 (1991) 465–478.
[5] R.M. Haralick, Ridges and valleys on digital images, Comput. Vision Graphics Image Process. 22 (1983) 28–38.
[6] R.W. Smith, Computer processing on line images: a survey, Pattern Recognition 20 (1987) 7–15.
[7] Maximilian Lückenhaus, A multi-agent system for parallelizing image analysis tasks, Proceedings of IAS'98, Sapporo, Japan, 1998, pp. 579–586.
[8] J. Liu, Y.Y. Tang, Adaptative image segmentation with distributed behavior based agents, IEEE Trans. Pattern Anal. Mach. Intell. 6 (1999) 544–551.
[9] A. Guillaud, H. Troadec, A. Benzinou, J. Le Bihan, V. Rodin, A multi-agent system for edge detection and continuity perception on fish otolith images, EURASIP J. Appl. Signal Process. 7 (2002) 746–753.
[10] H.C. Welleman, F. Storbeck, Automatic ageing of plaice (*Pleuronectes plasseta*) otoliths by means of image analysis, in: Secor, Dean, Campana (Eds.), Recent Developments in Fish Otolith Research, University of South Carolina Press, Columbia, USA, 1995, pp. 271–282.
[11] H. Troadec, A. Benzinou, V. Rodin, J. Le Bihan, Use of deformable templates for otolith 2D growth ring detection by digital image processing: application to plaice (*Pleuronectes plasseta*) otoliths, Fish. Res. J. 46 (2000) 155–163.
[12] V. Rodin, H. Troadec, H. De Pontual, A. Benzinou, J. Tisseau, J. Le Bihan, Growth ring detection on fish otoliths by a graph construction, Proceedings of IEEE ICIP'96, Lausanne, Switzerland, Vol. II, 1996, pp. 685–688.

[13] D.F. McCoy, Artificial immune systems and aerial image segmentation, Proceedings of IEEE SMC'97, Orlando, USA, 1997, pp. 867–872.
[14] D.J. Smith, D.H. Ackley, S. Forrest, A.S. Perelson, Using lazy evaluation to simulate realistic-size repertoires in models of the immune system, Bull. Math. Biol. 60 (1997) 647–658.
[15] D. Dasgupta, Artificial Immune Systems and Their Applications, Springer, Berlin, 1999.
[16] P. Ballet, V. Rodin, J. Tisseau, Immune mechanisms to regulate multi-agent systems, Proceedings of GECCO'00, Las Vegas, USA, 2000, pp. 33–35.

About the Author—VINCENT RODIN was born in 1966. Lecturer at École Nationale d'Ingénieurs de Brest (France), he is working on image processing, multi-agent systems and computer simulation of biological processes.

About the Author—ABDESSLAM BENZINOU was born in 1970. Lecturer at École Nationale d'Ingénieurs de Brest (France), he is working on signal analysis, computer vision and pattern recognition.

About the Author—ANNE GUILLAUD was born in 1974. She has studied at École Nationale d'Ingénieurs de Brest (France). She received in 2000 the Ph.D. degree in signal and image processing.

About the Author—PASCAL BALLET was born in 1971. He has studied at École Nationale d'Ingénieurs de Brest (France). He is currently lecturer at University of Brest (France) and he is working on multi-agent systems and computer simulation of biological processes.

About the Author—FABRICE HARROUET was born in 1971. Lecturer at École Nationale d'Ingénieurs de Brest (France), he is working on multi-agent systems and virtual reality.

About the Author—JACQUES TISSEAU was born in 1953. Professor at École Nationale d'Ingénieurs de Brest (France), he is working on multi-agent systems, virtual reality and computer simulation of biological processes.

About the Author—JEAN LE BIHAN was born in France. Professor at École Nationale d'Ingénieurs de Brest (France), he is working in the area of optical communication systems and image processing.