Agents situés dans l'image et organisés en pyramide irrégulière: Contribution à la segmentation par une approche d'agrégation coopérative et adaptative

Edouard Duchesnay

N ° ordre : 2643
de la thèse

THÈSE

présentée

DEVANT L'UNIVERSITÉ DE RENNES 1

pour obtenir

le grade de : ***DOCTEUR DE L'UNIVERSITÉ DE RENNES 1***

Mention : Traitement du Signal et Télécommunications

PAR

Édouard Duchesnay

Équipe d'accueil : Laboratoire du Traitement du Signal et de l'Image/
Groupe de Recherche en Architectures Intelligentes Distribuées
École doctorale : Mathématiques, Informatique, Signal, Électronique, Télécommunications
Composante universitaire : Structure et Propriétés de la Matière

Agents situés dans l'image et organisés en pyramide irrégulière

Contribution à la segmentation par une approche d'agrégation coopérative et adaptative

Soutenue 13 Décembre 2001 devant la commission d'examen

COMPOSITION DU JURY :

Christian Roux	président
Marinette Revenu	rapporteur
Jean-Michel Jolion	rapporteur
Mireille Garreau	examinateur
Lotfi Senhadji	examinateur
Jean-Jacques Montois	directeur de thèse

Remerciements

Je tiens à remercier tout particulièrement M. Jean-Jacques Montois, maître de conférence HDR à l'IUT de St-Malo, pour m'avoir accueilli et impliqué au sein du GRAID (Groupe de Recherche en Architectures Intelligentes Distribuées), groupe qu'il avait constitué lors de son arrivée à St-Malo. Sa confiance, son constant et effectif encadrement scientifique, la pertinence de ses remarques et, "last but not least", ses qualités humaines ont permis la création dans un site délocalisé d'une thématique de recherche délicate car émergente et interdisciplinaire.

Je remercie Mme Marinette Revenu, Professeur à l'université de Caen, pour avoir accepté d'être rapporteur de ma thèse ainsi que pour ses remarques judicieuses qui ont permis d'en préciser certains aspects scientifiques. J'ai été sensible à l'intérêt qu'elle a porté très tôt à notre thématique de recherche.

Je remercie également M. Jean-Michel Jolion, Professeur à l'INSA de Lyon, d'avoir accepté d'être rapporteur de ma thèse et pour le soin qu'il a consacré à cette tâche. Ses travaux sur les pyramides irrégulières sont à l'origine de leur utilisation dans notre architecture.

Je remercie M. Christian Roux , Professeur à télécom Bretagne, pour avoir accepté de présider le jury de thèse.

Je remercie M. Jean-Louis Coatrieux, directeur de recherche à l'INSERM et directeur du LTSI (Laboratoire de Traitement du Signal et de l'Image), dont le GRAID fait partie, pour la confiance qu'il a accordé à notre jeune équipe ainsi que pour m'avoir accueilli au sein d'un laboratoire motivant et agréable à vivre.

En outre, je suis très sensible à l'honneur que me font Mme Mireille Garreau, maître de conférence, ainsi que le Professeur Lotfi Senhadji, de l'université de Rennes 1, d'accepter de faire partie de mon jury de thèse, manifestant ainsi leur intérêt pour mon travail.

Je tiens à remercier Yann Jacquelet doctorant et voisin de table au LTSI/GRAID pour nos longues discussions sur l'IA et l'image ainsi que pour l'aide qu'il m'a apporté lors des expérimentations.

Je n'oublie pas non plus l'équipe enseignante du département GTR de l'IUT de St-Malo pour l'ambiance de travail, la bienveillance à l'égard de la recherche et la compréhension dont ils ont fait preuve lors de ma rédaction de thèse.

Comment oublier Abel Kinié maître de conférence à l'IUT de St-Malo pour avoir amené un peu de soleil camerounais dans nos contrées bretonnes.

Comme rien ne saurait se faire sans financement, je souhaiterai remercier : la mairie de St-Malo, l'université de Rennes 1, l'IUT de Rennes et le LTSI. Ces structures et leurs responsables respectifs ont bien voulu croire en nous en nous fournissant les moyens indispensables à l'émergence de la recherche à St-Malo.

Il y a aussi tous ceux qui ont favorisé l'aboutissement de ma thèse d'une manière personnelle directe ou indirecte.

J'aurai tout d'abord une pensée pour toi, Anna, tu as patiemment relu ma prose (je ne suis pas un littéraire). Tu as su garder ma motivation intacte en me faisant rêver qu'un jour nous lirions la Camif devant un feu durant nos longues soirées d'hiver !

À mes parents, ma grand mère et mon frère (qui a aussi choisi de faire carrière dans l'informatique et commence à parler bizarrement).

À mes amis de "quinze ans" et mes "nouveaux" amis bretons (qui ne sont pas tous alcooliques) pour nos diverses activités diurnes et nocturnes et pour m'avoir souvent demandé "mais au fait, c'est quoi le sujet de ta thèse ?" puis avoir attentivement écouté la réponse.

Je dois aussi ce goût pour la recherche et l'enseignement à l'influence qu'ont eu sur moi trois enseignants de l'EPITA : Mohamed Regragui pour m'avoir fait aimer les mathématiques, Pierre Oysel pour m'avoir fait apprécier les belles choses de l'informatique (avec beaucoup de λ, (cdr ...) et autre reg-exp) et Miklos Santha pour m'avoir intéressé à l'algorithmique.

Table des matières

Chapitre

1

Introduction

1.1 Positionnement de la thématique de la thèse

La conception d'un système de perception, limité à l'analyse d'image ou intégrant toutes les étapes de la vision, est un problème complexe dont il convient d'identifier clairement les différents niveaux d'analyse. Les trois niveaux d'analyse [Mar82] des systèmes de vision proposés par David Marr nous aident à situer les thématiques abordées par cette thèse. Ces dernières sont au nombre de trois, et elles sont illustrées figure 1.1.

1. Méthodologie. On cherche une méthodologie applicable en segmentation d'image et plus largement en vision. L'identification de nouvelles contraintes est une orientation dans la recherche de nouvelles méthodologies.

Il est cependant nécessaire de préciser la nature de ces contraintes, par conséquent, dans le cadre général des méthodologies systémiques, nous en proposons une basée sur trois paradigmes : l'intégration, l'adaptation (focalisation) et la coopération.

2. Architecture logicielle. À la manière des langages objets qui sont la traduction d'une méthodologie, nous proposons une architecture logicielle, basée sur des **agents situés dans l'image**, permettant de penser et de mettre en œuvre notre méthodologie.

3. Méthodologie logicielle. L'utilisation des SMA dans la conception d'un système d'information nécessite l'emploi de méthodologies logicielles "systémiques" qui mettent l'accent sur la conception individuelle, les interactions au sein d'une structure organisationnelle et les caractéristiques globales émergentes.

Nous n'avons pas comme objectif de proposer un nouvelle méthode de segmentation, nos travaux se situent plus sur un axe transversal et méthodologique sur lequel on cherche un cadre conceptuel riche et flexible permettant une expression harmonieuse et efficace des activités de la vision par ordinateur.

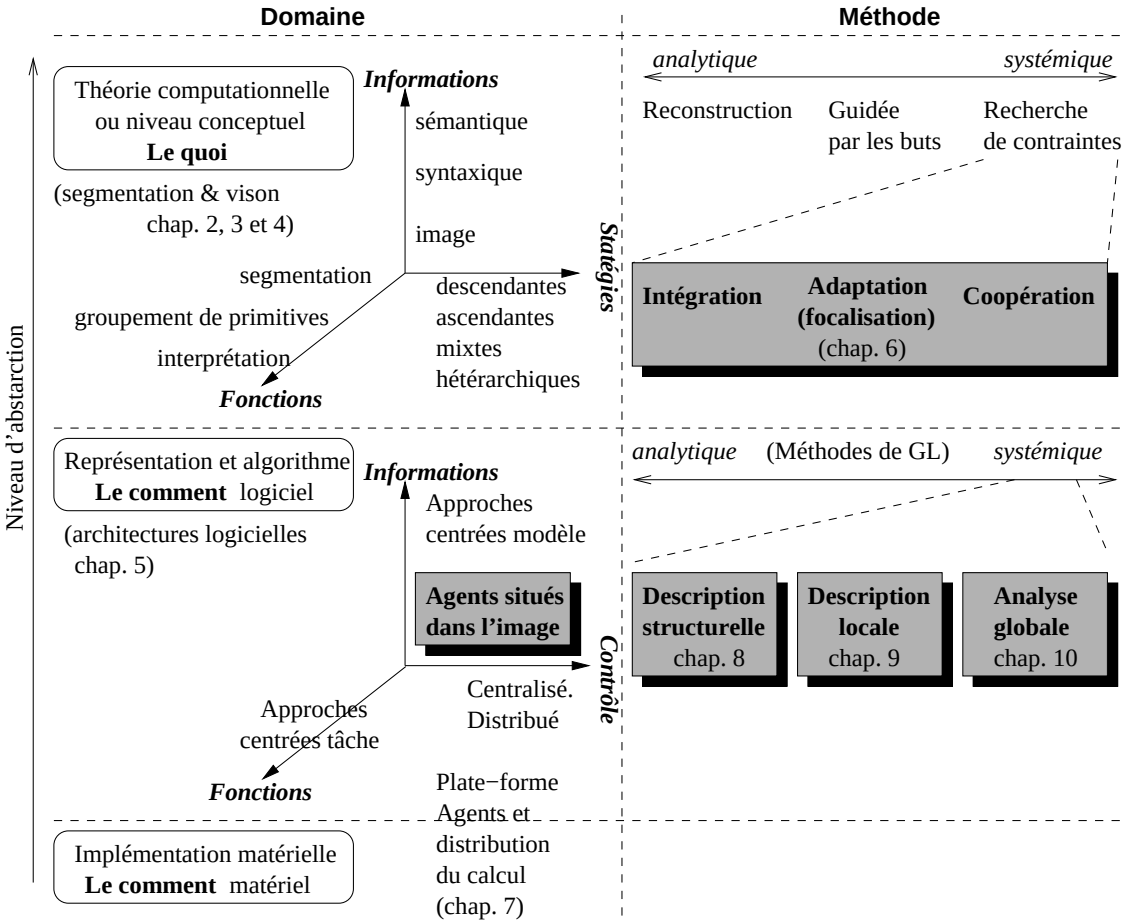


FIG. 1.1 : *Les trois niveaux d'analyse des systèmes vision, et, en grisé, les thèmes abordés par la thèse.*

Nous allons utiliser une métaphore issue de l'évolution des langages informatiques pour illustrer nos préoccupations. Deux programmes ont la même fonctionnalité (le quoi), le premier est écrit en langage assembleur, le deuxième en langage objet (ils diffèrent sur la méthodologie de conception et sur le comment logiciel). De nombreuses critiques peuvent être adressées au programme conçu en objet (ainsi qu'à notre approche) : il est plus lent ; plus volumineux ; il réalise le même objectif ; ce n'est qu'un habillage conceptuel dans lequel des vocables comme "envoi de message" ont remplacé "appel de fonction". Ces critiques sont justifiées sur le niveau d'analyse de la théorie computationnelle, mais on ne peut négliger l'apport conceptuel et méthodologique des méthodes objet au niveau de la réalisation logicielle.

Si cet exemple n'est qu'une métaphore et que nous proposons bien une approche agent, et non simplement objet, il illustre bien le positionnement de nos travaux, les apports que l'on peut en espérer et les critiques que l'on peut leur adresser.

1.2 État de l'art et identification des besoins (chapitres 2 à 5)

Les langages de communication utilisés par les humains sont d'avantage qu'un simple habillage de la pensée, ils la structurent. De la même manière, une architecture logicielle de contrôle doit fournir un cadre **méthodologique** permettant une intégration et une expression harmonieuse des différents composants du système.

Il est donc indispensable d'identifier au préalable les besoins et les problèmes rencontrés lors des différentes étapes du traitement des informations par le système de perception. Cette étape consiste à remonter au niveau de la théorie computationnelle pour identifier les besoins et les problèmes afin de pouvoir fournir un outil logiciel adapté.

- La segmentation d'image étant le domaine d'application central de notre architecture de contrôle, nous lui consacrons les *chapitres 2* et *3* (ce dernier chapitre étant dédié aux représentations pyramidales). L'étude de différentes approches de segmentation nous a permis d'identifier trois besoins rencontrés à ce niveau de la vision :

1. les besoins d'adaptation locale des traitements ;
2. les besoins d'intégration et d'expression plus ou moins précise de l'information *a priori* ;
3. les besoins d'approches coopératives.

- Notre objectif futur étant d'intégrer les étapes du moyen et du haut niveau de la vision, nous présentons un état de l'art des systèmes de vision au *chapitre 4*. Plusieurs raisons justifient cette étude :
  - L'élargissement du domaine d'analyse est indispensable à la conception d'un module de segmentation dont la vocation est d'être intégré dans un système plus vaste.
  - Les recherches sur les systèmes de vision ont donné lieu à des réflexions sur une méthodologie plus générale de conception des systèmes de vision incluant l'étape de segmentation. Ces courants de pensée, dont nous essayons de rendre compte, ont grandement influencé notre approche.
  - Les systèmes de vision fournissent des exemples d'architectures logicielles de contrôle que nous ne pouvions ignorer. Le *chapitre 5* leur est consacré.

L'étude des systèmes de vision nous a permis d'identifier trois types de besoins :

1. besoins d'intégration de modules hétérogènes traduisant des connaissances opératoires et descriptives ;
2. besoins de spécification du problème du contrôle du système et des stratégies de reconnaissance d'objets ;
3. besoins de formalismes pour la représentation des connaissances.

1.3 Propositions (chapitres 6 à 10)

Comme évoqué au *chapitre 6* (chapitre introductif de la deuxième partie) les propriétés des **systèmes multi-agents situés dans l'image** en font une approche adaptée à la conception d'une architecture logicielle de contrôle permettant la mise en œuvre des besoins identifiés ci-dessus. Ils sont l'expression en informatique d'un courant de pensée contemporain (systémique, holisme, émergentisme) visant à complexifier l'analyse des systèmes par une observation globale d'interactions locales. On retrouve des traces de cette approche, basée sur la dynamique et l'incarnation des systèmes qui s'opposent à l'approche analytique classique, dans la recherche en systèmes de vision (vision active, intentionnelle, animée).

Il nous semble pertinent d'inscrire notre démarche dans ce courant de pensée qui fournit des pistes sur la manière de rendre compte de la complexité d'un système de vision.

Nous présentons au *chapitre 7* une plate-forme multi-agents générique pouvant servir à d'autres applications que l'image.

Nous utilisons cette plate-forme pour construire une architecture logicielle d'agents situés dans l'image. La description et l'analyse de cette dernière respecte une méthologie de conception de logiciels articulée en trois niveaux :

1. Description *globale* et *structurelle* de **l'organisation** regroupant les agents. Cette étape de description, présentée au *chapitre 8*, s'attache à établir les liens entre agents. Nous proposons comme élément organisationnel la **pyramide irrégulière** qui va imposer sa structure à la population d'agents afin de garantir un comportement globalement contrôlable et convergent de ces derniers.

2. Description *locale* et *fonctionnelle* des agents composant le système. Au cours du *chapitre 9* nous donnons une description individuelle du contrôle des comportements de l'agent et des interactions locales avec ses voisins dans l'organisation évoquée ci-dessus.

Dans ce cadre conceptuel, nous proposons une mise en œuvre particulière de l'architecture logicielle de contrôle dans laquelle deux familles d'agents, qui traduisent des primitives région et contour, interagissent au sein de la pyramide.

Notre objectif est de montrer comment cette méthodologie permet une implémentation riche, flexible et distribuée des aspects précédemment identifiés ; à savoir : l'adaptation locale, l'intégration et l'expression d'incertitudes dans l'information *a priori* et des traitements coopératifs région/région et région/contour.

3. Finalement, une analyse *globale*, *comparative* et *fonctionnelle* présentée au *chapitre 10* vérifie que l'ensemble des interactions locales produit une bonne segmentation des images. Nous comparons notre approche avec d'autres méthodes de segmentation sur des images médicales et des images de synthèse.

Première partie

État de l'art et identification des besoins

Chapitre

2

Segmentation

La segmentation d'image étant le domaine d'application central de notre architecture de contrôle, il est indispensable d'identifier au préalable les besoins et les problèmes rencontrés lors de cette étape de traitement de l'image.

Dans ce chapitre nous présenterons certaines méthodes de segmentation puis nous conclurons avec les approches coopératives.

Lors de cette étude des techniques de segmentation, nous identifierons trois besoins qui, selon nous, sont essentiels si l'on souhaite améliorer la qualité et la robustesse de l'analyse d'image. Ces besoins sont :

1. les besoins d'adaptation locale des traitements ;
2. les besoins d'intégration et d'expression plus ou moins précisément de l'information *a priori* ;
3. les besoins d'approches coopératives entre plusieurs méthodes.

2.1 Introduction

La segmentation est une étape essentielle du processus de vision. Elle a pour objectif de partitionner l'image en zones stationnaires qu'on espère les plus proches possible des objets ou régions d'objets réellement présents dans l'image. En règle générale, les régions extraites ne correspondent que partiellement aux objets et une interaction avec les couches haut niveau de vision , qui utilisent des connaissances spécifiques au domaine, s'avère nécessaire.

Cette introduction de connaissances spécifiques au domaine divise les chercheurs du domaine : certains considérent que cette étape peut s'effectuer indépendamment du domaine "à l'aveugle", quitte à reporter la complexité du traitement dans les étapes de vision haut niveau ; d'autres chercheurs, au contraire, pensent qu'il faut y introduire au plus tôt de l'information *a priori* afin d'éviter les erreurs de partitionnement qui compliquent ou même induisent en erreur l'interprétation conduisant, dans le meilleur des cas, à une reprise de la segmentation avec de nouveaux paramètres.

Quoi qu'il en soit, l'étape de segmentation doit transformer l'image en une information quantitativement réduite ou/et qualitativement rehaussée. Cette transformation par regroupement ou classification est confrontée aux problèmes d'ambiguïtés et de bruit qui affecte certains pixels de l'image.

Les techniques de segmentation peuvent être classées en trois grandes familles :

1. les approches de type classification dans lesquelles nous retrouvons des méthodes monodimensionnelles (seuillage), multidimensionnelles et markoviennes ;
2. les approches frontières ;
3. les approches régions.

Les deuxième et troisième approches étant duales, la frontière entourant chaque région définit un contour fermé et chaque contour fermé décrit une région.

Pour chaque approche de segmentation abordée, nous essayerons d'en dégager les limites, puis nous évoquerons les solutions proposées en réponse à ces problèmes. Ces solutions s'articulent souvent autour des trois principes suivants :

1. adaptation locale de la stratégie de segmentation ;
2. coopération de plusieurs techniques ;
3. introduction d'informations *a priori*.

Ces trois principes justifient en partie l'utilisation d'une architecture basée sur des agents qui utilisent à bon escient des techniques classiques de segmentation. Par "à bon escient" nous entendons : adaptation locale et contextuelle des traitements et des paramètres, aptitude à utiliser plusieurs approches de façon coopérative et capacité à représenter et à mettre en œuvre les connaissances que l'on a du domaine.

2.2 Classification

2.2.1 Méthodes monodimensionnelles ou seuillage

L'attribut analysé est, dans la plupart des cas, le niveau de gris. Une analyse de l'histogramme de l'image permet de dégager automatiquement des classes et donc des seuils séparant ces classes. Les seuils sont les minima locaux de l'histogramme permettant de séparer les modes (classes) entre eux. Pour les images dont l'histogramme ne possède pas de "vallées" nettes (minima faibles ou locaux), des techniques [Ros82] permettent dans l'histogramme d'en dégager des modes.

Il est parfois nécessaire d'adopter une approche locale lorsque l'histogramme est globalement inexploitable (histogramme global unimodal résultant de la superposition de plusieurs modes). Dans la méthode proposée par Nakagawa [Nak79], on effectue une découpe de l'image en blocs de taille fixe à définir. Une analyse locale de l'histogramme de chaque bloc est menée afin d'extraire un seuil (cas bimodal) ou deux seuils (cas trimodal). Si cette analyse ne permet pas de dégager des modes, les seuils sont obtenus par interpolation des seuils des blocs voisins. Ainsi on peut effectuer une analyse d'histogramme calculé sur une sous-image résultant du découpage de l'image.

Ces approches, cf. [Sah88], ont l'avantage de la rapidité des traitements ; elles appliquent pour la plupart le même traitement sur toute l'image. Elles nécessitent en tout cas de fixer des paramètres (seuils, nombres de classes souhaitées, paramètres des distributions) traduisant une information *a priori* conditionnant la qualité de la segmentation obtenue.

2.2.2 Méthodes multidimensionnelles

Les approches multidimensionnelles sont couramment utilisées en analyse d'images issues de la télédétection [Lum83]. Elle permettent de traiter (i)les images multispectrales (ii) et les images texturées dont la simple analyse du niveau de gris se révèle insuffisante. Nous abordons ici un ensemble d'outils de discrimination/classification qui sont aussi employés dans la vision haut niveau. Nous nous concentrerons sur les approches de discrimination statistique. Les approches neuronales de classification seront abordées dans la section portant sur l'interprétation.

Dans le cas d'images texturées, il devient nécessaire d'analyser chaque pixel à l'aide d'un groupe d'attributs calculés localement, sur une fenêtre centrée sur le pixel. On parlera alors de *texel* (abréviation de texture élément), composé des coordonnées du pixel, de la taille de la fenêtre d'analyse et du vecteur d'attributs. Ces attributs peuvent être de simples moments statistiques comme la moyenne ou la variance. Ils peuvent aussi prendre en compte l'aspect anisotropique d'une texture (matrices de cooccurrence) et même l'aspect contour si on leur ajoute les sorties de filtres détecteurs de contours (orientés ou non). Il devient alors possible de prendre en compte simultanément l'aspect contour et région et donc de mettre en œuvre

une coopération simple région/contour. La construction d'un outil de classification comporte au moins les quatre étapes suivantes :

1. calcul des attributs des texels ;
2. sélection des attributs discriminants ; cette étape est abordée ci-dessous ;
3. construction du classifieur (phase d'apprentissage) ; cette phase est présentée en annexe (§A p. 249) ;
4. utilisation du classifieur sur l'ensemble des texels (phase d'exploitation) ;

La sélection des attributs discriminants par l'analyse en composantes principales. Cette sélection peut être faite par le traiteur d'images ou à l'aide de méthodes comme l'analyse en composantes principales (ACP Analyse en Composantes Principales ou transformée Karhunen-Loeve). Nous verrons ultérieurement l'usage de cette méthode (§9.6.3 p. 187) afin de permettre aux agents d'ajuster localement et automatiquement les coefficients de pondération portant sur les attributs région. À cet effet, nous allons donner les principaux concepts de l'ACP. Une étude approfondie peut être trouvée dans [Leb95].

On considère un nuage de n individus (les texels) dans l'espace des p attributs ($\mathbb{R}^p$). On cherche à déterminer un sous-espace vectoriel $\mathbb{R}^q \subset \mathbb{R}^p$ de dimension faible ($q << p$) qui lorsque l'on y projette les individus, déforme le moins possible les distances entre individus projetés, ou autrement dit maximise l'inertie du nuage projeté.

$$X = \overbrace{\begin{bmatrix} x_{11} & \dots & x_{1p} \\ \vdots & x_{ij} & \vdots \\ x_{n1} & \dots & x_{np} \end{bmatrix}}^{les\ p\ attributs} \ les\ n\ individus$$

Dans l'analyse générale, on cherche tout d'abord un sous espace vectoriel à une dimension passant par l'origine qui maximise l'inertie du nuage projeté et, donc, minimise, au sens des moindres carrés, la distance entre l'individu et sa projection sur la droite :

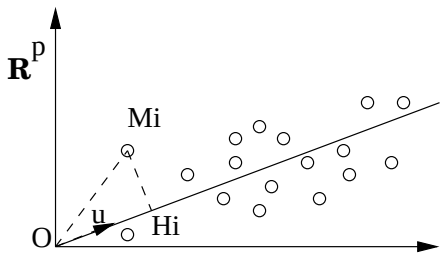


Fig. 2.1 : *Axe de projection maximisant l'inertie d'un nuage de point : cas général.*

$$\min_u \sum_{i=1}^{n} M_i H_i^2 \iff \max_u \sum_{i=1}^{n} OH_i^2$$

or

$$\sum_{i=1}^{n} OH_i^2 = (Xu)^T Xu = u^T X^T Xu$$

$X^T X$ étant la matrice de variance/covariance. On cherche donc :

$$\begin{cases} \max_u u^T X^T Xu \\ \text{sous la contrainte de normalisation : } u^T u = 1 \end{cases}$$

On minimise le lagrangien :

$$\mathcal{L} = u^T X^T Xu - \lambda(u^T u - 1)$$
$$\frac{\delta \mathcal{L}}{\delta u} = 2X^T Xu - 2\lambda u$$

En annulant la dérivée du lagrangien, on obtient :

$$X^T Xu = \lambda u$$

u est donc le vecteur propre associé à la valeur propre λ, comme

$$u^T X^T Xu = \lambda$$

Il faut donc que λ soit la plus grande possible. Il faut que les axes de projection choisis soient les vecteurs propres u_i associés aux plus grandes valeurs propres de la matrice $X^T X$ (matrice de variance/covariance).

Dans l'analyse en composantes principales, les axes ne passent pas forcément par l'origine. On va donc effectuer un recentrage de nos données pour se ramener au cas général.

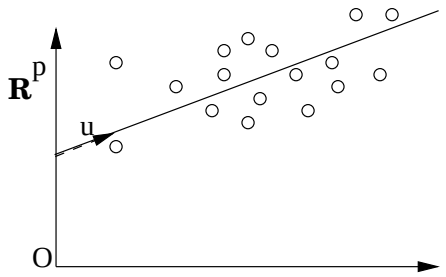


Fig. 2.2 : *Axe de projection maximisant l'inertie d'un nuage de point.*

Ainsi traitée, notre matrice de variance/covariance risque d'être mal conditionnée. En effet, les attributs ayant une grande échelle porteront les axes factoriels principaux. À cet effet, on réduit notre matrice en divisant les valeurs des attributs par l'écart type pour se ramener ainsi à une valeur sans échelle. La matrice des données X sera remplacée par une matrice de données centrée et réduite Y de terme

général :

$$y_{ij} = \frac{x_{ij} - \overline{x_j}}{S_j\sqrt{n}}$$
$$\overline{x_j} = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$$
$$S_j^2 = \frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \overline{x_j})^2$$

Les q axes factoriels principaux (ceux associés aux plus grandes valeurs propres) de $R_{XX} = Y^T Y$ (matrice de corrélation) forment une base orthogonale du sous-espace vectoriel portant l'essentiel de l'inertie du nuage des individus. L'analyse se fait donc dans un sous-espace de taille réduite (ex : $q = 2$), faisant intervenir seulement les attributs les plus discriminants.

2.3 Approches frontière

Nous allons présenter les différents traitements de l'information contour que nous utilisons pour générer les "agents contour" (§8.5.2 p. 168).

Nous présentons en annexe A.3 un traitement complémentaire (fermeture de contour) permettant de finaliser une segmentation par approche contour.

Les approches contour ont pour objectif de détecter les discontinuités en niveaux de gris, couleurs ou textures dans l'image. On procède généralement en quatre étapes :

1. application d'un opérateur de détection de contours (§2.3.1) ;
2. localisation des contours, devant correspondre aux frontières entre régions homogènes de l'image (§2.3.2 p. 23) ;
3. suivi des points de contour pour extraire les chaînes de points qui constituent les frontières (§2.3.3 p. 24) ;
4. fermeture des contours afin d'obtenir les régions définissant ainsi une partition de l'image (§A.3 p. 255).

2.3.1 Les opérateurs de détection de contours

Les approches dérivatives procèdent par une détection des variations de luminance dans l'image. Une première technique consiste à appliquer des filtres à réponse impulsionnelle finie (RIF) qui calculent une approximation de la dérivée première ou seconde du signal bidimensionnel associé à l'image.

Les opérateurs de Prewitt & Sobel sont des masques de convolution qui détectent un gradient dans une direction donnée. On peut de cette façon extraire la norme du gradient et sa direction.

L'opérateur de Marr-Hilbert (approche dérivative de second ordre) se présente sous la forme d'un masque défini selon le laplacien d'une gaussienne et dont les passages à zéro traduisent une discontinuité maximum.

Ces opérateurs sont des filtres passe-haut qui ont tendance à amplifier le bruit. Cette sensibilité au bruit, encore plus forte pour les opérateurs de second ordre (d'où la gaussienne), peut être diminuée en augmentant la taille du masque, permettant par là même un accroissement du rapport signal à bruit. Cette approche a l'inconvénient de multiplier les réponses correspondant à un contour donné, ce qui nuit à la bonne localisation des contours.

L'approche de Canny [Can83] cherche un filtre optimal pour détecter un échelon noyé dans un bruit blanc selon trois critères :

1. une bonne détection qui consiste à avoir le meilleur rapport signal à bruit ;
2. une bonne localisation du contour qui suppose que la distance entre la réponse maximale du filtre et le contour réel doit être faible ;
3. la faible multiplicité des réponses pour un seul contour.

Canny propose une solution à base de filtres RIF [Can86] combinant dérivée de gaussienne comme réponse du filtre et laplacien de gaussienne pour la bonne localisation du contour. Le résultat est seuillé par hystérésis puis étudié à différentes échelles (on modifie la variance de la gaussienne).

Les approches paramétriques : on se dote d'un polynôme modélisant une surface plane ou non ; puis on cherche les paramètres du polynôme approximant au mieux "la surface de l'image". Ce procédé est répété pour chaque pixel en prenant en compte pour l'approximation un voisinage plus ou moins étendu et un polynôme de degré choisi. Haralick propose le "facet model" [Har81] dans lequel chaque facette représente un polynôme d'approximation. On doit ensuite comparer chaque polynôme aux polynômes modélisant les contours afin de juger de la présence d'un contour.

Cette approche autorise une description plus fine des contours, permettant même une étude sub-pixelique grâce à l'introduction d'une modélisation analytique du contour.

2.3.2 Localisation des contours

Certains détecteurs de contour (approche optimale) ont déjà pris en compte le problème de la bonne localisation des contours. Mais les techniques de simple filtrage de type Sobel fournissent une image de gradient qu'il convient de traiter afin de ne conserver que les pixels de contour correspondant à une vraie frontière dans l'image.

Pour certains problèmes, un simple seuillage peut souvent permettre d'extraire les points de contour significatifs si l'on dispose de suffisamment d'informations *a priori*. Le choix d'un tel seuil est spécifique au type d'image traitée : un seuil trop bas laissera trop de points de contour (contours épais et faux contours dus au bruit) tandis qu'un seuil trop haut éliminera des points de contours de façon précoce.

Un tel seuil global est donc souvent difficile voire impossible à déterminer. Une approche locale et adaptative peut souvent donner de bons résultats si l'on apporte de nouvelles contraintes (informations *a priori*) à la sélection d'un point de contour.

Les contraintes traduisent le fait qu'un point de contour significatif fait partie d'une ligne de crête c'est-à-dire :

1. la réponse du filtre (le gradient) en ce point doit être forte ;
2. le gradient le long de la ligne reste fort, autrement dit le gradient des sites perpendiculaires à la direction du gradient est fort ;
3. le gradient s'affaiblit à droite et à gauche de la ligne de crête, autrement dit le gradient des sites placés dans la direction (devant, derrière) du gradient est plus faible qu'au site courant.

De ces contraintes supplémentaires, a été extrait un ensemble d'algorithmes permettant de localiser les points de frontière. Citons l'algorithme de suppression des non-maximums locaux [Son99] qui supprime tous les points dont la norme du gradient n'est pas supérieure à celle des deux points situés dans la direction du gradient.

Un seuillage par hystérésis procède dans une première étape à un marquage de tous les sites dont la norme du gradient est supérieure à un seuil t_1. Ce seuil doit être suffisamment élevé pour éliminer les contours dûs au bruit. Dans une seconde étape, on analyse tous les sites dont la norme du gradient est supérieure à un seuil t_0 (avec $t_0 < t_1$) et qui ont un voisin marqué. On itère cette deuxième étape tant qu'il y a au moins un nouveau site marqué.

2.3.3 Suivi de contours

Les points de contour significatifs étant localisés, il s'agit maintenant d'extraire les chaînes de points qui constituent les frontières entre les régions de l'image. Il existe un vaste échantillon d'algorithmes traitant ce problème, ayant en commun une approche locale sur le choix du prochain point de contour à ajouter à une chaîne. Ils sont plus ou moins bien adaptés à une bonne prise en compte de l'information *a priori* qui conditionne la qualité du résultat obtenu. Prenons par exemple [Son95] le cas de l'extraction de veines (ou artères) qui sont des structures fines composées de deux chaînes de points de contour parallèles ; cette information *a priori* peut être intégrée comme contrainte supplémentaire dans une procédure de construction simultanée des deux chaînes (de points de contour) parallèles. Cette méthode est ainsi plus robuste face à un artefact local sur une des deux chaînes.

Le suivi de ligne de crête par approche classique [Pit93] cherche à fabriquer une chaîne de points de contour à partir des sites où le gradient est maximum. Afin d'éviter les contours dûs au bruit, l'image des gradients est seuillée ; une alternative consiste à rajouter des contraintes aux seuils portant sur les normes de gradient dans les procédures expliquées ci-dessous. Cette construction incrémentale s'appuie sur les critères énoncés à la section 2.3.2.

Soient $|g_i|, \phi_i$ la norme et la direction du gradient au site i qui est le site à l'extrémité de la chaîne courante en construction.

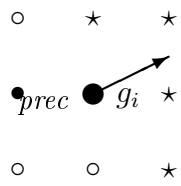


FIG. 2.3 : *Les deux sous-voisinages : $V_i^\star$ et V_i° du site courant (au centre) en fonction du gradient g_i en ce site.*

On partitionne le voisinage de site i en deux sous-voisinages $V_i^\star$ et V_i° en fonction de ϕ_i et du site précédent $\bullet_{prec}$ de la chaîne de contour. On choisit un site j parmi $V_i^\star$, le site prolongeant au mieux le contour c'est-à-dire celui qui optimise les critères suivants :

1. $j = \arg\min |\phi_i - \phi_j| \mod 2\pi$
2. $j = \arg\max |g_j|$

Les deux critères donnent des résultats similaires, on pourra par exemple choisir le deuxième critère dans un voisinage où l'on aura retiré les sites dont la direction du gradient varie trop, c'est-à-dire :

$$j = \arg\max |g_j| \text{ tel que } j \in V_i^\star - \{k \in V_i^\star \text{ tel que } |\phi_i - \phi_k| \mod 2\pi > T\}$$

2.3.4 Conclusion sur les approches contour

Ces approches basées sur la détection de ruptures de stationnarités ne peuvent supporter à elles seules un système robuste de segmentation, ceci pour plusieurs raisons :

1. Elles ne sont pas adaptées aux images texturées, employées seules elles fourniraient un résultat sur-segmenté.
2. Elles sont sensibles au bruit de par leur nature "passe-haut", ce qui peut provoquer la création de faux contours.
3. La fermeture de contour est une tâche ardue, car de nombreuses erreurs sur lesquelles il est difficile de revenir, peuvent être commises ; la fusion de deux contours distincts en est un exemple. Une certaine robustesse peut être obtenue avec l'ajout d'un grand nombre d'informations *a priori* apportant des contraintes supplémentaires afin d'éviter les erreurs. Par exemple, en segmentation d'imagerie aérienne, l'extraction des routes est facilitée par la structure parallèle des deux bords de la route.

Les approches contour apportent une information significative sur la qualité des ruptures tant en intensité (longueur et gradient le long d'une chaîne de contours) qu'en localisation, information qui doit être intégrée à travers un processus de segmentation par approche région, et qui peut même servir en interprétation sur la caractérisation de la forme d'une région.

2.4 Approches structurales ou agrégatives

Dans ces approches, les traitements sont transposés dans un univers s'appuyant sur des **structures de données**. On utilisera par exemple un arbre quaternaire "quadtree" (§3.3.1 p. 35) ou bien un graphe d'adjacence de région (RAG) (§3.4 p. 37), ce dernier pouvant même être utilisé dans une structure hiérarchique (pile de graphes) [Mon91].

Les méthodes structurales [Har85, Ada94, Cha94] sont aussi appelées méthodes **agrégatives** ou **incrémentales** [Bel98] car elles cherchent à constituer de façon itérative (par agrégation ou division) une partition de l'image en régions homogènes R_i respectant un critère d'homogénéité ($H(R_i) = VRAI$) à partir d'un ensemble de régions initiales.

Elles fournissent un **cadre décisionnel riche** (prise en compte aisée de l'information *a priori*) et local. Ce dernier point autorise une parallélisation des traitements permettant de contrebalancer l'important coût calculatoire propre à ces approches.

On distingue deux familles de méthodes :

1. les approches descendantes : découpe récursive (split) ;
2. les approches ascendantes de type accroissement de régions (région *growing*).

Les approches varient suivant le critère d'homogénéité adopté. Les approches ascendantes appliquent ce prédicat sur l'union des régions candidates à la fusion. Les approches descendantes, quant à elles, l'appliquent à une région qui sera divisée si le prédicat n'est pas respecté. Le prédicat d'homogénéité $H()$ seuille le résultat d'une fonction, évaluant l'homogénéité $h()$ d'une région ou la similarité de deux régions :

$$H(R_p \cup R_q) = \begin{cases} \text{VRAI} & \textbf{si } h(R_p \cup R_q) \leq T \\ \text{FAUX} & \textbf{sinon} \end{cases}$$

Voici les critères les plus couramment employés :

1. Une simple contrainte sur la différence entre la valeur maximum et minimum des niveaux de gris dans la région :
$$h_{trivial}(R_p \cup R_q) = |\max_{x_p \in R_p \cup R_q}(x_p) - \min_{x_q \in R_p \cup R_q}(x_q)|$$
2. Une mesure de similarité entre les moyennes des niveaux de gris de deux régions :
$$h_{moy}(R_p \cup R_q) = |moy(R_p) - moy(R_q)|$$
3. Une mesure de similarité multidimensionnelle portant sur des attributs caractérisant les textures respectives des deux régions comme par exemple les matrices de cooccurrence [Che78].
4. Une mesure du contraste entre les deux régions. Soit F_{pq} les pixels de R_p ayant dans leur 4-voisinage $V_4(x_i)$ un voisin appartenant à R_q. Le contraste mesure les variations en niveau de gris le long de cette frontière (de longueur $L(F_{pq})$)
$$h_{cont}(R_p \cup R_q) = \frac{1}{L(F_{pq})} \sum_{x_i \in F_{pq}} \sum_{x_j \in V_4(x_i)} |x_i - x_j|$$

5. Les critères peuvent aussi porter sur des aspects géométriques. L'union de deux régions doit, par exemple, être suffisamment compacte : calcul du facteur de circularité (périmètre/surface).

2.4.1 Approches descendantes par division récursive

Cette procédure récursive proposée par Horowitz et Pavlidis [Hor74] s'appuie sur un arbre quaternaire "quadtree" (§3.3.1 p. 35). Elle prend en paramètre une image qui est divisée en quatre sous-images si un critère d'homogénéité (prédicat) n'est pas respecté. La procédure est appelée avec une fenêtre de la taille de l'image à segmenter. Le résultat est le noeud racine de l'arbre.

SPLIT(IMAGE, FENÊTRE, NOEUD)
si (H(fenêtre, image) **ou** taille de la fenêtre $= 1$) noeud.fenêtre $\leftarrow$ fenêtre
sinon
– noeud.NO$\leftarrow$SPLIT(image,**new** fenêtre(i_1,$(i_1+i_2)/2$,j_1,$(i_1+i_2)/2$),**new** noeud)
– noeud.NE$\leftarrow$SPLIT(image,**new** fenêtre(i_1,$(i_1+i_2)/2$,$(i_1+i_2)/2$,j_2),**new** noeud)
– noeud.SO$\leftarrow$SPLIT(image,**new** fenêtre($(i_1+i_2)/2$,i_2,j_1,$(i_1+i_2)/2$),**new** noeud)
– noeud.SE$\leftarrow$SPLIT(image,**new** fenêtre($(i_1+i_2)/2$,i_2,$(i_1+i_2)/2$,j_2),**new** noeud)
fin si
retourne noeud

ALGO. 2.1 : *Segmentation descendante par découpe récursive sur une structure de quadtree. Les structures de données "fenêtre" et "noeud" sont présentées (§3.3.1 p. 35).*

Cette approche "top down" est plus adaptée que les approches ascendantes aux textures "gros grain" car, avec un prédicat adapté, la division s'arrête à l'échelle de la texture sans provoquer de sur-segmentation. Ceci est obtenu grâce à une estimation des statistiques effectuée sur des zones étendues dès le départ.

Cependant, on constate un effet de "pavage" des segmentations obtenues (voir chap. 10). Cet effet est causé par un seuil unique appliqué à toute l'image, ce dernier devant être suffisamment élevé pour éviter en certains endroits une sur-segmentation, mais qui sur d'autres zones de l'image provoque une sous-segmentation.

De plus, on peut obtenir deux régions connexes R_p, R_q pour lesquelles le prédicat est vérifié $H(R_p \cup R_q) = VRAI$. Cette situation peut se produire si les deux régions ne sont pas dans le même sous-arbre de découpe récursive. D'où l'intérêt d'adjoindre un algorithme de fusion de régions.

2.4.2 Approches ascendantes par fusion de régions "region merging"

Cette approche ascendante va itérativement fusionner les régions dont l'union respecte le prédicat d'homogénéité jusqu'à ce que plus aucune fusion ne soit possible.

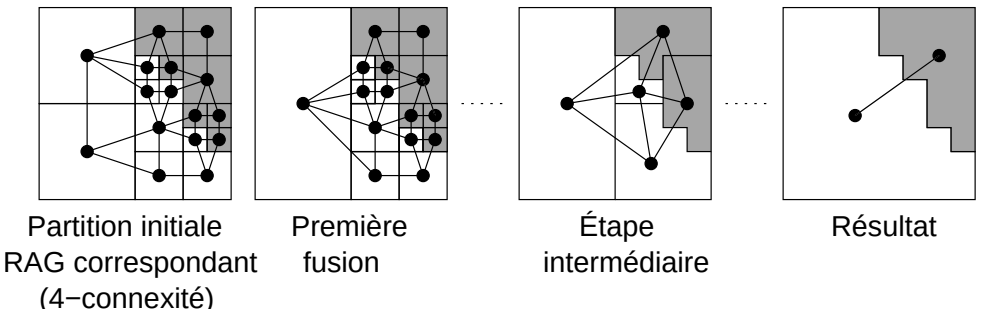


FIG. 2.4 : *Graphe d'adjacence de région (RAG) associé à une partition initiale (obtenue à l'aide d'un "split") puis application de la procédure de fusion.*

Le travail s'effectue sur un graphe d'adjacence de régions (RAG : Region Adjancy Graph) (§3.4 p. 37). Outre le choix du prédicat d'homogénéité, le point clé de cet algorithme est l'ordre dans lequel les fusions sont effectuées. Il est donc nécessaire (comme le précise Bellet [Bel98]) de trier les régions candidates à la fusion sur la valeur de la fonction d'homogénéité afin d'effectuer en premier les meilleures fusions.

REGION MERGING
$\{R_1, R_2, ..., R_n\}$ $\leftarrow$ effectuer une première (sur)segmentation (ex. : SPLIT)
$G(S, A) \leftarrow$ construire le graphe d'adjacence
pour tout(les couples (s_p, s_q) tels que $\exists (p, q) \in A$)
– $h_{pq} \leftarrow h(R_p \cup R_q)$ // *fonction d'homogénéité*
– **si** $(h_{pq} \leq T)$ insérer (s_p, s_q, h_{pq}) dans `liste` par ordre de h_{pq} croissant
fin pour tout
tant que(`liste` n'est pas vide)
– fusionner le meilleur couple : $(s_p, s_q, h_{pq}) \leftarrow$ pop(`liste`) ; $s_n = s_p \cup s_q$
– mettre à jour le graphe d'adjacence $G(S, A)$ prenant en compte le retrait de s_p, s_q et l'ajout de s_n
– calculer les attributs du noeud s_n résultant de la fusion des sommets s_p, s_q
– retirer les couples de `liste` où l'une des 2 régions s_p ou s_q intervient
– **pour tout**(les couples (s_n, s_q) tels que $\exists (n, q) \in A$)
  – $h_{nq} \leftarrow h(R_n \cup R_q)$
  – **si** $(h_{nq} \leq T)$ insérer (s_n, s_q, h_{nq}) dans `liste` par ordre de h_{nq} croissant
  fin pour tout
fin tant que

ALGO. 2.2 : *Segmentation ascendante par fusion de régions sur une structure de graphe d'adjacence.*

Une combinaison de divisions et de fusions (algorithme "split and merge"[Hor74, Pav77]) permet de profiter des avantages des deux approches.

2.4.3 Approches ascendantes par agrégation de pixels

On fait croître des régions pixel par pixel à partir d'un ensemble de germes déposés dans l'image. La procédure de croissance autour des germes déposés (régions

initiales) est similaire à celle de la fusion de régions à ceci près que l'entité de base agrégée est le pixel.

2.4.4 Conclusion sur les approches structurales

Jiang et Toriwaki [Jia93] proposent une analyse comparative des différents opérateurs. On peut dégager deux tendances générales dépendant des méthodes d'agrégation :

- En terme de précision, on choisira plutôt les opérateurs d'agrégation de pixels naturellement plus précis que les opérateurs de fusion de régions dont l'élément primitif d'agrégation est une fenêtre carrée.
- En terme de sensibilité au bruit, la tendance s'inverse en faveur des fusions de régions dont les attributs sont calculés sur un plus grand nombre d'échantillons permettant ainsi un lissage du bruit.

L'évaluation doit ensuite être différenciée suivant le prédicat de fusion choisi. En terme de facilité de paramétrage et de rapidité d'exécution, les approches utilisant des critères simples de type $h_{trivial}$ donnent de meilleurs résultats que les approches multidimensionnelles ou utilisant des heuristiques. Les approches simples (monodimensionnel à un seuil global) produisent des images sur-segmentées sur des images texturées ou même sur des images où la distribution globale des niveaux de gris résulte d'une fusion de plusieurs distributions locales empêchant la détermination d'un seuil global [Bev89].

Ces arguments jouent en faveur d'approches hybrides adaptant localement et temporellement (temps d'exécution de l'algorithme de fusion) le critère de fusion. Chang et Li [Cha94] proposent un test d'homogénéité adapté à :

1. la distribution des niveaux de gris des régions locales, en faisant varier le seuil autorisant la fusion ;
2. la taille des régions considérées, permettant ainsi la prise en compte du nombre d'échantillons utilisés pour l'estimation des paramètres.

Beveridge et al. [Bev89] proposent de découper l'image en "secteurs" permettant une classification sur la base d'histogrammes calculés localement. Un algorithme de fusion de régions est ensuite appliqué pour fusionner les régions connexes ne faisant pas partie du même secteur.

2.5 Approches coopératives

Les travaux comparatifs [Coc95, Jia93] sur les différentes techniques de segmentation n'ont pas dégagé une prééminence (indépendamment du type d'image) d'une approche particulière. La tendance actuelle pour espérer une certaine robustesse et des qualités "tout terrain" consiste donc à faire coopérer différents types d'algorithmes. La richesse de la littérature sur les procédures coopératives n'est pas en adéquation avec l'apport qu'on pourrait en espérer. Ce manque est peut être dû selon Pavlidis et Liow [Pav90] à la difficulté du problème :

> *"The general principle of integration is well accepted in vision. But carrying out the general principle is quite challenging and this may explain the sparsity of the literature on integrated techniques."*

Ou selon Pavlidis [Pav92] à la manière dont les chercheurs du domaine abordent le problème :

> *"The main obstacle to integration of methodologies is that various approaches are offered as "panaceas" rather than as member of a toolset."*

La plupart des approches coopératives exploitent la dualité contour/région. Cette dualité en traduit souvent une autre : la dualité local/global [Cho97]. En effet, les approches contour sont souvent locales, donc plus sensibles au bruit, mais permettent une bonne localisation des ruptures. Au contraire, les approches s'appuyant sur l'homogénéité entre régions sont plus globales (de part la définition de leurs seuils) et robustes au bruit, mais elles aboutissent à une mauvaise localisation des frontières (on obtient souvent des frontières déchiquetées, voir chapitre 10). On distingue plusieurs approches coopératives, décrites ci-après, exploitant la dualité contour/région.

2.5.1 Coopération confrontative par fusion de données

Dans cette première classe de méthodes la coopération intervient comme un consensus ou une confrontation *a posteriori* entre plusieurs segmentations concurrentes.

Chu et Aggarwal [Chu93] suggèrent l'intégration *a posteriori* de cartes de segmentation calculées en parallèle. Ces cartes de contours sont issues de l'application de détecteurs de contours et de l'extraction des contours après une segmentation par approche région. Ils utilisent, de plus, plusieurs sources de la même scène. L'intégration finale cherche à générer un consensus entre les différentes segmentations sur la base d'une pondération, réglable par l'utilisateur, apportée à chacune des cartes.

Cho et Meer [Cho97] proposent une approche basée sur le consensus entre plusieurs segmentations. Les résultats des diverses segmentations permettent la construction d'un graphe (RAG) reflétant pour chaque paire de pixels la "probabilité de cooccurrence", autrement dit, la probabilité d'appartenance à la même région. Le consensus est obtenu en traitant ce RAG de façon classique afin de regrouper les pixels ayant une forte probabilité d'appartenir à la même région.

2.5.2 Coopération intégrative

Dans cette deuxième classe de méthodes, l'information contour est intégrée sous forme de contrainte par une approche région. La coopération n'est pas complètement dynamique en ce sens que l'information de contour est extraite de façon statique et ne profite pas de l'information région.

Pavlidis et Liow [Pav90] proposent un traitement en trois étapes :

1. l'image est sur-segmentée par une approche de division-fusion sur arbre quaternaire permettant ainsi de dégager une carte de contours fermés à partir des régions ;
2. les frontières entre régions connexes sont éliminées sur des critères de contraste et de changement de direction ;
3. ces frontières sont ensuite déplacées afin d'être mieux localisées, ceci à l'aide d'une approche par contours actifs travaillant sur une image de contours.

Wrobel et Monga [Wro87] effectuent une pré-segmentation afin d'obtenir un RAG initial dont les noeuds portent les attributs des régions et dont les arêtes sont pondérées par une carte des contours pré-calculée. Cette pondération compte le nombre de points de la frontière où le contraste est supérieur à un seuil et la longueur de cette frontière. Les fusions de deux régions dont le rapport (longueur de frontière)/(nombre de points de contraste) est inférieur à un seuil sont empêchées. On obtient ainsi une coopération par contrainte.

Anderson et al. [And87] utilisent une détection de contours pour initialiser et assister une procédure de croissance de régions basée sur le calcul d'un seuil de similarité locale. Ce dernier est aussi influencé par des informations *a priori* comme le nombre de régions dans l'image.

Xiaohan et al. [Xia92] utilisent une information de gradient local pour contrôler un processus de croissance de région.

Bertolino [Ber96, Ber95] propose une approche basée sur une pyramide de RAG (pyramide irrégulière) où de l'information fournie par une carte de contour (obtenue par un détecteur 1D spécifique) est prise en compte lors de l'évaluation des fusions entre régions voisines. L'information de contour se présente comme un couple (r, c) qui pondère les arêtes du graphe d'adjacence ; r et c représentant respectivement le nombre d'éléments de région (faible réponse au détecteur de contour) et le nombre d'éléments de contour (forte réponse) à la frontière des deux régions. Une analyse locale de r et c permet de forcer ou d'interdire des fusions entre sommets adjacents du RAG afin d'éviter la création d'artefacts ou la disparition de détails significatifs.

2.5.3 Coopération dynamique par influence mutuelle

Dans cette dernière classe de méthodes, les différentes approches s'influencent mutuellement et dynamiquement au cours du processus de segmentation. C'est une voie qui semble prometteuse car l'information extraite par une méthode est utilisée au plus tôt par les autres approches. La difficulté rencontrée est le problème du contrôle des modules et de la diffusion de l'information en leur sein. C'est donc naturellement que les chercheurs se sont portés vers des approches utilisant l'intelligence artificielle. L'IA propose une panoplie d'architectures et d'outils (tableaux noirs,

agents, systèmes experts) permettant une intégration distribuée d'une connaissance complexe et le contrôle entre les différents modules de connaissance.

Salotti, Bellet et al. [Bel94, Bel98] (§5.5.3.1 p. 100) proposent une approche à base de processus situés dans l'image. Certains de ces processus exécutent une croissance de région tandis que les autres exécutent un suivi de contour ; ils coopèrent de façon opportuniste et dynamique pour faire émerger de l'information supplémentaire au cas où ils ne peuvent décider quel nouveau pixel peut être agrégé à la primitive en construction.

Nazif et Levine [Naz84] proposent un système expert comportant des règles permettant l'analyse de primitives région et contour. Ces règles agissent sur les primitives en les divisant, fusionnant suivant les situations locales respectives des primitives en condition dans les règles. Ceci autorise la prise en compte de primitives contour dans l'action à mener sur des primitives région, et réciproquement.

Chapitre

3

Les Structures pyramidales

Une architecture de contrôle à base de système multi-agents ne peut être globalement analysée que si l'on a procédé à une spécification rigoureuse de l'organisation structurant la population d'agents.

Nous proposons d'utiliser la pyramide irrégulière comme élément structurant et régulant l'activité de ces derniers.

Nous allons lors de ce chapitre présenter les structures pyramidales en analyse d'image. Nous évoquerons d'abord les aspects multirésolution, puis nous détaillerons plus particulièrement les pyramides irrégulières, pour finir par évoquer les pyramides duales.

Certaines caractéristiques des structures pyramidales, comme le parallélisme, en font un modèle organisationnel particulièrement adapté à notre problématique.

3.1 Introduction

Les structures pyramidales ont été introduites par Tanimoto et Pavlidis [Tan75]. Il s'agit d'un empilement de structures de données représentant à chaque niveau l'image (ou une bande de fréquences de l'image) observée à une certaine résolution. On se dote d'une procédure de décimation (généralement récursive) pour passer du niveau k à $k+1$. En partant de la base qui représente l'image (ou l'image filtrée), on aboutit progressivement, par simplification de l'information, à l'apex, le dernier niveau de la pyramide, comportant un minimum d'informations quantitatives. Soulignons deux des principaux aspects liés à ce type d'approche :

Le parallélisme, inhérent à la structure pyramidale, permet une implantation naturelle sur des machines parallèles de type multigrilles aussi appelées machines pyramidales. La localité des calculs permet avec un nombre de N^2 processeurs[1] d'obtenir une complexité en temps de $O(\log N)$. Thomson Leighton propose une introduction générale aux architectures et machines parallèles dans [TL92], tandis que Mérigot dans [Jol01] chap. 6 traite plus particulièrement de la vision parallèle. On trouvera dans [Mér86] un exemple d'architecture pyramidale pour le traitement d'image.

La multirésolution, obtenue par la simplification niveau par niveau de la quantité d'informations, permet de choisir la résolution d'analyse la plus adaptée au problème. À basse résolution, l'atténuation du bruit et les détails non significatifs permettent un traitement robuste et rapide. Les résultats ([Pav77, p.74] obtenus à basse résolution peuvent être utilisés [Son95] comme informations *a priori* ou contraintes supplémentaires pour guider un traitement effectué à plus forte résolution dans un environnement bruité.

Toutefois, cet aspect comporte deux inconvénients :

- la simplification de l'information s'accompagne de la disparition de frontières significatives entre régions ;
- cette simplification peut aussi générer et amplifier, au fur et à mesure de la croissance, l'apparition de frontières qui n'existent pas ou sont peu accentuées dans l'image originale.

3.2 Aspect multirésolution : pyramides Gaussienne et Laplacienne

La pyramide Gaussienne [Bur81] est construite suivant une approche "bottom up" où chaque niveau G_{k+1} est une copie basse résolution (filtrage passe-bas puis décimation) du niveau inférieur G_k. Ce type de représentation présente les avantages évoqués ci-dessus : analyse rapide à basse résolution dont les résultats orientent ou/et accélèrent les traitements aux plus fortes résolutions.

[1] N^2 étant la taille de l'image

Dans [Li97] une segmentation à base de champs de Markov est effectuée à basse résolution (niveau G_n). L'information collectée (champs d'étiquettes) est propagée vers le niveau inférieur G_{n-1} comme information *a priori* , permettant une accélération de la segmentation à ce niveau. La procédure de propagation puis de segmentation est appliquée successivement à chaque niveau, de manière descendante, jusqu'à l'obtention d'une segmentation de l'image source.

Ce type de transformation pyramidale est aussi un outil utilisé dans le domaine du codage (compression) où le souci est d'exploiter les redondances dans l'image aussi bien localement que sur de grandes étendues. C'est l'objectif de la pyramide Laplacienne [Bur83] qui ne comporte que les erreurs de prédictions propagées du haut vers le bas.

Après obtention de la pyramide Gaussienne $\{G_0, ..., G_k, ..., G_n\}$, on estime chaque niveau G_k à partir du niveau supérieur G_{k+1} par un sur-échantillonnage et une interpolation (une expansion de l'image). L'erreur entre l'estimation $EXPAND(G_{k+1})$ et G_k représente un niveau de la pyramide Laplacienne :

$$L_k = G_k - EXPAND(G_{k+1})$$

Finalement on ne transmet que $\{G_n, L_{n-1}, ..., L_0\}$. G_n est une représentation compacte basse fréquence de l'image concentrant l'essentiel de l'énergie, les L_k sont les erreurs d'estimation exploitant les redondances sur de grandes puis de petites étendues. La pyramide Laplacienne contient des informations de type passe-bandes, dont les valeurs sont souvent faibles ou nulles. Ce qui permet, à l'aide d'une compression statistique et/ou d'une quantification, de réduire significativement le débit associé à l'image. Le codage par pyramide Laplacienne augmente la robustesse de l'information en cas de rupture de transmission.

3.3 Modèles de décomposition hiérarchique de l'image

Les pyramides interviennent ici comme un modèle de décomposition hiérarchique de l'image. Ces pyramides servent de squelettes aux approches de segmentation structurales de type découpe récursive (§2.4 p. 26). Elles sont donc construites de façon descendante.

Nous allons maintenant évoquer les différentes techniques de partitionnement géométrique et adaptatif de l'image. Un partitionnement est dit adaptatif si la position de l'élément de partition est fonction du contenu informatif de l'image. Cette adaptation au contenu informatif est guidé par un algorithme de type découpe récursive "split".

3.3.1 Modèle rigide de partitionnement

Dans le *quadtree* [Hor74], l'élément de partitionnement est un carré. La pyramide est récursivement construite par une approche de découpe récursive descendante (§2.4.1 p. 27).

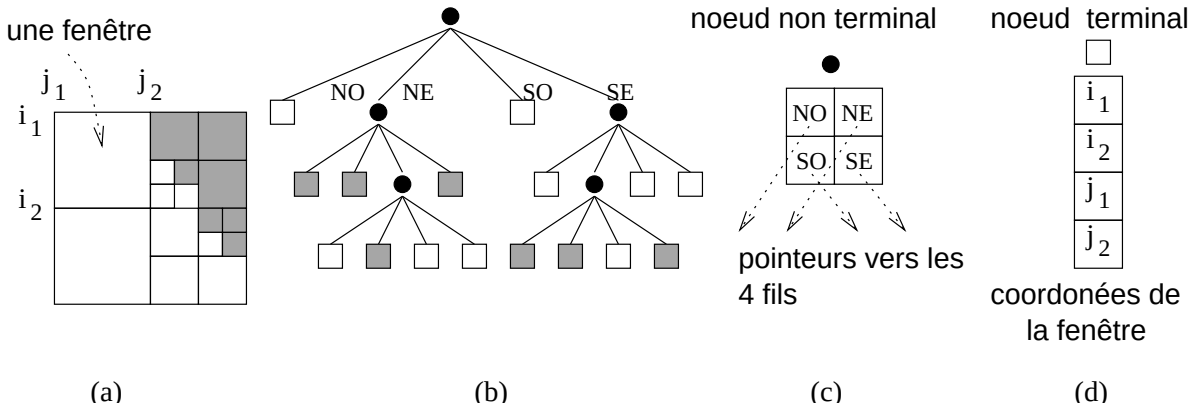


FIG. 3.1 : *(a) une image segmentée ; (b) le résultat de la segmentation sous forme d'arbre quaternaire ; (c) la structure d'un noeud non terminal puis (d) terminal.*

Les partitionnements rigides ont comme inconvénient majeur la création d'artefacts reflétant la géométrie de l'élément de partition (carrés pour les "quadtree"). Afin de limiter ces artefacts, il est nécessaire de placer un grand nombre d'éléments de petite taille.

3.3.2 Partitionnement non rigide

Les partitionnements non rigides permettent d'obtenir une meilleure partition avec moins d'éléments (de plus grande taille).

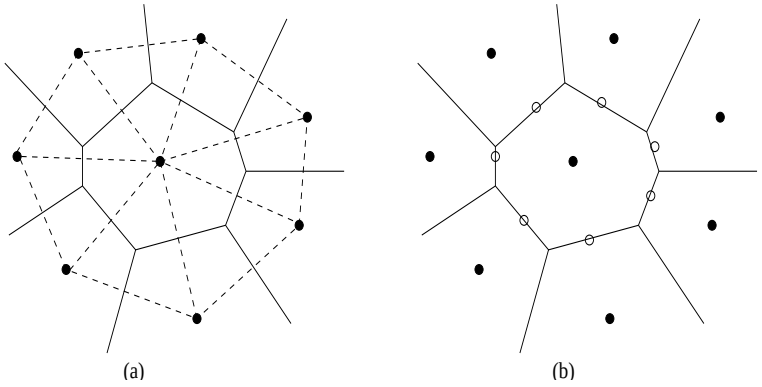


FIG. 3.2 : *(a) Partitionnement de l'image par le diagramme de Voronoï en traits pleins, la triangulation de Delaunay traits en pointillés ; (b) cercles : nouveaux germes pour la prochaine itération de segmentation par Voronoï en cas de non homogénéité.*

Le diagramme de Voronoï partitionne l'image à l'aide d'un ensemble de régions polygonales comportant un nombre de côtés variables. On dispose d'un ensemble de germes répartis dans l'image. Pour chaque germe, il est créé une région composée des points (de l'image) les plus proches du germe, d'où l'aspect de polygone convexe. La

segmentation, par découpe récursive (§2.4.1 p. 27), consiste à débuter avec un ensemble de germes peu nombreux, à calculer le prédicat d'homogénéité dans la région polygonale associée au germe et, si le prédicat n'est pas vérifié, à créer de nouveaux germes qui induiront de nouvelles régions polygonales. Les nouveaux germes peuvent être placés au centre des côtés du polygone.

La triangulation de Delaunay, structure duale du diagramme de Voronoï, partitionne l'image en triangles délimités par les segments joignant les germes.

Bertin propose [Ber94, p. 69] une comparaison entre différentes méthodes de partitionnement de l'espace 3D : Voronoï, Delaunay et Octree[2]. Les résultats, pouvant être étendus à une image 2D, montrent que le diagramme de Voronoï est le plus adapté à l'image car il nécessite beaucoup moins d'éléments que la partition à base d'octree, et sensiblement moins qu'une partition par triangulation de Delaunay. Cependant, le temps de calcul s'en trouve considérablement augmenté ; de plus, la mémorisation des polygones nécessite des structures de données plus lourdes à stocker en mémoire et plus complexes à traiter. Ceci permet de comprendre la popularité des quadtree sachant que les outils de traitement d'images travaillent plus rapidement sur des fenêtres rectangulaires sans masque.

3.4 Introduction aux pyramides irrégulières et graphes d'adjacence de région

Dans un premier temps, nous présenterons dans la section suivante une introduction aux pyramides irrégulières. Puis, dans un deuxième temps, nous préciserons dans le détail la structure essentielle de ces dernières : le graphe d'adjacence de région.

Comme les autres structures pyramidales, la pyramide irrégulière est un empilement de structures de données représentant chacune l'image à une résolution décroissante de la base au sommet. La particularité des pyramides irrégulières tient au fait que les cellules, composant les niveaux, représentent chacune une région dont la forme n'est pas contrainte par un motif géométrique (carré, triangle). En conséquence, le nombre de voisins d'une cellule est variable.

La structure de données composant chaque niveau est le graphe d'adjacence de région (voir ci-dessous). La construction de la pyramide s'effectue de manière ascendante, en appliquant sur le graphe d'adjacence du niveau courant les deux étapes ci-dessous pour obtenir le graphe du niveau suivant.

1. Première étape : décimation de certains sommets du graphe du niveau courant afin d'obtenir un sous-ensemble de sommets qui formeront les sommets du graphe du niveau suivant (§3.5 p. 38).

[2]L'octree est le pendant 3D du quadtree, chaque feuille représente un volume cubique, chaque noeud intermédiaire a 8 voisins.

2. Deuxième étape : rattachement des sommets du niveau courant à un sommet du niveau suivant (§3.6 p. 42).

Le graphe d'adjacence de région [Hor78, Pav77], **RAG** pour **Region Adjancy Graph**, est un graphe non orienté $G(S, A)$ où $S = \{s_1, s_2, ..., s_n\}$ (ensemble des sommets) représente les régions $\{R_1, R_2, ..., R_n\}$ de l'image et $A = \{(s_p, s_q) \in S \times S\}$ est l'ensemble des arêtes.

Deux sommets s_p, s_q sont adjacents ($\exists(s_p, s_q) \in A$) si et seulement si les deux régions associées R_p, R_q sont connexes, autrement dit : $\exists x_i \in R_p, x_j \in R_q$ tels que x_i et x_j sont voisins.

Remarque : en 4-connexité les graphes d'adjacence sont planaires, ce qui signifie que représentés dans un plan, leurs arcs ne se coupent pas. Cette propriété n'est plus vérifiée en 8-connexité.

Informations portées par les sommets : chaque sommet du graphe dispose d'un ensemble d'informations permettant de caractériser la région qu'il représente :

1. Les attributs photométriques permettent caractériser la région représenté par le sommet. Ce sont des attributs de région qui peuvent être de simples moments : moyenne, variance [Jol92]. On peut néanmoins envisager des attributs plus complexes permettant de caractériser une texture, citons : les matrices de cooccurrence, décomposition fréquentielle, loi conditionnelle estimant au mieux la texture au sens de champs de Markov, ...
2. Les attributs géométriques permettent de caractériser la forme de la région. On peut y retrouver la surface, le centre de gravité, le périmètre. Des attributs plus sophistiqués peuvent aussi être envisagés citons, : des moments de forme comme la compacité, la direction principale d'inertie, des descripteurs de Fourier [Pav77, p. 154].
3. Différentes variables utilisées par les algorithmes, citons : les variables p et q de l'algorithme de décimation (§3.5 p. 38) ; on pourra aussi retrouver les listes d'adjacences permettant la mise en œuvre des différents graphes manipulés comme le graphe d'adjacence, de similarité (§3.6.1 p. 42), les relations père/fils entre sommets des différents niveaux.

3.5 Décimation sur un graphe

3.5.1 Introduction

Nous allons aborder la segmentation basée sur une pyramide de graphes d'adjacence. Il s'agit d'appliquer à chaque niveau une contraction du graphe d'adjacence du niveau k : $G^k(S^k, A^k)$ afin de passer au graphe du niveau $k+1$ c'est-à-dire : $G^{k+1}(S^{k+1}, A^{k+1})$. Cette contraction doit être menée de façon distribuée en parallèle sur le graphe d'adjacence suivant deux contraintes :

1. la contraction doit être significative, ce qui nous renvoie à la notion de sous-ensemble stable (voir ci-dessous) ;
2. mais chaque sommet de S^k doit être représenté dans S^{k+1}, ce qui nous renvoie à la notion de sous-ensemble dominant.

Définition 1 (Stable) *$S^{k+1} \subseteq S^k$ est un **stable** de G^k si les sommets de S^{k+1} sont deux à deux non adjacents dans G^k :*

$$\forall s \in S^{k+1}, \Gamma_{G^k}(s) \cap S^{k+1} = \emptyset$$

Où $\Gamma_{G^k}(s)$ est l'ensemble des voisins de s dans le graphe $G^k(S^k, A^k)$ c'est-à-dire :

$$\Gamma_{G^k}(s) = \{r \in S^k : (s, r)\ A^k\}$$

Définition 2 (Stable maximal) *$S^{k+1} \subseteq S^k$ est un **stable maximal** si aucun sommet ne peut lui être rajouté sans perdre la stabilité.*

Définition 3 (Ensemble dominant) *$S^{k+1} \subseteq S^k$ est un **ensemble dominant** de G^k si chaque sommet qui n'appartient pas à S^{k+1} est adjacent à un sommet de S^{k+1}.*

$$\forall s \in S^k \setminus S^{k+1}, \Gamma_{G^k}(s) \cap S^{k+1} \neq \emptyset$$

Définition 4 (Ensemble dominant minimal) *$S^{k+1} \subseteq S^k$ est un **ensemble dominant minimal** de G^k si aucun sommet ne peut lui être enlevé sans rompre la dominance.*

Théorème 1 *Un stable maximal est un ensemble dominant minimal.*

La notion de stable maximal remplit les propriétés de bonne contraction et de représentation évoquées ci-dessus.

Meer, dans son article [Mee89] sur les pyramides irrégulières stochastiques, aborde le problème sous l'angle de la décimation d'échantillons dans un signal, le processus de décimation élit des échantillons (sous-échantillonnage) survivants sur des informations uniquement locales (les états des deux échantillons voisins) en imposant une répartition régulière des échantillons survivants.

3.5.2 Décimation stochastique

Meer propose de généraliser la procédure de décimation de signal à la décimation dans un graphe. Afin d'obtenir une décimation régulière prenant en compte les relations spatiales entre sommets survivants et non-survivants, deux règles sont fixées :

Règle 1 Deux sommets adjacents dans G^k ne peuvent tous deux survivre dans G^{k+1} :

$$\forall (s_i, s_j) \in A^k \Rightarrow \neg(s_i \in S^{k+1} \wedge s_j \in S^{k+1})$$

Cette règle garantit une décimation maximale et donc une diminution significative du nombre de sommets au fur et à mesure de la croissance de la pyramide. Cette règle est souvent appelée contrainte [C1].

Règle 2 Chaque non survivant s_i de G^k doit avoir dans son voisinage $\Gamma_{G^k}(s_i)$ un sommet survivant dans G^{k+1} :

$$\forall s_i \in S^k : s_i \notin S^{k+1} \Rightarrow \Gamma_{G^k}(s_i) \cap S^{k+1} \neq \emptyset$$

Cette règle oblige la décimation à être régulière et permet à chaque sommet d'avoir un voisin survivant auquel se rattacher afin d'être représenté dans le niveau supérieur. Cette règle est souvent appelée contrainte [C2].

Le respect de ces deux règles garantit à S^{k+1} d'être un ensemble stable maximal de G^k. Le choix des survivants se fait de manière stochastique : on alloue à chaque sommet s_i un nombre aléatoire x_i respectant une loi uniforme entre $[0, 1]$. Un sommet survit s'il maximise dans son voisinage cette valeur (règle 1). Cette solution peut produire des configurations dans lesquelles un sommet n'a aucun survivant dans son voisinage (voir Fig. 3.3).

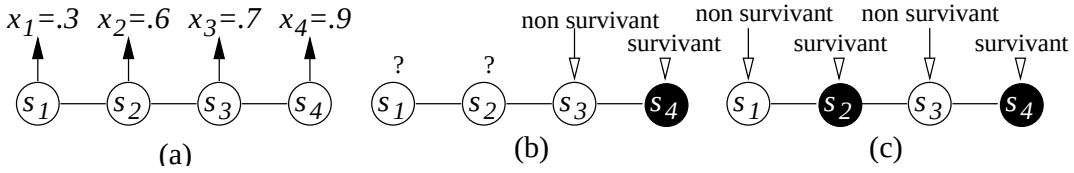


Fig. 3.3 : *Exemple de décimation dans une configuration 1D : (a) les sommets et les valeurs du tirage aléatoire (b) décimation triviale menant à un ensemble stable mais non maximal ; (c) solution itérative menant à un stable maximal.*

La décimation peut être effectuée par une procédure séquentielle qui s'appuie sur une liste triée. On retire et sélectionne le premier sommet de la liste puis l'on retire de la liste tous les sommets voisins (dans le graphe) du sommet selectionné . On itère cette procédure tant qu'il reste des sommets dans la liste. L'ensemble des sommets sélectionnés forme un stable maximal du graphe.

Afin de résoudre ce problème de manière distribuée, Meer adopte un algorithme itératif distribué (illustré fig. 3.4) basé sur deux variables booléennes : p_i^l et q_i^l, où i représente le sommet et l l'itération de l'algorithme.

À l'initialisation p_i et q_i sont positionnés à 0. Si le sommet est un maximum local (équation 3.1) p_i est positionné à 1 et s_i est survivant. Si le sommet n'a aucun survivant dans son voisinage et qu'il n'est pas lui-même survivant alors $q_i = 1$ (équation 3.2 & 3.5), ce qui signifie que le sommet s_i est candidat à devenir survivant à la prochaine itération. Lors des itérations suivantes, le sommet peut devenir survivant ($p_i = 1$) s'il est un maximum local (équation 3.4) parmi ses voisins encore candidats.

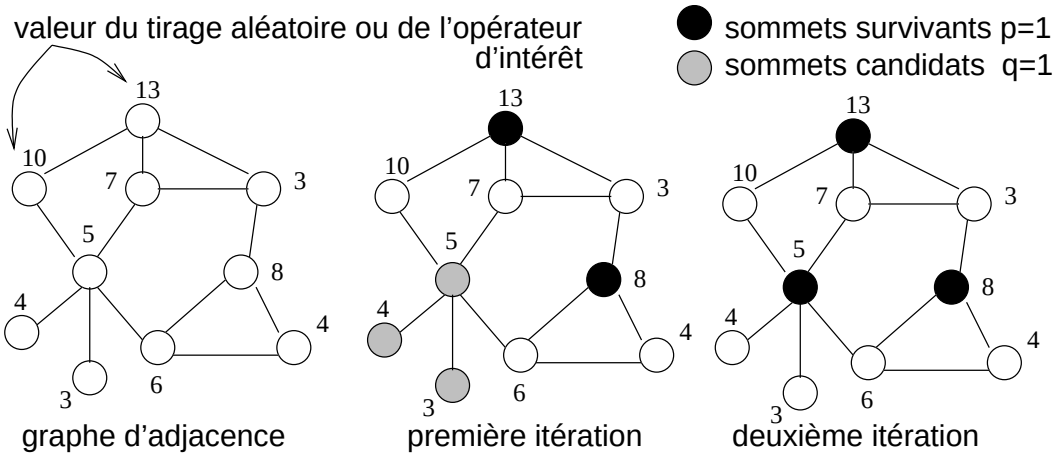


Fig. 3.4 : *Processus itératif de décimation sur un graphe.*

Calcul de p_i^1 et q_i^1 à la première itération :

$$p_i^1 = 1 \quad \textbf{si} \quad x_i \geq \max\left(x_j : s_j \in \Gamma_{G^k}(s_i)\right) \tag{3.1}$$
$$q_i^1 = 1 \quad \textbf{si} \quad (\forall s_j \in \Gamma_{G^k}(s_i) : p_j^1 = 0) \wedge (p_i^1 = 0) \tag{3.2}$$
$$\tag{3.3}$$

Puis à la l-ième itération :

$$p_i^l = 1 \quad \textbf{si} \quad x_i \geq \max\left(x_j : (s_j \in \Gamma_{G^k}(s_i)) \wedge (q_j^{l-1} = 1)\right) \tag{3.4}$$
$$q_i^l = 1 \quad \textbf{si} \quad (\forall s_j \in \Gamma_{G^k}(s_i) : p_j^l = 0) \wedge (p_i^l = 0) \tag{3.5}$$
$$\tag{3.6}$$

L'algorithme itère tant qu'il y a un sommet candidat c'est-à-dire $\exists x_i$ tel que $q_i^l = 1$.

La rapidité de convergence de cette procédure dépend de la distribution des x_i ; la figure 3.3 présente un cas défavorable où l'on ne peut dégager rapidement des survivants (sur la partie gauche) et où l'on est obligé d'attendre une vague survivant/non survivant... venant d'une partie du graphe dans lequel il y a un sommet localement maximum. Or ces configurations sont peu probables dans le cas où x_i est le résultat d'un tirage aléatoire.

3.5.3 Décimation adaptative

Jolion et Montanvert proposent dans [Jol92] d'adapter la décimation au contenu de l'image. Par conséquent, le tirage aléatoire est remplacé par un opérateur d'intérêt calculé sur l'image favorisant ainsi la survie des sommets qui maximisent ou minimisent[3] cet opérateur.

[3]Suivant le choix de l'opérateur

L'opérateur choisi est la variance de la région associée au sommet. Un sommet pour survivre doit être un minima ou un sous-minima local en termes de variance, ce qui revient à favoriser la survie des sommets associés à des régions homogènes. La variance des sommets du premier niveau, si ce premier niveau n'est pas issu d'une pré-segmentation[4], est calculée sur un voisinage 3×3.

Ce choix a un impact négatif [Jol92] sur le nombre d'itérations nécessaires à la convergence. En effet, dans les zones homogènes de l'image, d'importantes zones du graphe auront des variances égales, empêchant ainsi de dégager des maximas locaux. Il sera nécessaire d'avoir recours à un choix aléatoire.

3.6 Construction du graphe du prochain niveau : fusion des sommets

Nous avons évoqué la transposition de régions dans un graphe d'adjacence et la manière de le contracter en parallèle dans une approche ascendante. Dans l'optique d'une segmentation d'image, il est nécessaire que ces regroupements autour des sommets survivants se fassent sur des critères d'homogénéité. L'algorithme (algo. 2.2 p. 28) propose une approche de fusion séquentielle sur un RAG par la création d'une liste des sommets fusionables. Si l'on souhaite profiter de l'aspect parallèle des traitements pyramidaux, on doit se doter d'une structure de données reflétant les similarités entre sommets et supportant un traitement en parallèle.

3.6.1 Graphe de similarité

Le graphe de similarité $Sim(S, B)$ est issu du graphe d'adjacence $G(S, A)$, il traduit les propriétés d'homogénéité locale et donc de similarité entre régions.

Ce graphe est rarement connexe (à moins d'être issu d'une image homogène) il est constitué de composantes connexes (représentant chacune une région homogène de l'image) dont les arêtes relient les sommets similaires deux à deux.

Ce graphe est pondéré car les arêtes (cas non orienté) ou les arcs (cas orienté) portent une valeur calculée par une fonction $h()$ évaluant la similarité entre deux régions (§2.4 p. 26). $h()$ prend en paramètres les attributs de régions associés aux sommets.

3.6.1.1 Graphe de similarité : seuillage global

Ce graphe, $Sim(S, B) \subseteq G(S, A)$, est un graphe non orienté dont les sommets respectent deux à deux un prédicat global d'homogénéité ou de similarité.

$$\forall (s_p, s_q) \in A \text{ alors } (s_p, s_q, h(s_p, s_q)) \in B \iff h(s_p, s_q) \leq T_g$$

$h()$ étant dans ce cas une fonction globale, alors $h(s_p, s_q) = h(s_q, s_p)$, ce qui implique que le graphe n'est pas orienté.

[4] par exemple : utilisation d'une procédure de type "split"

3.6.1.2 Graphe de similarité : seuillage local

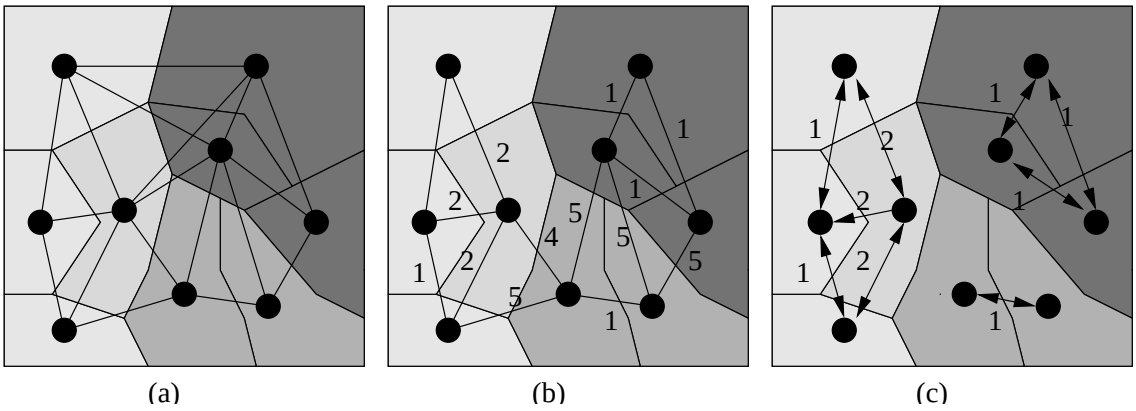


Fig. 3.5 : *(a) graphe d'adjacence ; (b) graphe de similarité obtenu avec un seuil global $T_g = 5$; (c) graphe de similarité obtenu avec un seuil local.*

Afin d'obtenir une analyse localement adaptée, il peut être intéressant d'utiliser un seuil local. Le seuil global définit un seuil en-deçà duquel deux régions ne sont pas similaires. Le seuil local est plus restrictif (donc inférieur) que le seuil global, il traduit une notion de similarité en fonction de l'environnement local.

Si l'environnement (le voisinage) de la région est "favorable" c'est-à-dire composé de régions qui lui ressemblent, on souhaite que la relation de similarité soit plus exigeante. La figure 3.6 page 45 illustre ce phénomène selon les cas (b) et (c). Dans le cas (c), les fusions entre régions de similarité moyenne sont retirées du graphe de similarité au niveau courant. Si l'environnement est "hostile" alors le seuil local se relâche avec comme borne inférieure le seuil global.

Dans cette approche, les fusions "moyennes" peuvent s'effectuer à un niveau supérieur mais on force d'abord les meilleures fusions. Cet aspect permet de pallier un des inconvénients des pyramides où des fusions inadaptées (respectant de peu le seuil global) font dériver progressivement niveau après niveau les caractéristiques des régions issues des fusions, ce qui a comme effet la création de discontinuités n'existant initialement pas dans l'image.

Montanvert et al. [Mon91] proposent une approche supposant qu'il n'y a que deux classes dans le voisinage de chaque sommet : ceux qui lui sont similaires et ceux qui ne le sont pas. On se dote d'une distance $h(s_p, s_q)$ entre deux sommets calculée sur les attributs de chaque sommet. Soit s_i le sommet courant et $\Gamma_{G^k}(s_i)$ son voisinage, on range les distances par rapport à s_i dans un ordre croissant : $\big(h(s_i, s_1) < ... < h(s_i, s_l) < ... < h(s_i, s_n)\big)$

Soit $m_1(l)$ et $m_2(l)$ les moyennes des deux partitions :

$$m_1(s_i, l) = \frac{1}{l} \sum_{j=1}^{l} h(s_i, s_j)$$

$$m_2(s_i, l) = \frac{1}{n-l} \sum_{j=l}^{n} h(s_i, s_j)$$

On cherche la partition en deux classes de cette liste maximisant un critère de variance interclasses, c'est-à-dire :

$$T_m(s_i) = \max_{l=1}^{n}(m_2(s_i, l) - m_1(s_i, l))$$

Finalement le seuil local $T_l(s_i)$ vaut :

$$T_l(s_i) = \min(T_m(s_i), T_g)$$

L'introduction du seuillage local va générer un graphe orienté car pour s_i, s_j voisins, $T_l(s_i) \neq T_l(s_j)$. La procédure de construction des relations de similarité est décrite (algo. 8.2 p. 155).

3.6.2 Décimation sur le graphe de similarité

La décimation, afin d'être adaptée au contenu de l'image, s'effectuera désormais sur le graphe de similarité. En effet, deux sommets non similaires ne sont pas en compétition pour représenter au niveau supérieur une zone homogène. Comme on peut l'observer en comparant les cas (a) et (b) (fig. 3.6, p. 45), en (a) le sommet de valeur 9 empêche le sommet de valeur 8 de survivre, lequel aurait été un meilleur représentant pour le sommet 6 que le sommet 7.

De ce fait, les deux règles de décimation (§3.5.2 p. 39) portent sur le graphe de similarité, ce qui signifie que le voisinage $\Gamma_{Sim^k}(s_i)$ d'un sommet s_i est considéré dans le graphe de similarité Sim^k.

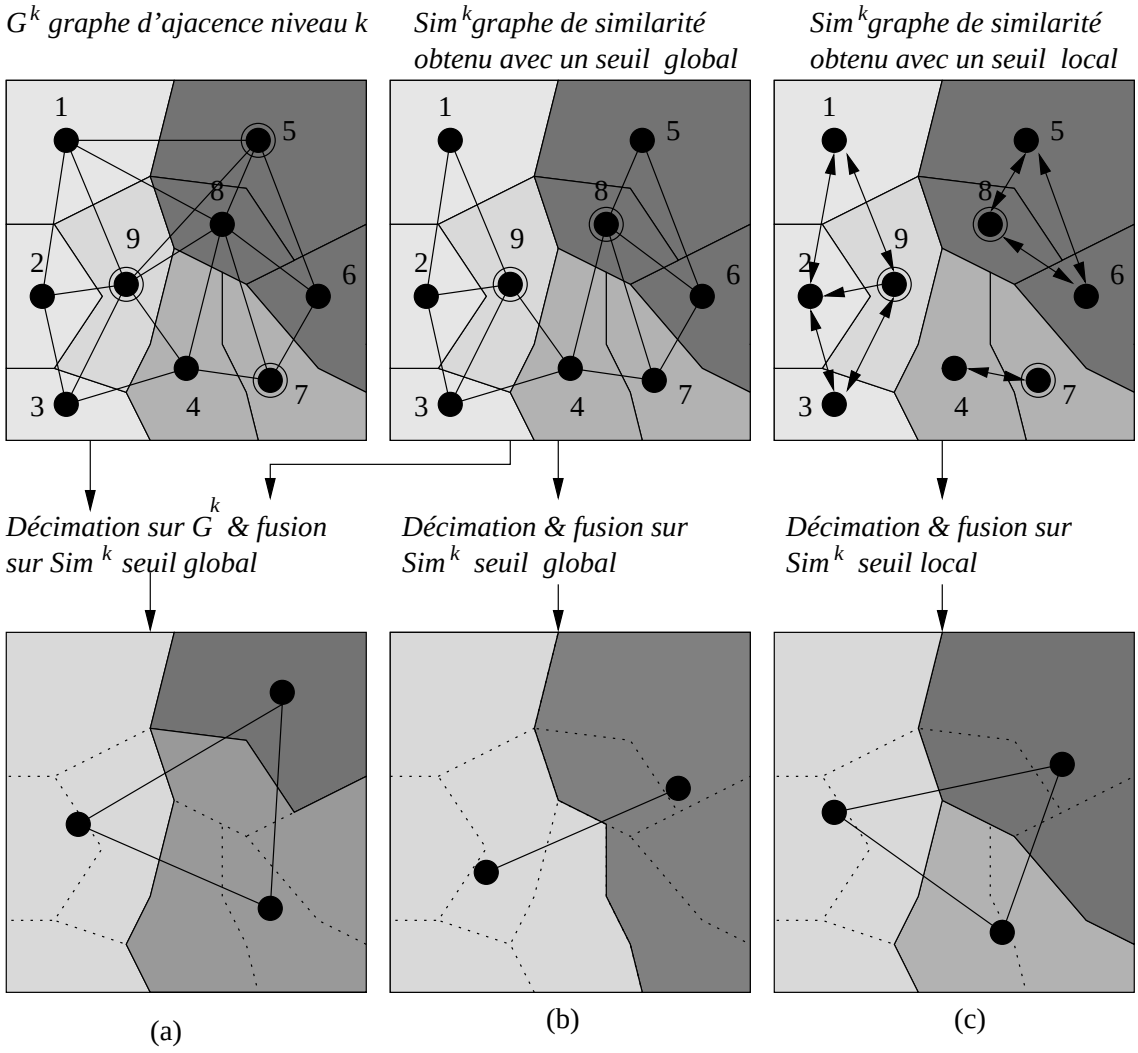


FIG. 3.6 : *Trois méthodes de décimation/fusion ; la valeur portée par les sommets est issue d'un tirage aléatoire (cas stochastique), ou de la réponse d'un opérateur d'intérêt(cas adapté).*

3.6.3 Fusion sur le graphe de similarité

On dispose d'un ensemble de sommets survivants : S^{k+1} et non-survivants : $S^k - S^{k+1}$.

Chaque sommet s_i non-survivant doit se rattacher (choisir un père) à un sommet survivant $\texttt{père}(s_i)$ de S^{k+1} en s'adaptant au contenu de l'image c'est-à-dire suivant des contraintes d'homogénéité. On sait qu'il existe un tel sommet dans le voisinage $\Gamma_{Sim^k}(s_i)$ de s_i car S^{k+1} est un ensemble stable maximal (règle 2) du graphe de similarité. La solution consiste à choisir parmi $\Gamma_{Sim^k}(s_i) \cap S^{k+1}$ le sommet le plus similaire c'est-à-dire :

$$\texttt{père}(s_i) \leftarrow \arg\min \left(h(s_i, s_j) : s_j \in \Gamma_{Sim^k}(s_i) \cap S^{k+1} \right)$$

3.6.4 Construction du graphe d'adjacence du niveau supérieur

Deux sommets du niveau $k+1$ vont être liés dans le graphe d'adjacence G^{k+1} s'il existe au moins un arc dans A^k liant un fils du premier sommet à l'un des fils de l'autre sommet :

$$\forall s_i, s_j \in S^{k+1}, (s_i, s_j) \in A^{k+1} \iff \exists (s_p, s_q) \in \texttt{fils}(s_i) \times \texttt{fils}(s_j) : (s_p, s_q) \in A^k$$

3.7 Construction de la pyramide de graphes

La segmentation avec les pyramides irrégulières est donc une construction ascendante à partir d'une partition initiale, qui peut être l'image elle même. Chaque graphe d'un niveau est obtenu en contractant le graphe du niveau inférieur et en rattachant les sommets non-survivants à un survivant de leur voisinage. Cette procédure, que nous allons résumer ci-dessous, est itérée tant que le nombre de sommets décroît significativement d'un niveau à l'autre.
Tant que ($Card(S^{k-1}) - Card(S^k) \geq$ contraction minimale)

1. Construire le graphe de similarité du niveau courant Sim^k. On peut avoir recours dans cette étape à un seuil global (graphe non orienté) ou local (graphe orienté) sachant que les solutions locales permettent de limiter certains artefacts inhérents aux pyramides.
2. Décimer les sommets du graphe. Le graphe en question peut être le graphe d'adjacence ou de similarité. La méthode de décimation pourra être stochastique ou adaptée. On préférera en général une approche de type décimation adaptée sur graphe de similarité car elle est plus adaptée au contenu de l'image, même si elle augmente sensiblement le nombre d'itérations pour atteindre la convergence.
3. Rattacher les non-survivants au sommet survivant de leur voisinage le plus similaire.
4. $k \leftarrow k+1$
5. $\forall s_i S^k$, calculer les attributs de la région associée au sommet s_i. Cette étape peut généralement être effectuée à partir des attributs des fils du sommet courant.
6. Construire le graphe d'adjacence du niveau courant en fonction des adjacences des fils des sommets du niveau inférieur.

3.8 Perspectives et conclusions sur les pyramides de graphes

Les pyramides irrégulières proposent une approche élégante pour la segmentation distribuée des images. Un de leurs atouts est le plongement du problème dans le cadre des graphes permettant donc de s'appuyer sur leurs fondements théoriques et d'utiliser le vaste outillage algorithmique mis à leur disposition.

L'approche débouche sur une pile de graphes permettant aux phases d'analyse ultérieures de choisir un niveau de résolution adéquate. De plus, ces graphes fournissent des informations sur la topologie entre régions, facilitant les étapes de vision haut niveau.

Cette approche "algorithmique" permet d'introduire de nombreux points de contrôle permettant d'intégrer de la connaissance *a priori* et de guider au mieux les fusions.

Cependant, comme Bertolino le fait remarquer dans [Ber95] : "d'une part des discontinuités significatives sur l'image originale ont tendance à disparaître ; d'autre part des frontières apparaissent dans des régions assez homogènes".

L'enrichissement du cadre décisionnel peut apporter une solution à ces problèmes. Par exemple, Bertolino et Montanvert [Ber96] proposent de prendre en compte les informations de contours afin de forcer ou interdire des fusions. Montanvert, Cho et Meer [Mon94, Cho97] proposent une segmentation par consensus de plusieurs pyramides obtenues à partir de la même image source.

3.9 La pyramide irrégulière duale et les noyaux de contraction équivalents

3.9.1 Introduction

Les pyramides sont des représentations hiérarchiques de l'image organisées en pile de graphes dont le nombre de noeuds décroît lorsque la pyramide est parcourue de la base à l'apex. La pyramide irrégulière stochastique [Mee89] s'appuie non pas sur des grilles rigides comme le font classiquement les pyramides, mais plutôt sur des pavages irréguliers.

Cette flexibilité se paie par un algorithme de décimation dont ni la structure de données, ni le nombre d'itérations ne sont bornés. D. Willersinn propose des pyramides irrégulières duales [Wil94] dont la décimation s'effectue sur le graphe d'origine ainsi que sur son dual. Ces traitements opèrent sur une structure de données bornée par un algorithme itératif dont le nombre d'itérations est lui aussi borné.

Les pyramides irrégulières ont obtenu des résultats prometteurs en segmentation, néanmoins un problème récurrent est l'extraction des régions de petite taille et des régions de grande taille à des niveaux différents.

Cette difficulté peut être dépassée par l'usage d'un graphe de similarité [Mon91] qui permet de bloquer les contractions indésirables. Un autre moyen est simplement d'autoriser des facteurs de contraction différents suivant la localité. W.G. Kropatsch y parvient par l'utilisation des noyaux de contraction équivalents [Kro95].

Les sections 3.9.2 (définition du dual), 3.9.3 (paramètres de décimation) et 3.9.4 (décimation duale) définissent la pyramide duale à l'aide du formalisme de W. G. Kropatsch, introduisant ainsi la section 3.9.5 qui porte sur l'extension de son formalisme aux noyaux de contraction équivalents. Finalement, la section 3.9.6 propose quelques perspectives offertes par de tels outils.

3.9.2 Graphe dual G^*

Soit un graphe planaire $G_i(N_i, A_i)$, où N_i est l'ensemble des noeuds et A_i l'ensemble des arcs. Ce graphe peut être un graphe d'adjacence de régions obtenu directement depuis l'image ou à l'aide d'une pré-segmentation. N_i est alors identifié aux régions et A_i aux relations binaires de voisinage entre régions. On observe des facettes ; les arêtes sont des arcs de A_i et les sommets des noeuds de N_i.

Le graphe dual $G_i^*(F_i, D_i)$ est tel que F_i est identifié à l'ensemble des facettes, et D_i à l'ensemble des relations binaires de voisinage entre facettes.

Deux facettes sont voisines si elles partagent une arête en commun. De facto, il existe une bijection reliant les arcs de A_i aux arcs de D_i. Supprimer un arc de A_i revient à supprimer un arc de D_i et réciproquement.

W. D. Kropatsch [Kro94] montre qu'en général ni G_i, ni G_i^* ne sont simples, i.e. ils possèdent des boucles et des arcs doubles. Or, travailler systématiquement avec des graphes d'adjacence de régions simples correspond à une mauvaise description de la topologie des régions, ce qui se répercute dans la construction du dual.

3.9.3 Paramètres de décimation et contraintes associées

La décimation duale proprement dite, tout comme la décimation classique [Mee89], ne commence qu'après que les noeuds et les arcs aient été partionnés en survivants et non-survivants.

On note $S_i \subset N_i$ les sommets survivants et $A_{i,i+1} \subset A_i$ les arcs non-survivants. Ces arcs relient chaque non-survivant ($s \in N_i - S_i$) à un de leur voisin survivant.

Connaissant G, le couple $(S_i, A_{i,i+1})$ suffit à retrouver ces deux partitions i.e. les sommets survivants et non-survivants. Le couple $(S_i, A_{i,i+1})$ est appelé paramètre de décimation [Kro94].

Des techniques de sélection de $(S_i, A_{i,i+1})$ sont détaillées dans [Mee89, Mon91, Jol92, Ber93]. Le couple $(S_i, A_{i,i+1})$ doit vérifier des contraintes que ce soit pour la pyramide irrégulière classique [Mee89] ou duale [Wil94]. Dans la pyramide irrégulière (§3.5.2 p. 39), les contraintes sont les suivantes :

- deux noeuds de S_i ne peuvent être voisins [C1] ;
- un arc de $A_{i,i+1}$ a pour extrémités, un noeud survivant et un autre non-survivant. Chaque noeud non-survivant est associé à un seul arc de $A_{i,i+1}$. [C2]

Dans le cadre de la pyramide duale, [C1] est abandonnée. On remarque alors que les paramètres de décimation munis de la contrainte [C2] forment une forêt recouvrante.

Les arbres prennent leurs racines parmi les noeuds S_i et leurs branches sont constituées par les arcs de $A_{i,i+1}$. Ces arbres sont appelés noyaux de contraction [Kro95]. La contrainte [C2] impose aux arbres d'être de profondeur 1, ce qui implique que le facteur de contraction soit identique quel que soit le noyau de contraction considéré.

3.9.4 Décimation duale

La contraction duale (appelée aussi décimation duale) est une fonction C qui associe aux graphes (G_i, G_i^*) les graphes (G_{i+1}, G_{i+1}^*) et dont le processus est contrôlé par les paramètres de décimation $(S_i, A_{i,i+1})$. Une contraction peut être formulée de la manière suivante [Kro94] :

$$(G_{i+1}, G_{i+1}^*) = C\Big[(G_i, G_i^*), (S_i, A_{i,i+1})\Big]$$

Le processus de décimation au sens large se compose de 5 étapes [Wil94] [Kro94] :

1. sélection des paramètres de décimation $(S_i, A_{i,i+1})$;
2. contraction des arcs de $A_{i,i+1}$:
   - identification des extrémités de chaque arc de $A_{i,i+1}$;
   - suppression des arcs de $A_{i,i+1}$ et des arcs duaux ;
3. sélection des paramètres de décimation $(S_i^*, A_{i,i+1}^*)$ pour l'élimination des facettes inutiles :
   - Toute facette dont le degré est supérieur à 2 appartient à S_i^* ;
   - S_i^* et $A_{i,i+1}^*$ vérifient les contraintes [C1] et [C2] ;
4. contraction des arcs de $A_{i,i+1}^*$;
5. retour en 3, tant qu'il existe des facettes de degré inférieur à 3.

L'étape (2.) crée des redondances topologiques, i.e. des arcs doubles et des boucles. Les étapes (3.), (4.) et (5.) sont chargées de les éliminer. D. Willersinn [Wil94] montre que le voisinage d'une facette reste identique ou diminue au cours des décimations successives.

L'algorithme n'utilisant pas le voisinage des noeuds du graphe G_i mais au contraire ceux de son dual G_i^*, la structure de données utilisée peut être bornée en espace mémoire.

L'étape (4.) peut créer de nouvelles facettes à éliminer, c'est pour cela que l'on réitère le procédé en (5.). L'apparition de ces nouvelles facettes ne peut se faire que dans un sous-graphe situé autour du lieu de la contraction (4.). La portée limitée des effets de la contraction et le degré borné des facettes sont les arguments à la base de la démonstration sur l'existence d'une limite sur le nombre d'itérations de l'algorithme [Wil94].

3.9.5 Noyaux de contractions équivalents

On remplace la contrainte [C2] par la contrainte [C3] : un arc de $A_{i,i+1}$ a pour extrémités un noeud survivant et un autre non-survivant ou deux noeuds non-survivants. Chaque noeud non-survivant est relié à un unique noeud survivant par un chemin constitué des arcs de $A_{i,i+1}$.

Les noyaux de contraction munis de la contrainte [C3] deviennent alors des arbres de profondeur quelconque. L'ensemble de ces arbres forme toujours une forêt recouvrante [Kro95]. On peut en tirer deux conséquences :

1. il devient alors possible d'avoir des facteurs de contractions multiples sur un même niveau ;
2. la composition de plusieurs contractions devient elle-même une contraction. Cela signifie aussi que n'importe quel niveau de la pyramide duale est calculable à partir du graphe initial et des noyaux de contraction équivalents [Kro95].

$$\begin{aligned} C\Big[C\big[(G_i, G_i^*), (S_i, A_{i,i+1})\big], (S_{i+1}, A_{i+1,i+2})\Big] &= C\Big[(G_i, G_i^*), (S_i, A_{i,i+2})\Big] \\ &= (G_{i+2}, G_{i+2}^*) \end{aligned}$$

En composant toutes les contractions nécessaires pour obtenir une pyramide de sa base à son apex, il en résulte un unique noyau de contraction équivalent, i.e. un arbre recouvrant dont la racine représente l'apex. En décorant les noeuds et les arcs de cet arbre par des labels désignant le niveau où ils deviennent non-survivants, nous obtenons une autre représentation de la pyramide irrégulière. En se dotant d'opérations modifiant les labels et la structure même de l'arbre, il est alors possible d'énumérer toutes les pyramides irrégulières duales existantes [Kro95].

3.9.6 Perspectives

Les travaux sur la pyramide duale ne semblent concerner qu'une implantation matérielle parallèle. Néanmoins, l'abandon de la contrainte [C1] et le remplacement de la contrainte [C2] par [C3] accélère la convergence du processus de décimation : il n'y a plus de synchronisation globale portant sur tous les sommets d'un niveau avant de passer au suivant. La pyramide duale permet une extraction rapide des maximas locaux c'est-à-dire des régions importantes et faciles à extraire. Cette information peut ensuite être utilisée dans une approche descendante pour assister l'extraction des régions plus délicates.

Comme nous le verrons dans le chapitre 4 consacré à la vision, on retrouve souvent ce principe d'un processus guidé par les données se focalisant sur les zones les plus faciles à traiter. L'information extraite permet dans un second temps d'apporter des contraintes supplémentaires au traitement des zones sensibles.

Chapitre

4

Les systèmes de vision

Notre objectif futur étant d'intégrer les étapes du moyen et haut niveau de la vision, nous présentons dans ce chapitre un état de l'art des systèmes de vision. Cette étude permet un élargissement du domaine d'analyse indispensable à la conception d'un module de segmentation, dont la vocation est d'être intégré dans un système plus vaste.

Les recherches sur les systèmes de vision ont donné lieu à des réflexions sur une méthodologie plus générale de conception des systèmes de vision incluant l'étape de segmentation. Nous commencerons par évoquer ces courants de pensée qui ont grandement influencé notre approche. Nous montrerons que ces approches aboutissent au cadre pluridisciplinaire de la systémique que nous tâcherons de résumer.

Nous décrirons ensuite (§4.3 p. 58) les différents composants intervenant dans un système de vision dans le cadre de l'approche traditionnelle. Ainsi, nous évoquerons la théorie computationnelle décrivant le modèle c'est-à-dire *le pourquoi* et *le quoi* des entités calculées.

Puis, nous évoquerons les différentes stratégies de navigation (§4.5 p. 68) dans cet environnement d'informations et les différentes approches de mises en correspondance entre un modèle et des observations.

Finalement, nous aborderons *le comment* c'est-à-dire les techniques de représentation de la connaissance (§4.6 p. 72).

4.1 Introduction

Un système de vision a pour objectifs de construire et maintenir une description utile du monde extérieur à partir d'images perçues. Le système procède par une mise en correspondance entre les informations perçues et un modèle.

Ce processus est confronté à des ambiguïtés dues à une information entachée d'erreurs, subissant ainsi des distorsions. De plus, le problème à résoudre ne peut être complètement spécifié, les objets cherchés étant imprédictibles et multiformes (une maison n'a pas toujours le même nombre de fenêtres). C'est donc un problème inverse mal posé qui subit l'absence de bijection entre un modèle et une observation. Intuitivement, il convient d'ajouter de nouvelles contraintes afin de lever au mieux les ambiguïtés.

4.2 Méthodologies de l'analyse des systèmes de vision

Les différentes méthodologies d'analyse du problème de la vision reflètent les différents courants de pensées philosophiques et scientifiques d'analyse des systèmes complexes.

4.2.1 De l'approche traditionnelle

La première approche, le reconstructionnisme, formalisée par Marr [Mar82] tend à considérer le système de vision indépendamment des objectifs et du contexte d'acquisition. L'objectif générique est de reconstruire en interne une représentation 3D de la scène. On procède par l'approche "classique" de Descartes [Des37] qui consiste à décomposer le problème en autant de fois qu'il est nécessaire pour obtenir des sous-problèmes solubles.

Axe d'analyse suivant le détail dans les spécifications du système : Marr propose de décomposer l'analyse du problème suivant un axe dénotant le détail dans les spécifications. On y trouve trois niveaux :

1. **La théorie computationnelle** concerne l'étude des entités à calculer, la justification et la logique de leurs calculs et les stratégies commandant ces derniers. C'est *le quoi et le pourquoi* du système.
2. **Les représentations et les algorithmes** affinent le premier point en précisant la mise en œuvre logicielle sous forme de structures de données et d'algorithmes.
3. **L'implémentation** est la dernière étape dans laquelle l'on traite l'implémentation matérielle.

Axe d'analyse suivant les niveaux de description de l'information : à ce premier axe d'analyse s'ajoute un deuxième axe complémentaire. Cet axe s'étend le

long des niveaux de représentation des informations traitées par système de vision : du numérique au sémantique en passant par le symbolique.

Nous avons choisi une décomposition plus détaillée que celle fournie par Marr traduisant les étapes généralement utilisées en vision (fig. 4.3, p. 61) :

1. **l'image**, ou niveau signal est une grille 2D d'intensités lumineuses ;
2. **le croquis élémentaire** : niveau symbolique ou syntaxique qui est constitué des attributs 2D de l'image (régions, segments) ;
3. **le croquis 2.5 D** : niveau symbolique ou syntaxique dénotant les propriétés tridimensionnelles locales des surfaces visibles comme leurs orientations et leurs profondeurs par rapport à l'observateur ;
4. **le croquis 3 D** : niveau symbolique si c'est une simple description des objets physiques en termes de volume. Ce peut être le début du sémantique si à l'instar de Marr on ajoute leur interprétation et leur organisation spatiale ;
5. **l'interprétation** : niveau sémantique interprétant les objets ou la scène observée. Certains auteurs fusionnent ce niveau avec le précédent.

McClamrock propose une méthodologie d'analyse combinant les deux axes, c'est-à-dire une analyse suivant le premier axe pour chaque niveau du deuxième axe. Dans cette approche, le système considéré est la scène, phénomène résultant, qui a une existence intrinsèque indépendante de l'observation faite par l'outil de vision. D'après Lewes [Lew75] les phénomènes résultants peuvent être abordés par une approche analytique. Ainsi, la méthodologie selon les deux axes précédents consiste à concevoir un modèle analytique que l'on va confronter à l'observation par une succession de transformations ascendantes de l'information. La validité des transformations est fondée sur la validité du modèle analytique et statique de la théorie computationnelle.

Pour expliquer la lenteur des progrès en vision, Pavlidis [Pav92] évoque la simplicité des modèles utilisés. Cependant, cette vision des choses se trouve confrontée à la complexité insurmontable de la conception de tels modèles pour traiter un problème mal posé (absence de bijection scène/modèle).

Pour résoudre ce problème, il s'agit plutôt de trouver de nouvelles **contraintes** à appliquer à l'outil de perception afin de résoudre les ambiguïtés. La question est : d'où pouvons nous tirer ces nouvelles contraintes ?

4.2.2 Vers une approche systémique

À la question précédente, certains chercheurs préconisent de tirer ces nouvelles contraintes des buts, de l'outil de vision lui-même et du procédé d'acquisition de l'image.

Il convient donc d'adopter une attitude d'analyse plus globale (systémique) en ajoutant au système qu'on analyse l'ensemble des éléments intervenant dans le traitement de la scène c'est-à-dire les buts, le système de vision lui-même, etc afin de

trouver de nouvelles contraintes. Nous allons évoquer différentes approches qui intègrent progressivement davantage d'éléments dans le système analysé.

La vision active (Aloimonos [Alo88]) propose un observateur actif se déplaçant de manière connue permettant d'enrichir l'information acquise et ainsi de mieux contraindre son traitement. Il utilise une reconstruction interne ne rompant pas avec l'approche reconstructionniste.

La vision intentionnelle énoncée par Aloimonos [Alo90] insiste sur les buts du système de vision. De cette manière la vision sera guidée par des buts précis ne nécessitant pas une reconstruction exhaustive de la scène. La prise en compte explicite de buts permet de contraindre le processus de vision selon le paradigme énoncé par Bajcsy *"we do not see, we look"*. Aloimonos illustre ses propos à travers un système de vision active intentionnelle : Medusa.

Les deux approches suivantes proposent des déclinaisons autour de la vision guidée par les buts en anglais : "goal-driven" ou "purposive vision".

La perception active proposée par Bajcsy [Baj88] enrichit l'approche de la vision active en introduisant une boucle de rétroaction sur le module d'acquisition, guidée par les buts de la perception.

La vision animée proposée par Ballard [Bal91] étend la notion de perception active en ne considérant le système que comme un élément constitutif d'une entité plus vaste interprétant un environnement, s'y déplaçant et y agissant. Cette approche rompt définitivement avec le reconstructionnisme et s'inspire de l'approche béhavioriste de Brooks [Bro86] qui considère que la meilleure représentation du monde est le monde lui même excluant ainsi tout modèle interne de celui-ci.

L'approche systémique proposée par Jolion [Jol94] énonce explicitement le fait que la vision est un élément d'un système plus global non réduit à l'analyse indépendante de ses parties, [Jol01].

> *"Le paradigme systémique fournit une approche à la conception d'une méthodologie générale pour la compréhension des systèmes, qui se veut non réductionniste, i.e. dans le sens de Descartes. Ce dernier proposait de décomposer un problème en sous-problèmes, de résoudre ces sous-problèmes séparément et ensuite de revenir au problème initial. Au contraire, la systémique privilégie le système vu dans sa globalité et propose donc que celui-ci soit étudié en premier".*

Ce cadre d'analyse permet de dégager de nouvelles contraintes émergentes des interactions entre éléments du système global. Ceci est fait en rendant explicite l'existence de contraintes de différentes natures. En effet, l'approche systémique identifie clairement plusieurs types de contraintes :

1. **Les contraintes objectives** sont intrinsèques à l'objet et ne prennent pas en compte l'observateur. L'approche traditionnelle a focalisé son attention sur ce type de contraintes. Les travaux menés en segmentation d'image en sont aussi un parfait exemple.
2. **Les contraintes idéalistes** sont centrées sur l'observateur (le sujet). Elles partent du principe que la scène est une réalité subjective observée, l'observateur dispose de procédures visuelles et d'une architecture spécifique. L'analyse de ces spécificités apporte des informations permettant de contraindre le système de vision. Par exemple, l'analyse des spécificités et des limites du système visuel humain est une importante source d'inspiration.
3. **Les contraintes systémiques** portent sur l'analyse globale du système et sur les propriétés émergentes de l'interaction entre ses parties. L'analyse des interactions entre composants et impact global en résultant, évaluée en termes d'optimalité, de robustesse ou d'efficacité, doit permettre d'améliorer **l'intégration** de ces différents composants.

4.2.3 Discussion

4.2.3.1 Digression sur les différents courants de pensée

L'évolution des approches en système de vision est symptomatique de l'évolution de courants de pensée forts et convergeants du XXème siècle. De plus, les concepts à la base des systèmes multi-agents, que nous utilisons, sont l'expression informatique de ces courants de pensée. Nous allons donc brièvement évoquer quelques points caractérisant ces derniers.

Face à l'échec de la méthodologie réductionniste de Descartes pour l'analyse des systèmes complexes, les philosophes, épistémologues et scientifiques ont suggéré la nécessité d'une approche pluridisciplinaire et globale. Cette approche s'appuie sur le fait qu'une décomposition analytique (reconstructionnisme en vision) transforme l'objet étudié et ne permet pas d'envisager les caractéristiques émergentes de l'interaction de ses composantes : "le tout est plus que la somme de ses parties".

La systémique est donc au carrefour d'un ensemble de courant de pensée. Le **holisme**, par exemple, conçoit le réel comme constitué d'un ensemble d'entités interdépendantes. Il concentre son analyse sur les interactions entre entités. **L'émergentisme** quant à lui considère que les caractéristiques d'un système ne se retrouvent dans aucune de ses parties mais qu'elles sont propres à la totalité du système [Ber99]. Ce principe est bien illustré par la *Gestaltpsychologie* ou "psychologie de la forme" apparue dans les année 20 en Allemagne qui énonce le fait que les phénomènes visuels sont perçus dans leur totalité. Appliqué aux activités cognitives des êtres vivants et plus particulièrement à la conscience humaine, ce courant de pensée dit qu'une analyse ne peut être menée hors de leur cadre de vie (le système analysé doit inclure l'environnement). Fransisco Varela [Var98] insiste sur l'aspect "incarné" dans un environnement au travers d'une interaction "dynamique" comme prérequis nécessaire au développement et à l'analyse des facultés cognitives. On retrouve ce principe

dans l'approche de Ballard ainsi que dans l'approche *réactive* de Brooks [Bro86] qui préconise un observateur sans mémoire propre (opposition au reconstructionnisme et au cognitivisme) utilisant à travers une interaction dynamique l'environnement comme mémoire externe. Ce mode de pensée est parfaitement illustré par la cybernétique de Norbert Wiener. Un problème comme le contrôle de la température dans une pièce traité par une approche analytique, nécessiterait la modélisation des impédances thermiques des murs de la pièce, de la prise en compte de la température extérieure, du vent, etc. Une approche cybernétique, dynamique et incarnée règle le problème très simplement à l'aide d'une boucle de rétroaction implantée sous la forme d'un thermostat.

Donnons un autre exemple de la pensée systémique. Dans un bâtiment relativement haut les usagers se plaignent de devoir attendre l'ascenseur, les ingénieurs optimisent les algorithmes de gestion de l'ascenseur mais rien n'y fait, les usagers se plaignent toujours. En complexifiant le problème c'est-à-dire en considérant les usagers comme faisant partie du système, on ajoute des miroirs dans lesquels ils peuvent se regarder à côté des portes de l'ascenseur. Les plaintes cessent !

4.2.3.2 Quelle méthodologie ?

Le débat sur le choix d'une méthodologie a donné lieu à une série d'articles [CVG94] opposant les reconstructionnistes (citons Tarr et Black [Tar94]) aux supporters de la vision intentionnelle (*purposive vision*) (citons Aloimonos [Alo94]). On reproche à la reconstruction de vouloir bâtir un système de **vision général** à l'instar de la vision humaine sans intégrer les buts du système, se confrontant ainsi à l'extrême difficulté de la conception de modèles des entités à calculer (la théorie computationnelle).

On reproche aux tenants de la vision guidée par les buts des conceptions trop simplistes et spécialisées n'enrichissant pas une théorie générale.

Les tenants de la vision intentionnelle argumentent que leur approche a donné des résultats et qu'elle fournit une nouvelle approche méthodologique (tendant vers la systémique). Tandis que les reconstructionnistes revendiquent une théorie générale dont la vision guidée par les buts n'est qu'un cas particulier.

Nous pensons qu'on ne peut pas feindre d'ignorer le saut méthologique apporté par la vision intentionnelle/active dans son élargissement du système analysé et dans son incarnation (vision animée). Cette étape vers une approche systémique ainsi que les nouvelles contraintes générées garantissent la viabilité de l'approche.

On ne peut pourtant condamner l'aspiration à faire un système de vision général, d'autant que les concepts qui y sont développés enrichissent la connaissance du domaine et ont parfois des retombées économiques (niveaux bas et intermédiaire du système VISIONS [Han78]). De plus, les niveaux de représentation introduisent une progressivité dans l'interprétation et également une diminution de la complexité des calculs dont on peut difficilement se passer dans certains cas complexes.

L'approche systémique dépasse ces deux approches. Ancrer l'analyse des systèmes de vision dans un cadre systémique permet d'identifier plus clairement les différents composants du système. Cependant, la complexité résultante peut paraître déroutante (par ex. le système de vision fait partie du système analysé), elle peut générer des problèmes d'auto-références [Hof85] et d'obscurs débats du type "tout est dans tout, et réciproquement". Il est donc nécessaire de se doter de méthodes capables d'appréhender cette nouvelle complexité. Tostsos [Tso94] et Jolion [Jol94] proposent des pistes articulées autour de l'analyse des contraintes.

Selon l'approche systémique, la vision est un phénomène émergeant de l'interaction de la scène, des capteurs, des buts et du système de vision qui lui même est composé d'un ensemble de modules traduisant des connaissances très variées. Il est donc nécessaire de se doter d'outils capables de traduire naturellement cette méthodologie émergentiste de la même manière que la programmation objet traduit une méthodologie de conception de logiciels. Ces outils doivent permettre une bonne **intégration** de ces éléments et fournir une **structure de contrôle** flexible. Les approches multi-spécialistes à base de tableaux noirs ou multi-agents proposent des solutions élégantes permettant une modélisation explicite des interactions entre composants.

Nous allons à présent retourner dans le cadre traditionnel de l'analyse des systèmes de vision. Nous nous plaçons donc dans la typologie proposée par D. Marr, et nous commencerons par évoquer le niveau conceptuel ou niveau de la théorie computationnelle.

Nous proposons trois axes de décomposition du niveau conceptuel :

1. l'axe des informations sur lequel on trouve le modèle et les observations (§4.3 p. 58) ;
2. l'axe des fonctionnalités sur lequel on trouve les tâches ou procédures visuelles (§4.4 p. 64) ;
3. l'axe des stratégies d'application des tâches sur les informations (§4.5 p. 68).

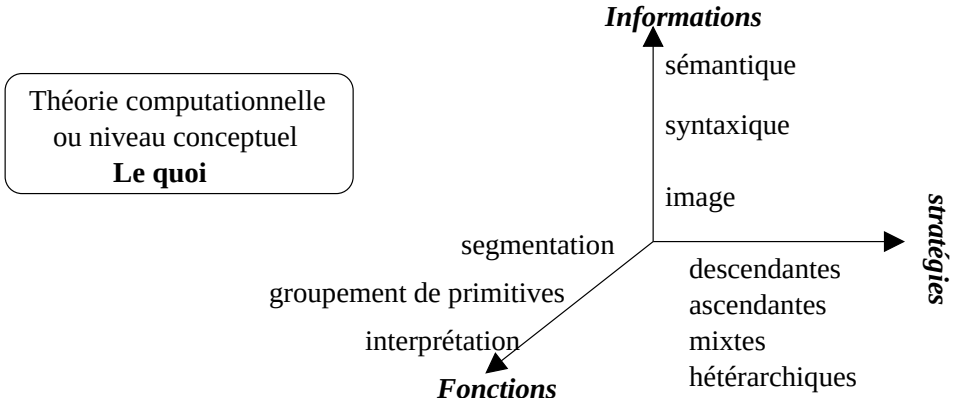


FIG. 4.1 : *Les trois axes de décomposition du niveau conceptuel.*

4.3 Les informations composant la théorie computationnelle

Nous allons dans cette section évoquer l'ensemble des composants intervenant dans le processus de vision par ordinateur selon l'approche traditionnelle. Un système de perception a comme objectif de fournir une représentation utile des images perçues. Les reconstructionnistes se sont fixé comme objectif de fournir une représentation sémantique de la scène afin de pouvoir en faire une **interprétation**. Sous cet aspect, la vision est un processus de transformation d'une information quantitative et numérique en une information qualitative et sémantique.

Cette transformation s'effectue généralement de manière progressive, à travers différents niveaux d'abstraction dont le nombre dépend de l'application. Les connaissances acquises sur la vision humaine confirment cet aspect des choses : elle est composée de couches de neurones hiérarchiquement structurées [Jol01, chap. 2]. C'est ce que décrit la décomposition selon l'axe sémantique présentée (§4.2.1 p. 52) (signal, symbolique, sémantique).

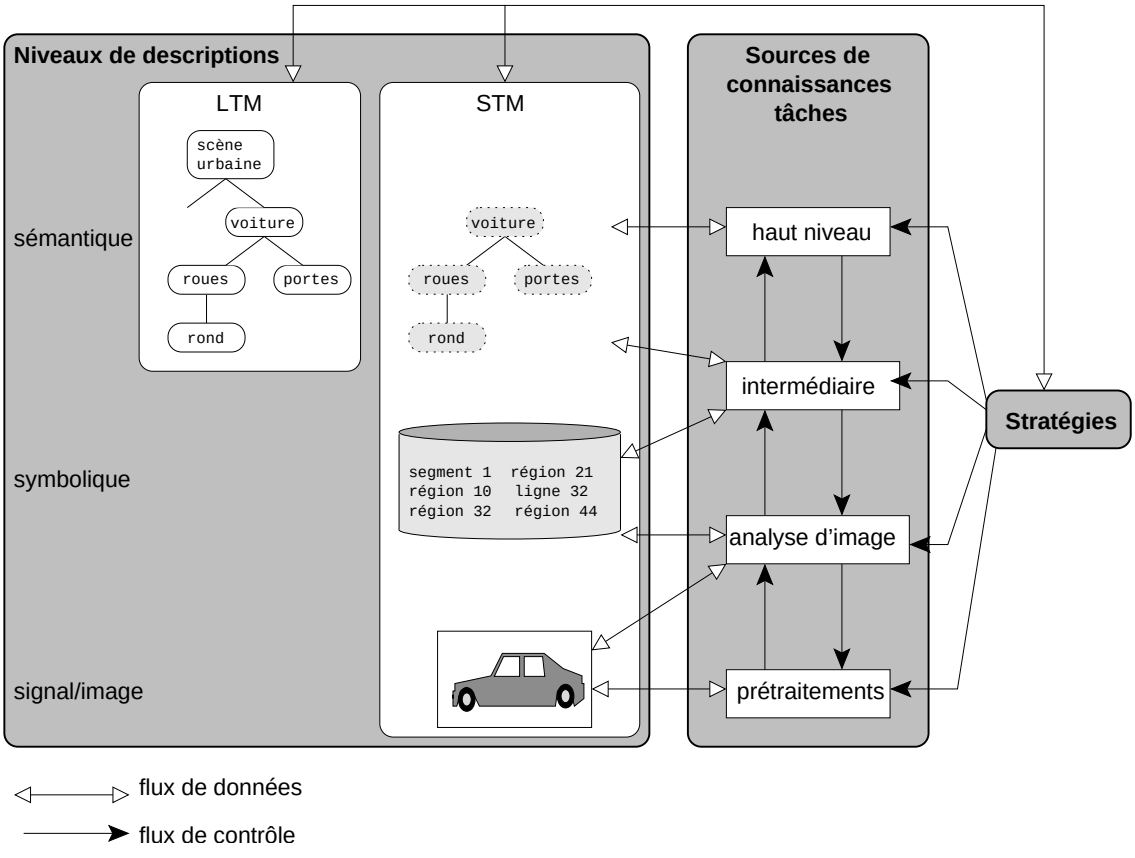


Fig. 4.2 : *Les composants d'un système de vision orienté reconstruction de concepts.*

Chaque plan de cette représentation hiérarchique se décompose en deux sous-parties (fig. 4.3, p. 61) :

1. **Les connaissances *a priori*** forment un prototype des entités (primitives d'image, objets, scènes) que l'on souhaite identifier dans l'image. On y retrouvera des contraintes portant sur des valeurs attendues des attributs et des relations spatiales (§4.3.1 p. 59) & (§4.3.2 p. 61). On utilise souvent le vocable de LTM (Long Time Memory) à l'image des règles dans un système à base de connaissances.
2. **Les observations** obtenues en transformant l'information (l'image à l'origine) niveau après niveau. Cette mémoire à court terme STM (Short Time Memory) représente l'état courant de perception du système.

Les connaissances dans le système de vision peuvent aussi être distinguées selon qu'elles sont descriptives ou opératoires :

1. **Les connaissances descriptives** se présentent comme une liste d'attributs numériques et/ou symboliques décrivant les entités manipulées par le système. Elles modélisent aussi les différents types de relations entre ces entités, relations qui peuvent être spatiales, de composition, de spécialisation ou de vue.
2. **Les connaissances opératoires** manipulent et combinent les connaissances descriptives pour en extraire de nouvelles. On utilise le vocable de source de connaissances, tâche ou KS (*Knowledge Source*) pour les procédures purement calculatoires : "tâche de segmentation" par exemple. Ces KS traitent généralement des informations du niveau bas ou intermédiaires, et sont indépendantes du domaine d'application.

   On utilise plutôt le terme de "stratégies" pour les procédures liées au contrôle du système, c'est-à-dire savoir quelle entité il est intéressant de traiter aux vues des connaissances descriptives. Les stratégies manipulent plutôt des informations de haut-niveau et sont dépendantes du domaine. Les connaissances opératoires sont la plupart du temps des connaissances à long terme, elles peuvent être amenées à évoluer dans le cadre de l'apprentissage.

4.3.1 Connaissances descriptives symboliques

Les spécialistes bas-niveau (early vision) du niveau intermédiaire extraient une description intermédiaire symbolique (*tokens* en anglais) de l'image. On retrouve couramment le vocable de primitives d'image pour qualifier ces données. On trouve tout d'abord des primitives d'image qui traduisent des **indices visuels intrinsèques** à l'image. Puis, progressivement, le système combine ces primitives en les confrontant au modèle (LTM) pour construire des entités plus complexes portant par exemple des indices de profondeur ou d'orientation de surface traduisant l'aspect 3D de la scène.

Voici un bref récapitulatif des entités que l'on peut trouver dans un système comme VISIONS [Han78] (fig. 4.3, p. 61) :

1. Régions issues d'étapes de segmentation représentées à l'aide de plusieurs attributs :

   (a) attributs photométriques, comprenant la moyenne, la variance, l'histogramme ou même les attributs de texture ;
   (b) attributs géométriques comme la surface, le périmètre, le défaut de circularité, l'axe principal, etc.
2. Segments de contours présents dans l'image, extraits par segmentation. On y code la longueur du contour, son orientation, le contraste le long du contour.
3. Groupements de primitives obtenus en combinant entre eux des primitives à l'aide de procédures visuelles intermédiaires. On y trouve essentiellement des chaînes de segments de contour, des lignes parallèles (perpendiculaires), etc.
4. Les surfaces ne sont présentes que si l'on souhaite faire une reconstruction 3D. Ces surfaces sont des régions auxquelles on ajoute des propriétés tridimensionnelles locales comme l'orientation.
5. Des objets 3D de la scène traduisant la structure tridimensionnelle des objets.

Cette représentation intermédiaire et symbolique de l'image permet de réduire la complexité des traitements ultérieurs. Mais comme le souligne Draper [Dra89], cette représentation intermédiaire peut être plus volumineuse que l'image elle même. En effet, une segmentation préliminaire peut aisément extraire 300 régions et 4000 segments pouvant "peser" quelques méga-octets.

Le problème récurrent rencontré à cette étape de la vision est l'explosion combinatoire générée par les différentes combinaisons (groupements) entre ces primitives. L'accès intensif fait par les procédures visuelles intermédiaires nécessite une organisation efficace. Les solutions existantes se présentent sous forme de bases de données (par ex. ISR Intermediate Symbolic Representation [Bro89]), permettant une indexation selon deux modes :

1. L'accès associatif qui correspond à des requêtes appliquant des contraintes sur les attributs des primitives, par exemple : accès à toutes les lignes de fort contraste.
2. L'accès spatial qui correspond à des requêtes ciblées sur une zone d'image précise.

Boucher [Bou99] suggère que l'accès à ces structures de données soit aussi libre et ouvert que possible, permettant une mise en commun précoce et accessible aux procédures visuelles des connaissances extraites. Cette base de données peut alors être vue comme un environnement modifié par des entités des traitements qui y accumulent de manière incrémentale des informations.

Certains systèmes [Han78] ajoutent à ces primitives un ensemble de liens vers d'autres connaissances descriptives. Ces liens peuvent traduire des relations d'instanciation, de vue entre éléments de différents niveaux de description, de contexte spatial. Ces liens peuvent donc traduire un ensemble de contraintes de cooccurrence entre primitives, il peut aussi permettre de propager des certitudes au sein de ce réseau d'entités.

Il est à noter qu'aucun consensus réel ne se dégage sur les structures de ces données. En l'absence de standard, chaque équipe adopte une solution *ad hoc* empêchant ainsi le partage de code au sein de la communauté rendant difficile les évaluations comparatives entre systèmes.

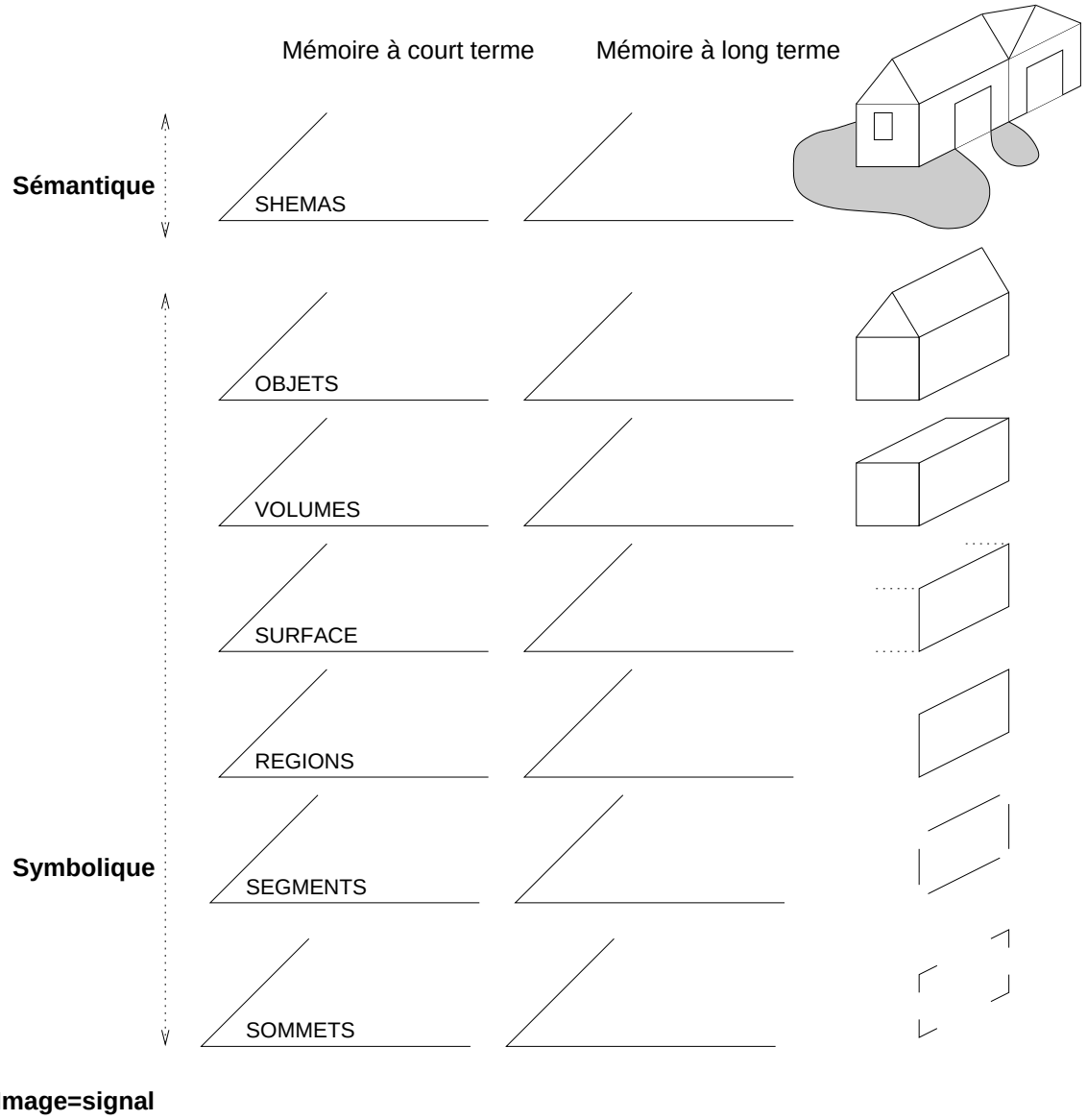


Fig. 4.3 : *Les niveaux de description selon Hanson & Riseman d'après [Han78].*

4.3.2 Connaissances descriptives sémantiques

Ce niveau propose une représentation **déclarative** des scènes et des objets que le système doit identifier dans l'image. C'est à ce niveau que les entités détectées acquièrent leur **sémantique**, c'est donc une connaissance *a priori* fortement **liée au domaine** d'application. On retrouve fréquemment le vocable d'hypothèses à propos de ces données à cause du caractère incertain qui leur est souvent associé.

Dans de nombreuses architectures [San95, Mat90, Dra89], ce niveau se présente comme un réseau hiérarchique des concepts (objet, scène) du monde observable.

Chaque sommet de réseau représente un ensemble de connaissances nécessaires à son identification dans l'image. Ces connaissances sont de deux natures :

1. Les connaissances descriptives représentent un prototype de l'objet observable. Elles peuvent être de deux types différents :
   (a) On utilise des attributs lorsque ces connaissances sont directement mesurables, comme c'est le cas pour les objets[1] non décomposables en sous-parties, on utilise des attributs se référant à la photométrie ou la géométrie de l'objet.
   (b) On utilise des liens vers d'autres sommets pour représenter les diverses relations spatiales existant entre objets.

   Il faut distinguer les concepts observés ou détectés qui sont des instances d'une certaine classe de concepts et les classes de concepts elles-mêmes. Les instances sont présentes en STM tandis que les classes représentent des prototypes dans la mémoire à long terme.
2. Les connaissances opératoires représentent les stratégies d'identification de la classe d'objets que le sommet représente. Nous reviendrons plus en détail sur ce point en §4.5.

Les Schémas de VISIONS [Han78] et Schema System [Dra89] détaillés à la section (§5.5.2.1 p. 98) illustrent bien notre propos.

Le Système SIGMA [Mat90] procède de façon similaire, chaque concept étant représenté par une classe dont le formalisme relève à la fois des classes au sens des langages objet et des *frames*. Cette classe contient les connaissances descriptives et opératoires nécessaires à la détection de l'objet.

Les liens entre sommets représentent les différents types de relations pouvant exister entre les concepts.

1. **Les liens de spécialisation** entre entités traduisent des relations *sorte-de* (*is-a* en anglais). Par exemple, un sapin est une sorte d'arbre.
2. **Les liens de composition** (*part-of* en anglais) entre entités traduisent la décomposition hiérarchique d'une entité en ses sous-parties (voir fig. 4.4).
3. **Les liens conflictuels** traduisent la non-consistence de l'existence simultanée et voisine de deux concepts.
4. **Les liens de vues** traduisent les relations entre un objet réel et son aspect dans une image. Par exemple un cylindre vu de face se voit comme un rectangle sur l'image.

[1] Le terme objet ou concept dénote seulement le caractère sémantique de l'entité, il se peut que l'objet considéré ne soit qu'une sous-partie d'un objet au sens commun. Par exemple : dossier de chaise.

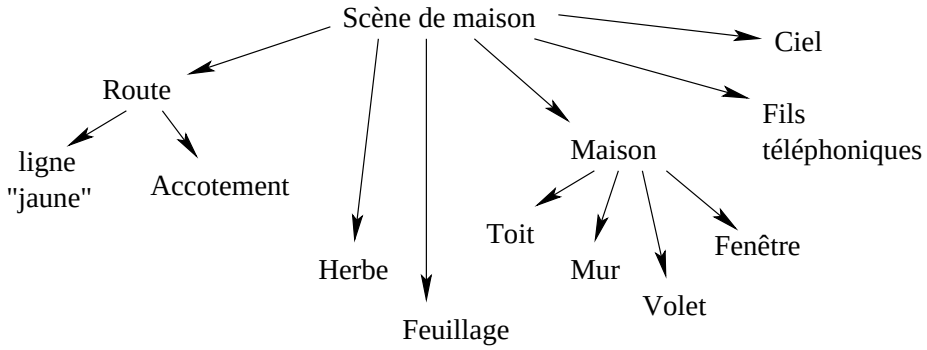


FIG. 4.4 : *Exemple de liens de compositions (part-of) entre entités (Schémas) tiré de [Dra89].*

5. **Les liens d'instanciation** traduisent le processus de mise en correspondance du modèle et des observations. Lorsque le système pense avoir identifié une classe, il génère une hypothèse qui prend la forme d'une instance de la classe identifiée. Nous reviendrons (§4.5.3 p. 70) sur la génération d'hypothèses.

En plus d'ancrer les concepts dans l'image, ces liens expriment des critères de cohérence spatiale, comme par exemple les liens reliant un arbre à son tronc et feuillage. En effet, ces liens permettent la **propagation** de deux notions duales :

1. **Les contraintes** de non compatibilité entre hypothèses peuvent être explicitement codées comme dans SIGMA [Mat90] sous forme d'arcs exprimant un conflit entre deux hypothèses. Ces contraintes peuvent aussi être propagées à l'aide d'algorithmes de relaxation discrète permettant de réduire l'ensemble des étiquettes possibles associées à un noeud. Dans [Ros76] Rosenfeld illustre cette approche de relaxation sur les étiquettes possibles à la reconnaissance d'objets. Tenenbaum et Barrow [Ten77] utilisent une approche similaire de propagation de contraintes sur les étiquettes s'appliquant dès la segmentation. Cette méthode permet d'illustrer la manière dont le modèle (la connaissance *a priori*) peut s'appliquer dès les étapes précoces de la vision.
2. **La vraisemblance** affectée à une hypothèse peut permettre de renforcer la vraisemblance des hypothèses voisines. Ce problème de **fusion de données** est traité dans VISIONS [Han78] par une approche Bayésienne à l'aide de probabilités conditionnelles liant la vraisemblance d'un objet par rapport à ses sous-parties. Exemple : ayant détecté une hypothèse `roue` avec une certaine vraisemblance, la connaissance *a priori* de la probabilité conditionnelle P(`voiture`|`roue`) va permettre de propager la vraisemblance de `roue` vers la vraisemblance de `voiture`.

Le modèle doit satisfaire plusieurs contraintes afin de faciliter **l'ingénierie des connaissances** :

1. **la distribution** nécessaire des connaissances permet une construction modulaire et incrémentale ainsi que l'intégration de connaissances très hétérogènes ;
2. **la déclarativité** de ces connaissances est nécessaire afin d'en autoriser une évaluation *a priori* .

Ces liens permettent aussi, comme nous le verrons plus tard, de guider la stratégie d'interprétation du système. À la section 4.6 p. 72, nous préciserons des outils comme les réseaux sémantiques ou les frames souvent utilisés pour la mise en œuvre de ces structures.

4.4 Les tâches composant la théorie computationnelle

Nous allons faire une brève description de l'ensemble des procédures visuelles intervenant dans un système de vision. Ces tâches représentent une connaissance calculatoire généralement procédurale. Ces procédures transforment les informations composant la théorie computationnelle. Ces transformations permettent d'obtenir d'autre données d'un niveau de représentation supérieur ou égal, voire parfois inférieur.

On distingue les différentes procédures visuelles sur la nature des données traitées et sur la quantité de connaissances *a priori* des scènes à traiter qu'elles contiennent.

On observe donc dans la plupart des architectures un ensemble de procédures allant de la vision bas-niveau jusqu'à parfois la vision haut-niveau. Cet ensemble de procédures assure une transformation progressive des données permettant de franchir le fossé quantitatif et qualitatif séparant les représentations bas-niveau des représentations haut-niveau.

4.4.1 Les prétraitements

Les procédures de prétraitement opèrent en amont de la chaîne de vision lors de cette étape. Elles visent à améliorer la qualité d'une image dégradée par l'environnement (par exemple, voile bleu en imagerie satellitaire) ou par la chaîne d'acquisition. Dans ce domaine réservé aux traiteurs de signaux, on cherche à produire de nouvelles images restaurées ou rehaussées en essayant de trouver un bon compromis permettant une amélioration de l'homogénéité des régions sans dégrader la netteté les transitions.

Cette étape, souvent négligée comme peut l'être l'acquisition, doit être prise en compte dans le cadre d'une analyse systémique, de nouvelles contraintes pouvant émerger de l'interaction entre procédures de prétraitement et les procédures en aval de la chaîne de vision. En effet, comme cela est souligné dans [Coc95], l'enchaînement et le paramétrage des opérateurs doivent être contraints par des connaissances *a priori* du contenu de l'image.

4.4.2 Les tâches d'analyse d'image

Ces tâches opèrent au niveau signal, c'est-à-dire l'image, afin d'en extraire une description symbolique intermédiaire : les indices visuels ou primitives d'images. On trouve ici principalement les techniques de segmentation (voir chap. 2) et d'estimation de mouvements. Historiquement, on tend à considérer les tâches de bas-niveau

comme indépendantes vis-à-vis du type d´image traitée ; les caractéristiques à extraire sont intrinsèques à l'image.

Ce dernier point explique en partie pourquoi la connaissance à ce niveau y est essentiellement procédurale, les systèmes à base de connaissances pour la vision bas-niveau étant relativement rares. Citons tout de même le système de Nazif et Levine [Naz84].

4.4.3 Les tâches de niveau intermédiaire

Les opérateurs de bas-niveau génèrent des primitives incomplètes : des bouts de segments non reliés pour une ligne existante ou une sur-segmentation, c'est-à-dire plusieurs régions pour une zone "homogène". Les primitives peuvent aussi être fausses : une chaîne de points de contours ayant pris un mauvais "aiguillage" dû à une évaluation locale perturbée par le bruit (voir les procédures de suivi (§2.3.3 p. 24) et fermeture (§A.3 p. 255) de contour). Les régions peuvent être sous-segmentées recouvrant ainsi plusieurs régions réelles dans l'image.

Les acteurs du domaine s'accordent à dire [Dra89, Cre97] que le problème récurrent rencontré à cette étape de la vision est l'explosion combinatoire des groupements possibles entre primitives. Les solutions proposées sont fondées sur l'utilisation massive de connaissances et de contraintes afin de lutter contre cette combinatoire, et lever les ambiguïtés. On distingue cependant deux approches :

1. Les approches basées sur un "modèle fort" (*strong model*) [Boy00, p. 192] utilisent au plus tôt une connaissance très pointue du domaine d'application afin de contraindre et guider les regroupements. Sous cet aspect, l'étape intermédiaire est orientée "reconnaissance", et produit donc des résultats à caractère sémantique correspondant aux objets de la scène.
2. Les approches basées sur un "modèle faible" (*weak model*) utilisent une connaissance plus générale sur l'apparence du monde visible pour contraindre les regroupements. Les résultats obtenus conservent souvent leur caractère *syntaxique* sans qu'il leur soit associé de sémantique particulière.

Pour les raisons que nous venons d'évoquer, le champ d'action de l'étape intermédiaire de la vision est assez mal délimité. Néanmoins, nous allons aborder les différentes natures de tâches intervenant à ce niveau de résolution.

L'organisation perceptuelle [Low85, Boy00] regroupe un ensemble de traitements cherchant à organiser les primitives sur les critères de psychologie perceptive énoncés par les psychologues de la Gestalt ("totalité" en allemand). Cette école allemande des années 20 soutient que la perception que l'humain a du monde n'est pas une somme d'éléments séparés, mais qu'elle procède par une organisation de formes globales qui donne un sens à ce que nous voyons. Cette approche décompose la perception en trois étapes [Dor99, p. 175] : une première étape précoce effectue un

codage séparé et non attentif des différents stimulis visuels ; la deuxième étape regroupe les informations en structures plus globales suivant trois critères ; la troisième étape est proprement cognitive, elle identifie les objets structurés par les deux étapes précédentes. Le fait que le carré illusoire de la figure de Kanizsa (fig. 4.5) soit perçu, suggère qu'il parvient à la conscience comme un produit fini (une totalité) et donc que l'attention ou le traitement guidé par des informations *a priori* (c'est-à-dire le modèle) n'intervient qu'ultérieurement.

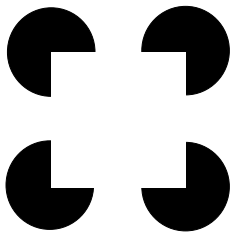


Fig. 4.5 : *La figure de Kanizsa : le carré illusoire.*

Le groupement perceptuel des primitives s'effectue suivant des règles perceptuelles dont voici les trois plus courantes :

1. la règle de proximité qui tend à regrouper les primitives proches entre elles ;
2. la règle de ressemblance qui tend à regrouper les primitives qui se ressemblent ;
3. la règle de clôture qui tend à combler les éléments manquants d'une forme familière.

La reconnaissance d'objets cherche à identifier dans l'image des motifs (*pattern*) connus qui sont stockés dans la mémoire à long terme. Cette étape connue sous le vocable de reconnaissance de forme (*pattern recognition*) est un point essentiel et délicat du processus de vision auquel est consacré une abondante littérature [Leb95, Thi97, Kun00, Bel92, The98].

Les approches pour traiter ce problème sont très variées :

- **Les méthodes statistiques** [Leb95, Kun00, The98] supposent que les observations suivent des lois. Dans l'espace des paramètres de ces lois, on cherche à séparer au mieux les différentes classes possibles. Nous abordons ce problème à la section 2.2.2 p. 19, à propos des méthodes de segmentation par classification.
- **Les méthodes stochastiques** fournissent des modèles adaptés à la grande variabilité des formes, les distances sont remplacées par des probabilités. On trouve entre autres les champs de Markov cachés (*Hidden Markov Model* HMM) et les champs de Markov aléatoires (*Markov Random Fields* MRF) présentés à la section A.1.4 p. 253.
- **Les méthodes neuronales** [Thi97, The98, Kun00] sont à comparer aux approches statistiques non paramétriques. En effet, on les utilise lorsque l'on ne peut modéliser les lois que suivent les observations. On distingue plusieurs familles de classifieurs neuronaux, citons :

1. Les classifieurs supervisés non-linéaires dont les perceptrons multi-couches (PMC) et les réseaux à fonctions radiales (RFR) sont de bons représentants.
2. Les classifieurs non supervisés, citons les cartes topologiques, qui sont à comparer avec l'algorithme des nuées dynamiques.

– **Les méthodes structurelles** [Kun00, Bel92] fournissent une modélisation explicite de la connaissance qui peut donc être évaluée *a priori* et qui peut fournir *a posteriori* une explication de l'échec ou la réussite de la procédure. Dans ces méthodes, le modèle indique la structure que doit respecter une combinaison de symboles (primitives) pour être reconnue. Ce modèle prend généralement deux formes :

1. Une structure de **graphe** qui sera confrontée à la structure des primitives observées à l'aide d'algorithmes d'isomorphisme de graphes ou de sous-graphes [Bel92].
2. Une **grammaire** des formes possibles des objets que l'on souhaite reconnaître [Kun00].

– **Les systèmes à base de connaissance** utilisés dans *Schema System* [Dra89] et SIGMA [Mat90] sont à comparer avec les méthodes structurelles par leurs aspects déclaratifs. La mise en œuvre relève des stratégies d'interprétation traitées (§4.5 p. 68). Ces méthodes sont d'ailleurs souvent utilisées comme outils d'intégration et de contrôle des autres méthodes précédemment citées. Nous reviendrons sur les techniques utilisées (§4.6 p. 72).

4.4.4 Conclusion

Comme le précisent les concepteurs de VISIONS [Han78, p. 304] ainsi que Boissier dans [Boi93], un système de vision doit permettre une intégration harmonieuse et flexible d'un grand nombre de procédures visuelles très hétérogènes. La modularité entre ces procédures doit être préservée afin de permettre un développement séparé sans affecter les performances du système. C'est pour cette raison que de nombreux chercheurs ont opté pour une architecture à base de tableaux noirs (*blackboard*). Ainsi, on retrouve souvent les vocables de *spécialistes, sources de connaissances* (*Knowledge Sources KS)* ou même *tâches* pour qualifier ces procédures.

Le moment où la connaissance *a priori* (c'est-à-dire un traitement guidé par le modèle) doit être prise en compte est un problème essentiel qui divise la communauté. Ce problème fait écho au débat vision reconstructionniste / vision guidée par les buts. Certains pensent que la vision bas-niveau et intermédiaire doit être guidée par les données afin de fournir des opérateurs de traitement génériques et robustes face à un environnement imprévisible (donc difficilement modélisable). D'autres chercheurs considèrent que le modèle doit être utilisé au plus tôt afin de répondre efficacement à un problème précis.

Une solution médiane consiste à introduire le modèle via un séquencement et un paramétrage particulier d'opérateurs génériques. C'est le rôle des outils de planification d'opérateurs que nous abordons à la section 4.5.5 p. 72.

4.5 Les stratégies de navigation parmi les éléments composant la théorie computationnelle

Lors des sections précédentes, nous avons abordé les différents composants constituant la théorie computationnelle, à savoir : des données structurées suivant les niveaux de représentation et des procédures visuelles transformant ces données à travers les niveaux.

Nous abordons maintenant le problème de la stratégie de navigation dans les éléments composant la théorie computationnelle afin d'identifier les scènes ou objets recherchés.

Nous allons présenter les différents modes ascendants, descendants, mixtes et hétérarchiques ainsi que le paradigme de génération/ vérification d'hypothèses et de résolution de conflits entre hypothèses. Ce type d'approche étant orienté "concepts" (primitives, hypothèses) nous aborderons finalement l'approche duale orientée "planification de tâches".

4.5.1 Stratégies ascendantes & descendantes

Les stratégies contrôlent et paramètrent les procédures visuelles afin de reconnaître un objet ou une scène. Suivant les systèmes et les concepts à reconnaître, elles varient autour de quatre modes différents :

1. Les stratégies ascendantes (*bottom-up*) procèdent par des transformations progressives, via les procédures visuelles, des données à travers les niveaux de représentations. Cette approche purement reconstructionniste, **guidée par les données** (*data-driven*), permet de faire émerger un grand nombre de représentations intermédiaires générales.

2. Les stratégies descendantes (*top-down*) sont **guidées par les buts** du système. On cherche à détecter des concepts particuliers dont le modèle permet de contraindre l'extraction des informations. Dans cette approche **guidée par le modèle** (*model-driven*), les procédures visuelles sont utilisées en mode **requête** décrivant précisément des centres d'intérêts.

Des systèmes comme BORG, OCAPI, LLVE de SIGMA ou GOLDIE de Schema System (§4.5.5 p. 72) permettent de traduire ces requêtes en un plan de séquencement d'opérateurs avec un paramétrage adapté.

Comme le suggèrent les tenants de la vision guidée par les buts, une reconstruction exhaustive de la scène est rarement nécessaire. En effet, le système de vision peut être contraint par les objectifs particuliers du système plus général qui l'utilise. Par exemple : un robot se déplaçant dans un environnement, donne à son système de perception des buts précis qui permettent de **focaliser** et donc de contraindre l'attention sur des régions ou concepts d'intérêt.

3. Les stratégies mixtes permettent de combiner les avantages des deux approches : lorsque l'on découvre une scène sans *a priori* sur celle-ci, une approche guidée par les données permet de faire émerger des caractéristiques potentiellement intéressantes. Les primitives et hypothèses générées permettent d'activer certains concepts du modèle déclenchant une stratégie attentive (c'est-à-dire contraignante), guidée par le modèle, et qui va tâcher de vérifier les hypothèses et d'en découvrir de nouvelles. Certains auteurs dont Rao et Jain [Rao88] font référence à la notion de cycle de perception pour qualifier cette alternance des deux modes.

4. Les stratégies hétérarchiques se distinguent des approches précédentes en rompant avec l'aspect hiérarchique : ascendant, descendant ou alterné. Dans ces approches centrées "concepts", le système contrôle et focalise son attention sur un concept puis déclenche une stratégie qui lui est adaptée. La stratégie va utiliser les arcs liés au concept pour vérifier l'hypothèse ou en générer de nouvelles. Suivant la nature des arcs, cette stratégie peut mêler de façon opportuniste approche ascendante ou approche descendante.

4.5.2 La focalisation

La focalisation est le point clé du contrôle dans un système de vision. Elle permet de concentrer l'attention du système sur les éléments les plus pertinents composant la théorie computationnelle et ainsi de contenir la combinatoire de la recherche dans cet espace. On distingue plusieurs types de focalisation :

1. La focalisation spatiale (R.O.I. *Region Of Interest*) permet de restreindre des traitements à une zone de l'image. C'est une focalisation perceptive concentrant l'attention du système sur **le où**. Elle intervient la plupart du temps dans une démarche descendante lorsque l'on cherche à vérifier ou détecter une hypothèse. Dans SIGMA [Mat90], lorsqu'une hypothèse, dont on prédit l'existence par une approche descendante, n'a pas encore été détectée, une requête de traitement (par ex. re-segmentation) contrainte par cette R.O.I est transmise au module LLVE par le module MSE. Le système de Nazif & Levine [Naz84] illustre aussi cette approche à l'aide de méta-règles sélectionnant des zones, focalisant l'attention des autres règles sur les régions les plus riches en information.

2. La focalisation sur un modèle (M.O.I. *Model Of Interest*) consiste à choisir suivant l'état du système, la partie du modèle à développer.

Cette focalisation concentrant l'attention du système sur **le quoi** est très polymorphe :

1. Le choix d'une hypothèse à développer parmi toutes celles qui sont existantes dans le système. Il est généralement basé sur une heuristique : par exemple dans MESSIE II [San95, p. 78], on choisit les hypothèses les plus faciles à traiter. En effet, un grand objet avec une photométrie bien identifiée posera moins de problèmes qu'un petit objet texturé.

2. La focalisation peut consister à filtrer un ensemble de primitives sur un critère de vraisemblance avec des types d'objets recherchés. C'est l'autre rôle du module MSE (Model Selection Expert) de SIGMA [Mat90]. Par ce procédé, on évite la prolifération d'hypothèses peu vraisemblables en focalisant l'activité du système sur les plus vraisemblables.
3. La focalisation, comme dans VISIONS [Han78], peut aussi utiliser la vraisemblance et les probabilités conditionnelles reliant les concepts. Ayant une hypothèse traduisant un concept, la stratégie pourra alors chercher à détecter les sous-parties du concept les plus importantes (approche descendante). Elle pourra aussi renforcer la vraisemblance d'un concept dont l'hypothèse courante est une partie (approche ascendante).

4.5.3 Génération et vérification d'hypothèses

Le paradigme génération et vérification d'hypothèses se retrouve à quelques variantes près dans tous les systèmes de vision. Ce paradigme prend essentiellement deux formes :

1. La prédiction d'hypothèses. Elle a lieu en mode descendant : si l'on dispose d'une hypothèse se décomposant en sous-parties, on va générer (prédire) des hypothèses reflétant l'existence de ces sous-parties.

a. *Génération par prédiction :*
Dans Schema System [Dra89], supposons qu'une instance du schéma `Maison` (fig. 4.4, p. 63) cherche à se reconnaître dans l'image. Le schéma va pour cela mettre en œuvre une stratégie spécifique à la détection des objets de type maison en invoquant, par instanciation, les sous-schémas constitutifs c'est-à-dire `Toit, Mur, Volet, Fenêtre`. L'ordre dans lequel ces invocations sont faites relève de la focalisation. Ces instanciations traduisent un mode prédictif de génération d'hypothèses.

b. *Vérification :*
La vérification va donc consister dans un premier temps à rétablir une attache aux données en cherchant à détecter cette hypothèse. Puis dans un deuxième temps, elle testera la consistance spatiale de l'hypothèse générée avec ses voisines.
Pour cela, le schéma `Toit` va mettre en œuvre une stratégie spécifique qui consistera à retrouver (car le schéma `Toit` n'est plus décomposable en sous-schémas) dans la base de données ISR, des primitives correspondant à ses connaissances descriptives. Si de telles primitives sont trouvées, le schéma `Maison` devra vérifier la consistance spatiale de la nouvelle hypothèse `Toit` avec ses voisines.
SIGMA [Mat90] procède de manière analogue en adressant au module MSE (Model Selection Expert) une requête accompagnée de contraintes de focalisation sur le type recherché et la région. Le module MSE transmet une requête au LLVE (Low-Level Vision Expert) qui va construire un plan d'opérateurs adapté à l'extraction de certaines primitives. Si la requête réussit, lors du prochain cycle d'interprétation, le module GRE Geometric Reasonning Expert

tâchera de vérifier la consistance spatiale de la nouvelle hypothèse.

2. L'identification d'hypothèses. Elle a lieu en mode ascendant : si l'on dispose d'un ensemble de primitives, le système va tâcher d'identifier les concepts qu'elles représentent.

a. *Génération par identification :*
Supposons que le système Schéma n'ait pas de but *a priori* , auquel cas il doit d'abord générer un ensemble d'hypothèses initiales à l'aide de la tâche IHS (Initial Hypothesis System) qui va attribuer à chaque région d'une présegmentation un degré d'appartenance aux différentes classes. Si l'une des régions a une vraisemblance suffisante pour appartenir à la classe `Toit`, une hypothèse `Toit` est générée en instanciant un schéma du même type.

b. *Vérification :*
Cette phase va consister à confirmer cette identification en vérifiant les contraintes relationnelles entre hypothèses (vérification de la consistance mutuelle, propagation de vraisemblance, etc.).
La classe de schéma `Maison` surveille à l'aide d'un démon l'apparition de l'hypothèse `Toit`. L'apparition de l'hypothèse `Toit` déclenche la création d'une hypothèse `Maison` (génération ascendante d'hypothèse), laquelle met en œuvre une stratégie de génération prédictive d'hypothèse (descendante) en générant les hypothèses `Mur, Volet, Fenêtre`. Puis `Maison` va vérifier la consistance de `Toit` avec les hypothèses voisines.

On voit bien une alternance opportuniste des modes ascendants et descendants dans la génération d'hypothèses caractérisant les stratégies hétérarchiques. En l'absence de connaissance *a priori* , le mode ascendant a un rôle d'amorçage du système qui, ensuite, doit s'appuyer le plus possible sur un modèle afin de contraindre par focalisation ses recherches.

4.5.4 Résolution de conflits

Un conflit apparaît lorsque le système donne deux interprétations (hypothèses) non compatibles pour une même zone de l'image. Le traitement du problème se décompose en deux parties : la détection du conflit et sa résolution.

La détection du conflit nécessite une collision entre les deux hypothèses. La mise en œuvre de cette collision est donc dépendante de l'architecture sous-jacente.

Dans le cas des architectures à mémoire partagée, comme les tableaux noirs, cette collision est facilement détectable en surveillant le tableau noir. Dans Schema System [Dra89], chaque hypothèse nouvellement générée est postée dans un tableau noir global, permettant ainsi de vérifier qu'il n'y a pas d'autre hypothèse conflictuelle.

Dans le cas des architectures à mémoire distribuée, comme les systèmes multi-agents n'utilisant que des envois de messages, la collision est détectée grâce à l'existence d'un lien entre agents pouvant générer des hypothèses conflictuelles. Nous proposons [Duc00] un algorithme distribué permettant de détecter les conflits à l'aide d'un réseau de liens reliant les agents potentiellement en conflit.

2. La résolution est un problème dont la complexité dépend du nombre d'hypothèses simultanément conflictuelles. Dans Schema System [Dra89], il ne peut y avoir que deux hypothèses conflictuelles, celle dont la vraisemblance est la plus forte est conservée.

Dans le cas d'une situation plus complexe où une multitude de conflits locaux apparaissent, nous proposons [Duc00] un algorithme basé sur le réseau de liens, évoqué ci-dessus, traduisant la topologie des conflits locaux à travers les agents (hypothèses). Un algorithme itératif distribué permet de résoudre le problème en utilisant *l'utilité* des hypothèses (§9.9 p. 202) comme critère de sélection. Cette utilité pouvant traduire la vraisemblance associée à une hypothèse, il n'y a pas de recours à une zone de mémoire centralisée ce qui permet d'éviter un goulet d'étranglement.

4.5.5 Planification de tâches

Nous avons abordé la navigation parmi les objets composant la théorie computationnelle à travers le problème de la construction de concepts. Nous allons brièvement aborder le problème dual d'une recherche dans l'espace des tâches. Le problème consiste à trouver un plan d'opérateurs, c'est-à-dire une séquence ordonnée d'opérateurs (tâches) correctement paramétrés afin de détecter des caractéristiques spécifiques dans l'image.

Un ensemble d'outils a ainsi émergé utilisant, pour construire leur plan, soit des systèmes à base de connaissances, soit des algorithmes de recherche de chemin à coût minimal dans un graphe d'opérateurs.

Exemples : L'expert de bas-niveau LLVE (Low-Level Vision Expert) du système SIGMA [Mat90, Mat89] permet de construire un plan d'extraction de primitive et de piloter l'exécution de ce plan à partir de requêtes exprimant des contraintes sur les primitives à extraire.

Citons aussi quelques outils remplissant des objectifs similaires : BORG (le système à base de connaissance) [Clo99] & PANDORE (la librairie d'opérateurs) [Pan01] du Greyc à Caen ; GOLDIE [Kho87] (utilisé dans Schema System) ; et finalement l'outil OCAPI proposé par Clément [Clé93] (INRIA).

Une autre approche [Cha91] consiste à utiliser les techniques de raisonnement à partir de cas (CBR Case-Based Reasonning) associant un plan à une situation donnée (le cas). Cette approche permet d'intégrer un mécanisme d'apprentissage.

4.6 Techniques de représentation de la connaissance

Après avoir abordé la justification et la nature (le pourquoi et le quoi) des différents composants de la théorie computationnelle, nous allons traiter les techniques permettant leur mise en œuvre c'est-à-dire **le comment**. Nous commencerons par évoquer les différentes techniques de représentation de la connaissance [Gia89, chap. 2], puis nous présenterons les architectures de contrôle.

La représentation de l'information est un souci majeur dans tous les domaines de l'informatique. Le souci initial était de permettre un traitement efficace des informations circulant au sein du système d'information. La complexité des systèmes grandissant avec la puissance des machines, la première préoccupation a progressivement laissé la place à une deuxième : fournir un formalisme facilitant l'ingénierie du système d'information.

4.6.1 Les différents formalismes

4.6.1.1 Le formalisme procédural

Dans le formalisme procédural, un programme est constitué de deux éléments distincts : les structures de données et les procédures. Le souci réside dans la modularisation des procédures de traitement facilitant l'ingénierie du système d'information. Cette tendance est parfaitement visible lorsque l'on observe l'historique des langages de programmation. Nous sommes passés des langages impératifs, regroupant les différents assembleurs, aux langages structurés assurant la modularité des traitements évitant, entre autre l'aspect "spaghetti" provoqués par les fameux "goto" ou "jump". Dans le formalisme procédural, les connaissances du domaine et la manière de traiter ces connaissances sont confondues au sein d'une même entité : la fonction ou la procédure.

4.6.1.2 Les langages objets

La préoccupation de modularité couplée à celle de disposer d'un langage dont le formalisme s'éloigne du matériel et se rapproche d'un mode de raisonnement humain, a permis la création des langages objets où données et traitements sont encapsulés dans une même entité conceptuelle : l'objet. L'objet fournit une palette d'outils conceptuels (héritage, polymorphisme, encapsulation) facilitant la maintenance et la compréhension des programmes. Ce dernier point est obtenu par une représentation éclatée et explicite des différentes entités intervenant dans le système d'information. La représentation explicite de la connaissance rejoint une préoccupation des chercheurs en intelligence artificielle que l'on peut ressentir dans la ressemblance qu'il y a entre langage objet et frames. Les formalismes objets et leur notion de spécialisation ont été utilisés dans des systèmes tels que SIGMA [Mat90] ou MESSIE II [San95].

4.6.1.3 Les formalismes déclaratifs

Les formalismes déclaratifs sont issus des tenants de l'approche cognitive [J.M94, p. 31] des chercheurs en intelligence artificielle. Elle met l'accent sur une représentation **explicite** de la connaissance s'apparentant ainsi à l'approche reconstructionniste des systèmes de vision. La parenté des deux approches se perçoit dans la critique faite par Rodney Brooks qui préconise, pour la vision, l'utilisation d'architectures réactives comme l'architecture de subsumption [Bro86] au détriment des systèmes de vision, disposant d'une représentation interne mise en œuvre à l'aide de systèmes à base de connaissances.

Remarquons au passage que la présence de chercheurs comme Marvin Minsky et Rodney Brooks dans l'IA et la vision illustre bien l'aspect fondamental posé par les problèmes de la vision et ses répercussions (frames, architecture de subsumption) sur l'IA.

Les points forts de la représentation déclarative sont :

- le découplage entre la connaissance du domaine que l'on peut trouver dans les règles et l'utilisation de ces connaissances qui se trouve dans le moteur d'inférence ;
- la déclarativité permet l'ajout "en vrac" de nouvelles connaissances ;
- la lisibilité permet de juger efficacement et *a priori* de la validité des connaissances ;
- elle facilite enfin la maintenabilité et donc l'ingénierie des connaissances.

Malheureusement les intentions des chercheurs du domaine ne se sont pas toutes concrétisées. En effet, le problème du contrôle des systèmes à base de règles, ou frames, s'avère délicat à résoudre et nécessite une bonne connaissance du moteur afin d'ajuster les priorités entre les règles. De plus, comme le précisent Alliot et Schiex dans [J.M94, p. 360], la notion de génie logiciel étant floue en IA, un système expert composé de milliers de règles dépasse les limites raisonnables de la maintenabilité. En fait, pour de tels systèmes, les systèmes experts deviennent plus délicats à maintenir que les approches classiques.

Nous allons maintenant présenter les trois approches déclaratives de représentation de la connaissance les plus couramment rencontrées en image.

4.6.1.4 Les règles de production

Les règles de production [Gia89] proposent un formalisme répandu de représentation déclarative de la connaissance. Dans ce formalisme, on représente l'expertise d'un domaine sous la forme de règles SI <condition> ALORS <action>. Cette base de règles forme la connaissance *a priori* ou mémoire à long terme du système qui s'applique sur la base de faits ou mémoire à court terme du système. Le moteur d'inférence se charge de l'application des règles en unifiant la partie condition d'une règle à un fait ou motif (*pattern matching*) de la base de faits, provoquant ainsi le déclenchement de la partie <action> qui éventuellement peut produire de nouveaux faits.

Ces systèmes peuvent être utilisés selon deux modes :

- le chaînage avant, ou raisonnement guidé par les données, propose de déduire tout ce qui est déductible à partir d'un ensemble de faits ;
- le chaînage arrière, ou raisonnement guidé par les buts, propose de tester la vérité d'un fait en essayant de trouver un chemin de déductions logiques jusqu'aux faits présents initialement (frontière axiomatique).

Le découplement –connaissances (règles) et utilisation des connaissances (moteur d'inférence)– pose un problème de contrôle lorsque plusieurs règles sont applicables en même temps ; on parle de l'ensemble des conflits ou agenda des règles déclenchables. Des systèmes comme OPS-5 ou CLIPS [Cli01] proposent alors différentes

stratégies de résolution basées sur des priorités associées aux règles ou des politiques de recherche en profondeur ou largeur d'abord.

Suetens et Oosterlink [Sue87] soulignent l'intérêt des systèmes experts en modélisation de connaissances dans la vision par ordinateur. Cependant, les règles sont le plus souvent employées pour modéliser :

- des connaissances de décisions, par exemple : les stratégies de SIGMA [Mat90] ;
- un savoir faire c'est-à-dire des connaissances sur les opérateurs afin de construire un plan d'exécution de tâches : BORG [Clo99].

Elles sont plus rarement employées pour modéliser une scène, des objets ou même un savoir perceptuel. Ce fait est en soi une information sur la capacité des règles à modéliser des objets d'aspect multiforme.

Citons tout de même le système de Nazif et Levine [Naz84] qui proposent une application à la segmentation d'image ou les règles modélisent une expertise perceptuelle des lois Gestalistes (voir groupements perceptuels 4.4.3 p. 65). Le système est composé de plusieurs modules ou spécialistes implémentés sous la forme de paquets de règles :

- un superviseur contrôlant et ordonnant le déclenchement des autres modules ;
- un module de focalisation composé de méta-règles permettant de concentrer l'attention des autres modules sur des zones riches de l'image (focalisation spatiale) ;
- un spécialiste des régions qui selon des critères perceptuels (similarité, proximité, fermeture) fusionne ou divise les régions en utilisant aussi des informations de contour ;

```
;; Exemple d'une règle de fusion de deux régions (règle 802)
IF (1) the region size is very low
   (2) the adjacency with another region is high
   (3) the difference in region feature 1 is not high
   (4) the difference in region feature 2 is not high
   (5) the difference in region feature 3 is not high
THEN merge the two regions
```

- un spécialiste des lignes qui selon des critères perceptuels (alignement, proximité, fermeture) fusionne, prolonge, efface les lignes en utilisant aussi des informations de région.

```
;; Exemple d'une règle de fusion de deux lignes (règle 1504)
IF (1) the line end point is open
   (2) the line gradient is not very low
   (3) the distance to the line in front is not very high
   (4) the two lines have the same region on the left
   (5) the two lines have the same region on the right
THEN join the lines by forward expansion
```

4.6.2 Incertitude et imprécision

La représentation de l'incertitude (par ex. : "je pense à 80% que Pierre mesure 1m85") et de l'imprécision (par ex. : "Pierre est grand") sont des problèmes récurrents en vision du fait du bruit, de la variabilité des observations et de la faible capacité à spécifier les modèles. Le deuxième enjeu est la fusion de données due à l'utilisation de plusieurs sources d'informations, ou de plusieurs traitements sur une même source. Une des préoccupations est de pouvoir intégrer un maximum d'informations avant de décider.

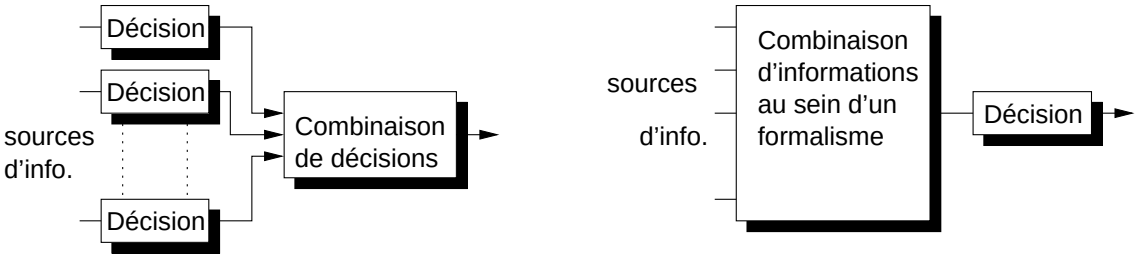


Fig. 4.6 : *Les deux approches de la fusion d'informations : fusion de décisions à gauche et fusion de données à droite.*

Remarque : la nécessité de disposer d'un formalisme de représentation pour combiner les informations suggère un certain niveau de reconstruction s'opposant ainsi partiellement aux approches réactives (Brooks, architecture de subsumption [Bro86]) qui utilisent plutôt la fusion de décision. Si l'argument d'une décision au plus tard intégrant un maximum d'informations est légitime, on doit prendre en compte les inconvénients majeurs des formalismes supportant cette fusion d'informations. Ces formalismes sont complexes, impliquant des difficultés de modélisation et un temps de calcul important. De plus, ils mènent parfois à des incohérences logiques comme le souligne Zadeh [Zad84] à propos de Dempster-Shafer.

4.6.2.1 Le modèle probabiliste

Le modèle probabiliste abordé dans le cadre Bayésien [Hat91, chap. 4] [Leb95] permet de modéliser l'incertitude portant sur les variables du domaine et leurs relations de causalité.

L'inférence Bayésienne utilise la règle de Bayes pour propager la certitude acquise d'observation O vers une hypothèse H.

$$P(H|O) = \frac{P(O|H)P(H)}{P(O)}$$

Par exemple : un médecin cherche une maladie H_i (hypothèse) observant un symptôme O (observation), l'expertise du médecin consiste à connaître le symptôme de chaque maladie c'est-à-dire $P(O|H_i)\forall i$; ainsi que la probabilité *a priori* de chaque

maladie : $P(H_i)\forall i$. Il cherche à inverser le phénomène c'est-à-dire à induire la maladie sachant le symptôme :

$$P(H_i|O) = \frac{P(O|H_i)P(H_i)}{P(O)} = \frac{P(O|H_i)P(H_i)}{\sum_{i=1,n} P(O|H_i)P(H_i)}$$

La combinaison d'évidences utilise la règle de Bayes pour déduire et fusionner les certitudes acquises par plusieurs sources d'observations $0_j j = 1, m$ vers une hypothèse H_i.

$$P(H_i|O_1 \wedge ...O_m) = \frac{P(O_1 \wedge ...O_n|H_i)P(H_i)}{P(O_1 \wedge ...O_m)}$$
$$P(H_i|O_1 \wedge ...O_m) = \frac{P(O_1 \wedge ...O_n|H_i)P(H_i)}{\sum_{i=1,n} P(O_1 \wedge ...O_m|H_i)P(H_i)}$$

S'il y a indépendance conditionnelle des observations O_j vis-à-vis des hypothèses H_i on obtient :

$$P(H_i|O_1 \wedge ...O_m) = \frac{P(O_1|H_i) * ...P(O_m|H_i)P(H_i)}{\sum_{i=1,n} P(O_1|H_i) * ...P(O_m|H_i)P(H_i)}$$

Replaçons nous dans le cadre médical où un médecin cherche une maladie H_i sachant qu'il observe m symptômes O_j. Les probabilités conditionnelles du terme de droite de l'égalité précédente sont par exemple P(fièvre|angine), P(inflammation|angine)... Les probabilités *a priori* $P(H_i)$ sont données par le médecin, par exemple P(angine), mais peuvent aussi être la certitude associée à une observation. La formule précédente sert de formalisme de combinaison d'informations où la décision intervient à la fin. On obtient le schéma de décision 4.7, similaire à la partie droite de la figure 4.6.

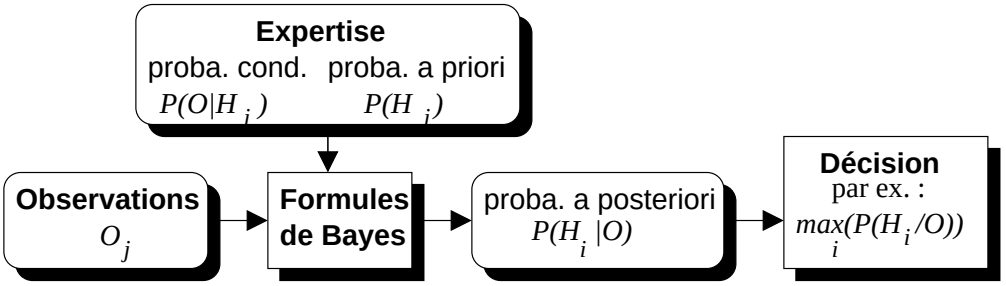


Fig. 4.7 : *Fusion d'informations et décision Bayésienne (inspiré de [Hat91]).*

Un réseau Bayésien [Bec99] [Rus95, chap. 15] se présente sous la forme de graphes de causalité où les sommets représentent les variables aléatoires, et les arcs pondérés par des probabilités conditionnelles représentent les relations de causalité entre V.A. Les algorithmes d'inférence propagent l'incertitude au sein de ce graphe. L'inférence peut suivre les liens de causalité mais elle peut aussi les remonter.

Ce modèle exige l'indépendance conditionnelle des observations, or les sources d'informations sont souvent en partie corrélées. De plus, ce modèle ne permet pas de représenter l'incertitude. Le système VISIONS [Han78] illustre bien l'utilisation du modèle probabiliste pour la propagation et la fusion d'informations.

4.6.2.2 Le modèle de Dempster-Shafer

Supposons qu'un système d'acquisition d'imagerie médicale renvoie une information sur la nature d'un tissu pouvant être un tissu sain S, une tumeur T ou une tumeur bénigne B. Le système nous indique qu'il a 40% de certitude que le tissu soit sain ($P(S) = 0.4$). Selon la théorie Bayésienne, cela implique que $P(\neg S) = 0.6$ auquel cas il faut tout de suite opérer le patient. Or ces 60% ne représentent pas une certitude de la présence d'une tumeur, ils représentent de l'ignorance, c'est-à-dire que le tissu peut être une tumeur ou du tissu sain.

Soit Θ l'univers des valeurs possibles mutuellement exclusives et $P(\Theta)$ l'ensemble des parties de Θ ; le modèle de Dempster-Shafer [Sha76] permet de modéliser une ignorance en n'imposant pas de distribuer les probabilités sur Θ. Dans ce modèle la connaissance acquise concerne les éléments de $P(\Theta)$ pouvant être des singletons (ex. $\{S\}$) ou des ensembles (ex. $\{T, B\}$). Cette connaissance se traduit par une masse de probabilité m distribuée à chacun de ces sous-ensembles. L'ignorance est rejetée non pas sur le complément de ces sous-ensembles mais sur Θ, incluant aussi les sous-ensembles auxquels on a déjà attribué une masse.
Par définition :

$$m(\emptyset) = 0$$
$$\sum_{A \subseteq \Theta} m(A) = 1$$

Ce qui donne dans notre exemple : $m(\{S\}) = 0.4$ et $m(\Theta) = m(\{S, B, T\}) = 0.6$

L'intervalle d'incertitude associé à une partie $A \subseteq P(\Theta)$ est borné par deux valeurs : la crédibilité $CR(A)$ et la plausibilité $PL(A)$. Cet intervalle va traduire l'ignorance sur A.

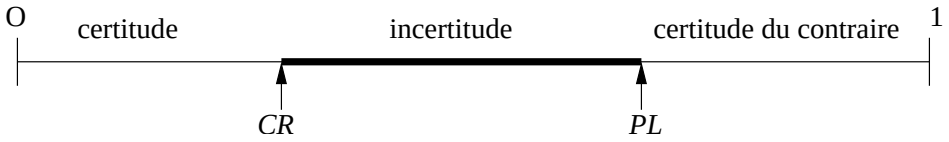


FIG. 4.8 : *La crédibilité et plausibilité encadrent l'incertitude.*

La crédibilité est la somme des masses des sous-ensembles de Θ impliquant A (au sens logique) ou contenus dans A (au sens ensembliste).

$$CR(A) = \sum_{B \in \subseteq A} m(B)$$

Par exemple, $CR(\{B,T\}) = m(B) + m(T) + m(\{B,T\}) = 0 + 0 + 0 = 0$.

La plausibilité est la somme des masses des sous-ensembles de Θ n'impliquant pas $\neg A$ (au sens logique) ou intersectant [2] A (au sens ensembliste).

$$PL(A) = \sum_{B:A\cap B\neq\emptyset} m(B)$$

Par exemple, $PL(\{B,T\}) = m(B) + m(T) + m(\{B,T\}) + m(\{S,B\}), m(\{S,T\}) + m(\{S,B,T\}) = 0+0+0+0+0+0.6 = 0.6$. La plausibilité de A peut aussi s'écrire :

$$PL(A) = 1 - CR(\neg A)$$

Il découle du calcul de la crédibilité et de la plausibilité les relations suivantes :
- $CR(A \vee B) \geq CR(A) + CR(B) - CR(A \wedge B)$
- $PL(A \wedge B) \leq PL(A) + PL(B) - CR(A \vee B)$
- $CR(A) + CR(\neg A) \leq 1$
- $PL(A) + PL(\neg A) \geq 1$
- $CR(A) \leq PL(A)$

L'analyse de l'intervalle $[CR(A), PL(A)]$ nous donne une indication sur le degré d'incertitude que l'on a sur A. Par exemple :
- $[0,1]$: rien n'implique A mais rien n'implique $\neg A \Rightarrow$ ignorance totale ;
- $[0,0]$: A est fausse ;
- $[1,1]$: A est vraie ;
- $[\lambda,\lambda]$ ($\lambda \in [0,1]$) : il n'y a pas d'incertitude, on retrouve le cas probabiliste où $P(A) = \lambda$;
- $[\lambda,1]$ on dispose d'une certaine certitude sur A ;
- $[0,\lambda]$ on dispose d'une certaine certitude sur $\neg A$.

La combinaison de masses de probabilité permet la fusion d'informations provenant de différentes sources. Cette combinaison va déplacer les masses de probabilité, de Θ vers les sous-ensembles d'hypothèses. Cela signifie que l'on déplace les masses de probabilité d'un ensemble contenant toutes les possibilités (ignorance) vers des sous-ensembles particuliers d'hypothèses. En conséquence, l'intervalle d'incertitude associé à ces sous-ensembles d'hypothèses va diminuer. L'information fournie par deux sources de connaissances indépendantes se traduit par des fonctions d'affectation de masses de probabilité m_1 et m_2. Soit C une partie de Θ, on cherche $m_{12}(C) = m_1 \oplus m_2(C)$ c'est-à-dire la masse de probabilité affectée à C, résultat de la combinaison (somme directe) des deux fonctions m_1 et m_2.

$$m_{12}(C) = m_1 \oplus m_2(C) = \frac{1}{K} \sum_{A_i\cap B_j=C} m_1(A_i) * m_2(B_j)$$

Où K la constante de normalisation vaut :

$$K = 1 - \sum_{A_i\cap B_j=\emptyset} m_1(A_i) * m_2(B_j)$$

[2] Si A est un singleton on peut remplacer intersectant par contenant.

Appliquons Dempster-Shafer à un notre cas de diagnostic médical :

- Le premier examen (par ex. un scanner) avait donné $(\{S\}, \Theta) = (0.4, 0.6)$. Soient les intervalles d'incertitude associés à la source 1 : $\{S\}[0.4, 1]_1$ et $\{B, T\}[0, 0.6]_1$.
- Les médecins décident de pratiquer un deuxième examen utilisant cette fois-ci l'IRM qui donne le résultat suivant : $(\{S\}, \{B, T\}, \Theta) = (0.7, 0.1, 0.2)$. Soient les intervalles d'incertitude associés à la source 2 : $\{S\}[0.7, 0.9]_2$ et $\{B, T\}[0.1, 0.3]_2$.

Combinons les deux sources : le tableau ci-dessous représente l'intersection des sous-ensembles associés au produit des masses de probabilité.

	$m_1(\{S\}) = 0.4$	$m_1(\Theta) = 0.6$
$m_2(\{S\}) = 0.7$	$\{S\} : 0.28$	$\{S\} : 0.42$
$m_2(\{B, T\}) = 0.1$	$\emptyset : 0.04$	$\{B, T\} : 0.06$
$m_2(\Theta) = 0.2$	$\{S\} : 0.08$	$\Theta : 0.12$

$K = 1 - 0.04 = 0.96$; $m_{12}(\{S\}) = 0.8125$; $m_{12}(\{B, T\}) = 0.0625$; $m_{12}(\Theta) = 0.125$;.
Soient les intervalles d'incertitude associés à la combinaison des deux sources :
$\{S\}[0.8125, 0.9375]_{12}$ et $\{B, T\}[0.0625, 0.1875]_{12}$.

On constate que la combinaison des sources a permis de réduire les intervalles d'incertitude par rapport aux intervalles obtenus indépendamment. Si le seuil décidant de l'opération est de 0.8, deux décisions indépendantes auraient forcé à opter pour l'opération, alors qu'une décision après fusion d'informations décide du contraire. Le calcul de ce seuil est un problème qui relève de la théorie de l'utilité [Rus95, chap. 16] qui consisterait dans le cas présent à quantifier les risques associés à l'opération.

Comme le souligne Zadeh [Zad84], ce modèle peut fournir des résultats inattendus. L'exemple qu'il donne est le suivant : deux médecins effectuent deux diagnostics différents

- médecin 1 : m_1(méningite) = 0.99, m_1(tumeur du cerveau) = 0.01
- médecin 2 : m_2(commotion) = 0.99, m_2(tumeur du cerveau) = 0.01

Remarquons que les médecins sont d'accord pour donner très peu de probabilité à la tumeur mais ils divergent fortement sur le diagnostic. La masse de probabilité combinée résultante allouée à la tumeur est 1, ce qui signifie que le patient selon ce modèle doit être opéré d'une maladie qu'aucun médecin ne pronostique.

Il existe de nombreuses applications de cette théorie, citons par exemple [Pro90] Hanson et Riseman qui ont aussi envisagé son utilisation dans VISIONS [Han78] pour choisir finalement, comme le précise Draper dans [Dra89, p. 217], une approche plus simple à base d'heuristiques.

La complexité calculatoire et de représentation d'un problème réel de cette approche, ses failles logiques et la contrainte d'indépendance des sources peuvent justifier l'utilisation, comme l'a fait Draper, d'heuristiques plus simples.

4.6.2.3 Le modèle des ensembles flous

Cette théorie proposée par Zadeh, fondée sur les ensembles flous [Zad65], permet de modéliser l'imprécision associée à une hypothèse. Une proposition imprécise est de la forme "Paul est grand", ce qui est différent d'une proposition incertaine mais précise de la forme : "il y a 80% de chance que Paul mesure 1m 82".

Soit T le domaine des valeurs possibles, et F un ensemble flou, F est défini par $\{(t, \mu_F(t)) : t \in T\}$ où $\mu_F(t) \in [0, 1]$ est la fonction caractéristique de l'ensemble flou traduisant un degré d'appartenance à T. Les opérations ensemblistes, d'intersection, d'union, ... entre ensembles flous définis permettent de construire un système de raisonnement flou.

L'expert va définir l'univers du discours composé de variables linguistiques, par exemple : `taille` ou `poids` pouvant prendre des valeurs comme `petit, moyen, grand`. A ces valeurs sont associés des ensembles flous s'appliquant sur le domaine numérique des valeurs possibles. L'univers du discours peut aussi être enrichi de modificateurs comme `très, peu` ... dont l'application sur un ensemble flou donne un autre ensemble flou dont la fonction caractéristique est modifiée.

On obtient ainsi un outil capable de modéliser une expertise dans un langage "haut-niveau" à caractère symbolique dont la formalisation sous-jacente fournit une transition douce entre les données numériques (quantitatives) et les données symboliques (qualitatives). De plus, le modèle est capable d'intégrer et de combiner l'imprécision.

La théorie des possibilités, [Dub87] issue de travaux sur les ensembles flous permet, quant à elle, de représenter l'incertitude. Similaire à la théorie de Dempster-Shafer, on y retrouve deux valeurs, la nécessité : $N(A)$ et la possibilité : $\Pi(A)$ bornant l'incertitude associée à une proposition A. Couplé à la logique floue, ce modèle, autorise une formalisation homogène de l'incertitude et de l'imprécision.

Chapitre

5

Architectures logicielles de contrôle et SMA

L'architecture logicielle de contrôle orchestre l'activité des composants actifs et leur fournit un moyen d'échanger et stocker les résultats de leurs activités. Pour choisir ou évaluer un outil permettant la mise en œuvre d'un système de perception, nous allons tâcher de dégager les différentes contraintes et propriétés que doit respecter l'architecture logicielle.

Ensuite, nous évoquerons les différents modèles proposés par l'intelligence artificielle distribuée en insistant sur les Systèmes Multi-Agents (SMA).

Nous finirons ce chapitre par une typologie des différents systèmes de vision à base de SMA.

5.1 Problématique, ou pourquoi l'IAD ?

Les systèmes de vision sont composés d'un grand nombre d'entités traduisant des connaissances hétérogènes, tant descriptives qu'opératoires. Il est donc nécessaire de fournir un cadre favorisant une intégration harmonieuse de ces composants que nous nommerons agents. L'immense majorité des systèmes de vision utilisent une des approches de l'IAD (Intelligence Artificielle Distribuée) comme architecture logicielle de contrôle ([Jol01, chap. 7]). Les justifications d'un tel choix sont nombreuses, elles s'articulent d'abord autour d'un souci **d'intégration** :

1. Préserver la modularité et l'ouverture. C'est une préoccupation d'ordre génie logiciel visant à contrôler la maintenabilité du système tout en garantissant la capacité d'améliorer ou d'ajouter de nouveaux agents, sans dégrader les performances du reste du système ou provoquer le besoin d'une réécriture massive du code.

2. Permettre la diversité des formalismes. Le mythe de Babel ayant aussi frappé les langages informatiques, les systèmes de vision sont des projets de grande envergure pouvant s'étendre dans le temps comme VISIONS ou impliquer de nombreuses équipes comme le projet européen VAP (Vision As a Process) [Cro89]. Ainsi, l'architecture sous-jacente, et plus particulièrement son aspect de partage d'informations, doit être suffisamment segmentée pour permettre une cohabitation des nombreux formalismes utilisés.

3. Flexibilité. Comme le souligne Boissier dans [Boi97], le contrôle doit être suffisamment flexible pour permettre, d'une part, une évolution en fonction des agents présents et, d'autre part, installer dynamiquement des boucles de contrôle entre eux. Nous allons détailler ce point sur deux aspects : la coopération et la coordination.

4. Focalisation et adaptation. L'éclatement des connaissances opératoires et descriptives favorise leur utilisation parcimonieuse et adaptée au contexte local. La focalisation et l'adaptation peuvent être spatiales : focalisation des agents sur des régions riches en informations [Liu99] ; ou adaptation des seuils et des paramètres en fonction d'un contexte local de l'image (comme nous le proposons). Une approche éclatée favorise également une focalisation sur les concepts à reconnaître comme le proposent les schémas de Draper [Dra89].

5. Robustesse. Comme l'ont montré les approches coopératives en segmentation d'image (§2.5 p. 29), une certaine robustesse peut être obtenue en fusionnant l'analyse de plusieurs agents.

6. Calcul distribué. L'aspect intrinsèquement distribué des agents est adapté à l'utilisation d'architectures matérielles parallèles pour les étapes bas-niveau de la vision comme pour les étapes haut-niveau [Dra89].

5.2 L'intelligence artificielle distribuée

L'Intelligence Artificielle Distribuée (IAD) traite de la compréhension et de la modélisation distribuée des traitements et des connaissances. Deux familles d'architectures ont vu le jour : les tableaux noirs et les systèmes multi-agents.

5.2.1 Les tableaux noirs

Les tableaux noirs [Erm80] ou *blackboards* permettent à différentes sources de connaissances (KS *Knowledge Sources*) (ou spécialistes) indépendantes de collaborer à la résolution d'un problème. Le tableau noir représente l'état de la résolution, il est aussi le lieu d'échange d'informations entre les différentes KS.

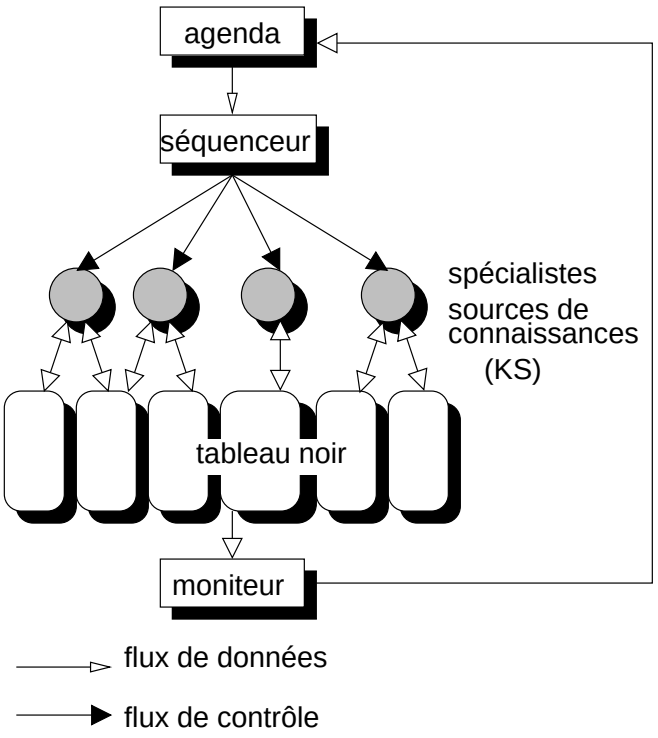


Fig. 5.1 : *Architecture générique d'un système à base de tableau noir.*

Le tableau noir est donc une mémoire partagée par tous les spécialistes. Ce tableau noir est généralement segmenté en différentes parties reflétant la nature hétérogène des données manipulées par les spécialistes. Bien qu'un spécialiste puisse accéder à toutes les régions du tableau noir, il n'est généralement concerné que par certaines d'entre elles.

Chaque spécialiste détient une partie de la connaissance opératoire nécessaire à la résolution du problème. Il peut être de nature procédurale ou déclarative, il poste et lit des messages dans le tableau noir sans savoir (en principe) qui a posté un message ou qui lira un message posté. Chaque spécialiste dispose d'un mécanisme

l'avertissant si un événement attendu, ou qui le concerne, se produit dans le tableau noir. Ce mécanisme appelé "démon", ou "condition", prend la forme d'un prédicat calculé sur l'état du tableau noir, et qui, s'il est vérifié, rendra le spécialiste exécutable en l'inscrivant dans l'agenda du moniteur. L'ensemble de ces conditions peut être regroupé sous la forme d'un moniteur de tableau noir.

Le séquenceur doit choisir à chaque cycle le ou les spécialistes à déclencher parmi ceux qui sont présents dans son agenda, suivant l'état du tableau noir. Différents types de contrôle existent :

- contrôle procédural basé sur un calcul dynamique de priorité (comme dans les systèmes d'exploitation) pour HEARSAY II[Erm80] ;
- contrôle hiérarchique simple comme dans CRISALYS [Ter83] ; ou amélioré ATOME [Lâa87] ;
- contrôle à base de tableau noir proposé par Barbara Hayes-Roth [HR85] utilise un tableau de contrôle pour piloter l'exécution des tâches du domaine. Le système BB-1 [HR88] en est une application.

Les systèmes de vision à base de tableaux noirs sont nombreux ; citons VISION [Han78] & Schema System [Dra89], SIGMA [Mat90] et MESSIE II [San95].

5.2.2 Les systèmes multi-agents

Les Systèmes Multi-Agents sont d'abord l'expression de préoccupations techniques : problèmes physiquement distribués, nécessité d'éclatement de la connaissance pour contrôler la maintenabilité et l'expression de connaissances hétérogènes, etc. Ils sont aussi au cœur d'un courant de pensée qui durant le siècle dernier a traversé tous les domaines de la recherche. Ce courant de pensée abordé section 4.2.3 p. 55, s'oppose à la vision analytique et réductionniste de la pensée Descartienne ; il suggère au contraire que seule une analyse globale d'un système peut rendre compte de sa complexité.

Qu'est qu'un système selon ce courant de pensée ? Pour répondre à cette question, nous citerons Norbert Elias [Eli91], sociologue, qui en 1939 donna une métaphore de la société sous la forme d'un système réticulaire ou réseau de fils entrecroisés :

> *"Un filet est fait de multiples fils reliés entre eux. Toutefois, ni l'ensemble de ce réseau ni la forme qu'y prend chacun des fils ne s'expliquent à partir d'un seul de ces fils, ni de tous les différents fils en eux-mêmes ; ils s'expliquent uniquement par leur association, leur relation entre eux. Cette relation crée un champ de forces dont l'ordre se communique à chacun de ses fils, et se communique de façon plus ou moins différente selon la position et la fonction de chaque fil dans l'ensemble du filet. La forme du filet se modifie lorsque se modifie la tension et la structure de l'ensemble du réseau. Et pourtant ce filet n'est rien d'autre que la réunion de différents fils ; en même temps chaque fil forme à l'intérieur de ce tout*

une unité en soi ; il y occupe une place particulière et y prend une forme spécifique"

Comment l'analyser ? Le fait de savoir si cette pensée complexe fournit des outils de compréhension des systèmes est hors de notre propos actuel. Néanmoins voici une piste simple d'analyse proposée par Marvin Minsky [Min88] :

" Il nous faut d'abord savoir comment fonctionne chaque partie séparée. Ensuite, nous devons connaître les interactions de chaque partie avec celles auxquelles elle est connectée. Enfin, nous devons comprendre comment toutes ces interactions locales se combinent pour exécuter ce que fait le système"

Qu'est qu'un agent ? Nous reprendrons la définition donnée par Ferber [Fer95] : on appelle agent une entité physique ou virtuelle ...

1. qui est capable d'agir dans un environnement,
2. qui peut communiquer directement avec d'autres agents,
3. qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser),
4. qui possède des ressources propres,
5. qui est capable de percevoir (mais de manière limitée) son environnement,
6. qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),
7. qui possède des compétences et offre des services,
8. qui peut éventuellement se reproduire, dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

Les tableaux noirs et les SMA divergent donc sur deux points : le contrôle centralisé pour les premiers, distribué pour les seconds, et le mode de communication respectivement anonyme et point à point. De ce fait, les spécialistes ne peuvent être réellement qualifiés d'agents sauf par abus de langage.

5.3 Types d'agents et structures de contrôle

5.3.1 Agents réactifs

Les problèmes dus à la complexité des agents utilisant des représentations internes symboliques ont poussé certains chercheurs, comme Brooks [Bro91], à privilégier le lien avec l'environnement considérant ce dernier comme le meilleur des modèles possibles, s'opposant ainsi à toute représentation interne de l'environnement. Dans les SMA réactifs, l'accent est mis sur l'aspect d'un comportement globalement émergent de l'interaction d'un grand nombre d'agents à fine granularité. Les agents

réactifs vivent sous "l'empire des sens", décidant des actions à mener uniquement en fonction de la perception qu'ils ont de leur environnement.

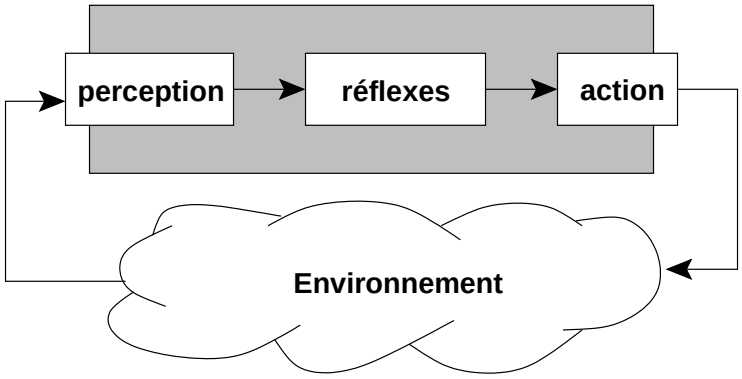


Fig. 5.2 : *Agent réactif.*

Brooks propose l'architecture de *subsumption* [Bro86] empilant hiérarchiquement un ensemble de comportements, chacun associant à une entrée sensorielle une action en sortie (il peut ne pas y avoir d'action). Plus on monte dans la hiérarchie des comportements, plus ils deviennent simples et vitaux. L'action en sortie d'un comportement va inhiber les actions en sortie des comportements des couches inférieures.

Cependant, comme le souligne Francisco Varela [Var98], l'amélioration de la boucle de rétroaction de l'agent et l'auto-organisation résultante sont des éléments indispensables à la viabilité de l'agent. Or cet apprentissage peut nécessiter un degré primitif de mémorisation, ce qui n'est pas en opposition avec l'approche réactive. Brooks dans [Bro91] stipule seulement que l'agent ne doit pas avoir de mécanisme interne de représentation. Ainsi la frontière avec la cognition est floue ; c'est pour cela que l'on emploie les termes de "purement réactif" ou "agent tropique" pour qualifier les agents sans mémoire.

L'approche réactive ou béhavioriste consistant à favoriser l'incarnation d'entités dans un environnement se retrouve dans les travaux de Ballard [Bal91].

5.3.2 Agents cognitifs

Les agents cognitifs ou délibératifs disposent d'une représentation interne du monde sur laquelle ils peuvent raisonner. Elle regroupe l'ensemble des **états mentaux** de l'agent. Un état mental ou *cognition* est une structure cognitive élémentaire. Les états mentaux ou attitudes intentionnelles peuvent être de natures différentes, nous allons en évoquer deux.

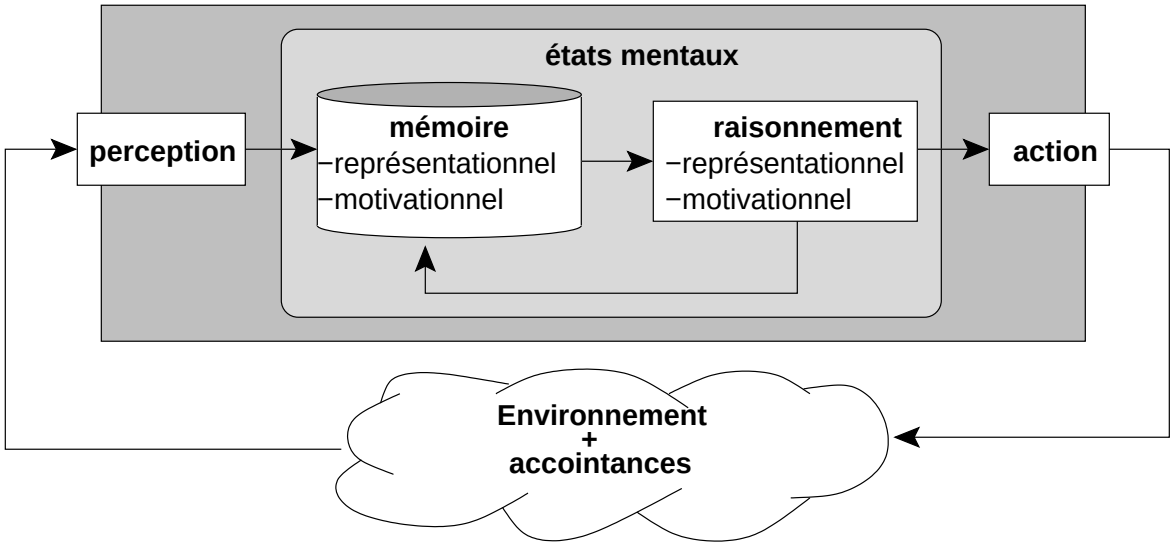


Fig. 5.3 : *Agent cognitif.*

5.3.2.1 États mentaux à caractère représentationnel

Les états mentaux représentationnels concernent les **croyances** que l'agent possède à propos du monde qui l'entoure. Elles peuvent porter sur l'agent lui même, son environnement et ses **accointances**, ces dernières étant l'ensemble des agents connus par cet agent.

L'agent n'étant pas omniscient, ses connaissances sont incomplètes et peuvent diverger de celles de ses voisins. En réalité, les connaissances de l'agent ne sont que des croyances locales. On modélise cet aspect des choses par des logiques modales ou la logique des défauts [Rei80] (prise en compte de l'incomplétude).

5.3.2.2 États mentaux à caractère motivationnel ou conatifs

Ces états mentaux représentent les différentes motivations de l'agent. Elles sont au cœur de la structure de **contrôle** de l'agent, permettant une sélection motivée des actions à mener. La présence ou l'absence de certaines attitudes, ou la capacité à les définir dynamiquement, vont conditionner la complexité de l'architecture de contrôle de l'agent.

1. **Les buts** de l'agent représentent sa raison d'exister. Peuvent-ils être définis dynamiquement par l'agent ou sont-ils fixés ?
2. **Le plan** d'exécution d'un but donné peut-il être adapté en fonction du contexte ou est-il prédéfini ?

L'intention est une attitude orientée vers l'action : elle va conduire l'agent à chercher les moyens et à mener des actions pour satisfaire cette intention. Si l'agent a pour but, entre autres, de satisfaire d'autres agents, il pourra prendre, si on le lui demande, des **engagements** vis-à-vis des demandeurs sur l'exécution d'une tâche. **Le désir** est une attitude voisine de l'intention. Cependant, sa durée de validité est

plus courte (par exemple, un seul cycle de raisonnement). Le désir par opposition à l'intention est une attitude changeante à court terme. De plus, le désir n'engage pas l'agent à l'action : un agent peut avoir un désir sans jamais le satisfaire ; mais sous certaines conditions le désir peut devenir intention.

5.3.3 Architectures de contrôle des agents cognitifs

Nous présentons trois architectures de contrôle des agents cognitifs. Elles vont de la plus simple, n'utilisant qu'un automate à états finis, jusqu'à des architectures plus complexes se basant sur les états mentaux à caractère motivationnel. Ces états sont, en effet, les éléments principaux de l'activité de contrôle de l'agent. Nous présentons deux modèles de contrôle pour ce type d'agents : la première (ASIC) illustre un type d'architectures en couche, et elle a des applications en système de vision *via* les systèmes de vision MAVI [Boi94b] et MAGIC [Boi94c] ; la deuxième, BDI, est assez représentative des agents délibératifs et connaît depuis quelques années une certaine popularité.

5.3.3.1 Automates à états finis

Les automates à états finis (ou AFD Automates Finis Déterministes) représentent l'approche la plus simple pour contrôler les actions de l'agent. L'automate est un graphe où les sommets représentent les états possibles S de l'agent. Un arc entre deux sommets s_i, s_j symbolise la transition entre ces deux états. Chaque transition est étiquetée par le couple `<événement,action>` représentant l'événement (parmi les événements possibles E) qui fait passer de l'état s_i à l'état s_j et l'action (parmi les actions possibles A) à mener lors de cette transition. L'automate de contrôle est donc défini par :

$$A =< E, S, A, \texttt{next}, \texttt{toDo}, s_o >$$

Où `next` est la fonction de transition de l'automate choisissant le prochain état en fonction de l'état courant et d'un événement :

$$\texttt{next} : S \times E \rightarrow S$$

Et `toDo` est la fonction d'activité choisissant l'action à entreprendre lors de cette transition :

$$\texttt{toDo} : S \times E \rightarrow A$$

Cycle agent AFD (événement)
- action ← `toDo`(état-courant, événement)
- état-courant ← `next`(état-courant,événement)
- exécuter(action)

Algo. 5.1 : *Un cycle d'exécution d'un agent contrôlé par un automate à états finis. Le paramètre "événement" correspond à la fonction de perception de l'agent, qui peut aussi être la réception d'un message.*

Pour un grand nombre d'états et d'actions possibles, l'automate peut rapidement devenir très complexe ; de plus, la prise en compte du parallélisme d'exécution y est délicate. L'utilisation des Réseaux de Pétri (RdP) peut s'avérer nécessaire.

5.3.3.2 ASIC

ASIC (Architecture for Social and Individual Control), proposée par Boissier [Boi93, Boi94a], se présente comme un modèle d'architecture de contrôle verticale en trois couches (fig. 5.4, p. 91) inspirée de l'automatique et des structures de contrôle hiérarchique. La couche décision définit l'objectif de l'agent. La couche adaptation adapte la loi de commande en fonction de l'évolution de l'environnement pour un objectif donné. La couche commande vise à déterminer la commande à appliquer au processus par application de la loi de commande précédemment définie à un ensemble d'observations en vue de satisfaire le but défini. Chaque couche contrôle la couche directement inférieure.

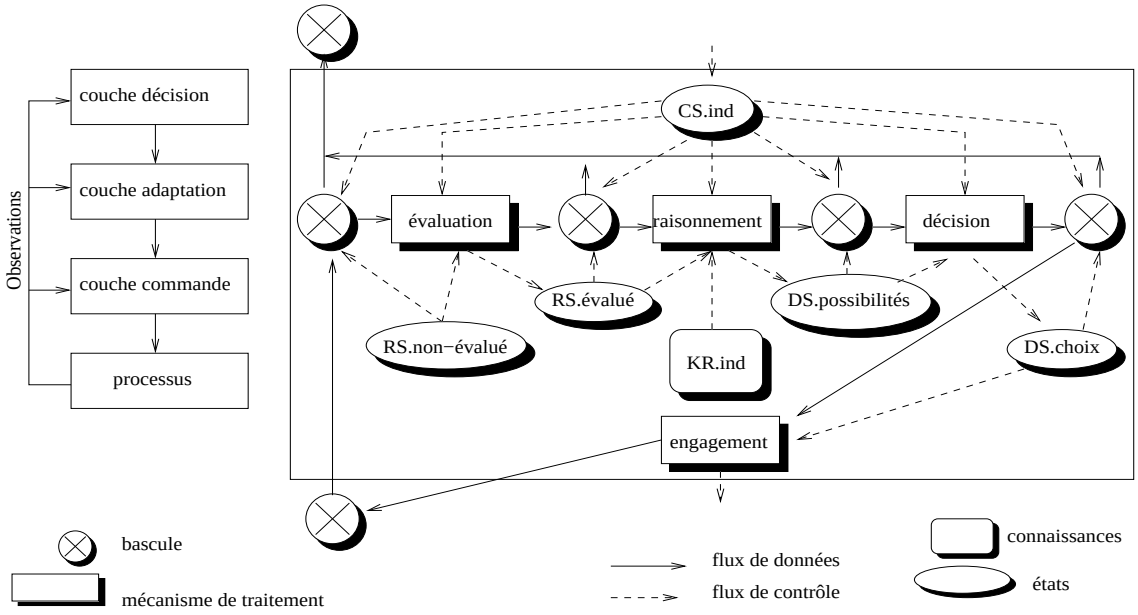


Fig. 5.4 : *À gauche : modèle en trois couches du contrôle dans ASIC ; à droite : architecture d'une couche de contrôle. D'après [Boi93].*

Les trois couches ne sont pas nécessairement présentes. Par exemple, un agent à but fixe n'aura pas de couche décision ; un agent n'adaptant pas sa manière de satisfaire ses buts selon le contexte n'aura pas de couche adaptation. Chaque couche respecte la décomposition horizontale classique perception→raisonnement→action. Ces différents modules agissent en fonction de l'état interne de l'agent, où RS est l'état de raisonnement du système, c'est-à-dire la mémoire représentationnelle et les plans et buts de l'agent, DS est l'état de décision regroupant les intentions possibles *(DS.possibilités)* et choisies *(DS.choix)*, CS représente l'état d'engagement individuel *(CS.ind)* ou social *(CS.soc)* (absent sur la figure). Enfin, dans KR on

trouve les connaissances de l'agent pouvant être individuelles *(KR.ind)* ou sociales *(KR.soc)*.

5.3.3.3 BDI

Les architectures BDI (*Beliefs Desires Intentions*) [Bra88, Rao91] articulent leur contrôle autour des croyances et des deux attitudes conatives que sont le désir et l'intention. Les désirs sont les options que l'agent souhaiterait réaliser au vu de ses croyances. Cependant, un agent agissant de la sorte serait très sensible au moindre stimulus provoquant du désir et rendant difficile voire impossible la réalisation d'objectifs à moyen ou long terme.

On dote donc l'agent d'un comportement délibératif consistant à choisir parmi ses désirs (ou options) ceux qu'il souhaite concrétiser. Les choix effectués et l'engagement de réalisation qui leur est associé sont des intentions [Coh90]. En plus d'être persistantes, les intentions guident directement l'agent à l'action. Les intentions une fois adoptées vont modifier l'activité motivationnelle de l'agent c'est-à-dire les deux fonctions "select-options" et "deliberate". En effet, un homme marié (en théorie) n'a pas le même comportement dans le choix de ses options et intentions qu'un célibataire.

Cycle agent BDI($Messages$)
- $\mathcal{B} \leftarrow$ beliefs-revision($\mathcal{B}$, $Messages$)
- $\mathcal{D} \leftarrow$ select-options($\mathcal{B}$, $\mathcal{I}$)
- $\mathcal{I} \leftarrow$ deliberate($\mathcal{D}$, $\mathcal{I}$)
- $\mathcal{P} \leftarrow$ planification($\mathcal{I}$)
- exécuter($\mathcal{P}$)

Algo. 5.2 : *Un cycle d'exécution d'un agent contrôlé par une architecture BDI.*

Néanmoins, l'agent ne doit pas être "trop têtu", en ce sens qu'il doit pouvoir abandonner une intention qu'une évolution de l'environnement a rendue obsolète. Cette rapidité de reconsidération codée dans la fonction "deliberate" doit être réglée en fonction du caractère que l'on souhaite donner aux agents et de la rapidité d'évolution de l'environnement.

5.4 Sociétés et organisations

Après avoir évoqué les différentes natures et structures d'agents, nous allons analyser la façon dont ils se constituent en société. L'organisation de la société s'évalue suivant plusieurs critères :

1. Y a-t-il des relations de type maître-esclave (ou contrôleur processus selon l'image de Boissier [Boi93]) ? Si oui, on parlera d'organisation **hiérarchique**, sinon elle est qualifiée d'**égalitaire**.

2. Les liens entre agents sont-ils fixes (figés au début de l'exécution du système) ou variables, évoluant selon les besoins ou les rencontres faites dans l'environnement ?
3. Les rôles respectifs (relations abstraites) entre agents sont-ils prédéfinis ou émergents ? Pour illustrer la différence avec le cas précédent, donnons un exemple : supposons que dans une entreprise un cadre C soit amené à travailler avec un employé E qu'il ne connaissait pas. Un lien s'est créé entre C et E, donc c'est une organisation variable. Cependant, les rôles respectifs (relations abstraites) étaient prédéfinis.

Les agents sont en relation dans l'objectif d'interagir. En effet selon Ferber [Fer95, p. 67] :

> *" Les interactions sont non seulement la conséquence d'actions effectuées par plusieurs agents en même temps, mais aussi l'élément nécessaire à la constitution d'organisations sociales."*

Nous allons examiner les deux formes principales d'interaction que sont la coopération et la coordination.

5.4.1 La coopération

L'architecture logicielle d'un système de vision doit permettre une mise en œuvre naturelle et efficace de mécanismes de coopération entre agents. L'identification de ces articulations dans lesquelles intervient une coopération entre agents permet de définir **l'organisation** structurant les rôles des différents agents. Boissier [Boi93] propose de chercher à identifier les couples contrôleur-processus au sein du système, ce qui revient à identifier les différentes coopérations entre agents.

Il est donc nécessaire au préalable de dégager la nature des différentes coopérations pouvant intervenir, afin de trouver une architecture adaptée à leur mise en œuvre . Selon Hoc [Hoc96], il y a trois types de coopérations :

1. **La coopération confrontative**
   Dans ce type de coopération, plusieurs activités concurrentes confrontent leurs jugements ou les informations extraites à propos d'un même problème. On peut procéder par fusion de données ou fusion de décisions (§4.6.2 p. 76) & (fig. 4.6, p. 76). La plupart des approches coopératives de segmentation fonctionnent selon ce mode (§2.5.1 p. 30).
2. **La coopération augmentative**
   La coopération augmentative correspond à une décomposition de la tâche à traiter en sous-tâches, plus ou moins disjointes et de même nature, qui s'exécutent de manière concurrente. Cette coopération est **émergente** : les agents remplissent des objectifs locaux sans intention explicite de coopération. Leurs activités modifient localement l'environnement influençant par effet de bord les agents voisins. La solution est l'union des sous-résultats locaux. La coopération n'existe que du point de vue de l'observateur, à l'image des sociétés de fourmis. Cette coopération correspond souvent en vision à une distribution

spatiale du traitement, comme l'illustrent les agents situés dans l'image (§5.5.3 p. 100).

3. **La coopération intégrative**
   La coopération intégrative traduit la pensée de Descartes, qui consiste à décomposer le problème ou la tâche à traiter en sous-tâches de natures différentes. Cette coopération est **intentionnelle** car elle nécessite la création d'un plan d'actions concernant la réalisation de la tâche. Les agents concernés vont alors travailler selon ce plan séquentiel, chaque agent intégrant les résultats des agents précédents. Certaines approches coopératives en segmentation d'image (§2.5.2 p. 30) illustrent bien cette coopération. Les architectures de vision centrées tâche (§5.5.1) obéissent à ce type de coopération.

Les systèmes réels résultent bien souvent d'un mélange de ces trois types de coopération.

5.4.2 La coordination

Lorsque des agents ont décidé de coopérer (coopération intentionnelle), la coordination vise à organiser l'activité des agents. Cette étape correspond au **contrôle social** ou collectif du système. On procède en deux étapes : le choix d'un plan d'exécution centralisé ou distribué, puis le suivi par synchronisation des agents de ce plan. Cette coordination peut prendre essentiellement deux formes :

1. Coordination centralisée : dans ce type de coordination, une seule entité, le coordinateur, va être responsable de la bonne exécution d'un plan. Il va donc, en suivant son plan, envoyer des ordres d'actions aux agents esclaves et attendre un accusé de réception ou le résultat de l'exécution de l'action pour ordonner l'action suivante. Cette coordination suppose donc une hiérarchisation (qui peut être temporaire) de l'organisation. Cette méthode facilite la conception du programme de coordination car elle permet d'éviter une dispersion des points de synchronisation.

Cependant, la centralisation des messages vers le coordinateur peut provoquer un goulot d'étranglement ralentissant les performances du système. De plus, cette approche plus naturelle aux informaticiens pousse à vider les agents esclaves de toute complexité qui se retrouve concentrée dans le coordinateur, diminuant ainsi l'intérêt d'une approche multi-agents.

On retrouve ce mécanisme dans les stratégies des schémas de Schema System [Dra89], où la stratégie pilote l'exécution des différentes sources de connaissances.

2. Coordination distribuée : plusieurs agents ont décidé de coopérer ; ils vont donc être responsables chacun d'une partie du plan et des synchronisations les concernant. Chaque agent va choisir un protocole d'interaction adapté à la coopération qu'il souhaite mettre en œuvre . Ces protocoles représentent des enchaînements

d'interactions possibles sous la forme d'un automate à états finis ou d'un réseau de Pétri.

Cette approche vise à augmenter l'autonomie décisionnelle des agents, ce qui, comme le souligne Boissier [Boi93], améliore la flexibilité et l'ouverture du système. Son architecture agent ASIC [Boi94a] illustre d'ailleurs parfaitement ce type d'interaction.

5.5 Typologie des systèmes de vision à base de SMA

Nous allons tenter de donner une typologie des systèmes de vision à base de SMA. L'utilisation des SMA en vision a donné lieu à trois familles d'approches se distinguant dans la décomposition qu'elles font du problème. Nous allons décrire l'organisation générale de chaque famille ainsi que la nature des agents qui la composent. Bien sûr, certaines architectures relèvent de plusieurs familles ; mais chaque famille traduit une orientation que les concepteurs ont voulu donner à un système et des points communs forts sur l'organisation et la nature des agents.

Nous allons présenter ces trois familles illustrées pour chacune d'elles par un ou plusieurs systèmes. Toutes relèvent des SMA, hormis SIGMA et MESSIE-II qui sont des architectures à base de tableaux noirs.

5.5.1 Les approches centrées tâche

Les différents niveaux de représentations évoqués section 4.2.1 p. 52, suggèrent une décomposition en tâches transformant les données d'un niveau vers le suivant.

Les organisations ainsi obtenues sont hiérarchiques (§5.4 p. 92), les agents des niveaux supérieurs adressant des requêtes aux agents des niveaux inférieurs. Le rôle respectif des agents est prédéfini et les relations sont fixées. L'organisation est donc de type *fixe - hiérarchique - prédéfinie*.

Cependant, les relations entre agents pourraient varier en fonction d'un besoin particulier, comme par exemple un appel d'offre par diffusion aux agents d'analyse d'image (inconnus à l'avance) pour réaliser une opération de segmentation. Ce n'est pas le cas des architectures présentées ci-dessous.

La coopération est généralement à caractère intégratif, les agents des niveaux supérieurs intégrant les résultats des agents des niveaux inférieurs.

Type et nombre d'agents : ces architectures sont constituées de peu d'agents cognitifs à forte granularité. L'objectif n'est pas d'obtenir des capacités d'auto-organisation mais plutôt des qualités en termes de génie logiciel, l'accent étant mis sur la modularité et la flexibilité.

5.5.1.1 Le système MAVI : une variante de VAP

Le projet européen VAP [Cro89] a pour objectif la conception d'un système intégré de vision active ; un des points saillants étant de maintenir une forte modularité entre six modules de traitement afin de permettre le développement séparé de chacun des modules, puis leur intégration.

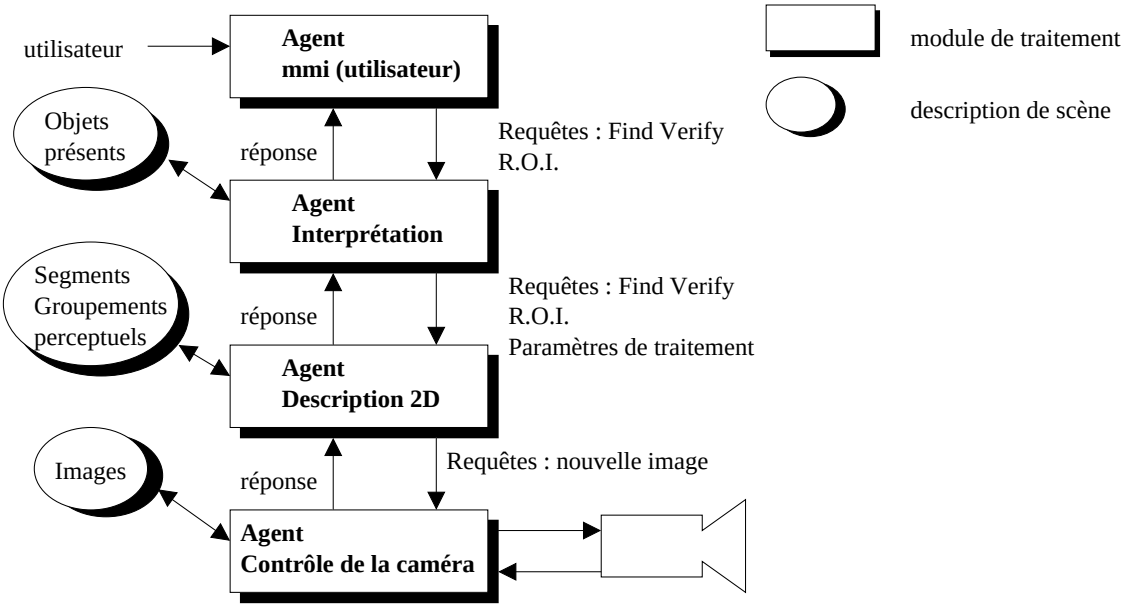


Fig. 5.5 : *Le système MAVI : une implémentation de VAP. D'après [Boi93, Boi97].*

ASIC (Architecture for Social and Individual Control) [Boi94a, Boi93] (§5.3.3.2 p. 91) propose un évolution de l'architecture de contrôle des agents de VAP. MAVI [Boi94b] est un sous-ensemble des agents VAP modifiés selon le modèle de contrôle d'ASIC. L'architecture de MAVI se présente sous la forme de quatre agents (fig. 5.5, p. 96) traitant chacun un niveau de représentation. Chaque agent se présente comme le contrôleur de l'agent de la couche inférieure (qui est vu comme un processus), il peut lui adresser des requêtes de recherche ou de vérification sur la présence d'objets ou de primitives. Ces requêtes sont accompagnées de paramètres de focalisation spatiale (ROI, *Region Of Interest*) ou de paramètres particuliers destinés aux procédures de segmentation ou groupements perceptuels.

Ce cycle de perception n'est pas figé (strictement ascendant ou descendant, ou ascendant puis descendant), des boucles de contrôle peuvent s'installer dynamiquement au sein d'un couple d'agent contrôleur/processus. La coordination locale nécessaire à la mise en œuvre de ces boucles de contrôle est effectuée à l'aide de protocoles, assurant une grande flexibilité de l'architecture.

5.5.1.2 SIGMA (*blackboard*)

SIGMA [Mat90], développé par Matsuyama pour l'interprétation d'imagerie aérienne, propose trois spécialistes représentant la connaissance opératoire nécessaire

à la reconnaissance des objets :

1. L'expert de vision bas-niveau LLVE (Low Level Vision Expert) construit des plans définissant un séquencement d'opérateurs et leur paramétrage en réponse à une requête d'extraction de primitives particulières.
2. L'expert de sélection de modèle MSE (Model Selection Expert) filtre les primitives pour trouver les objets du modèle leur correspondant (génération d'hypothèses ascendante par identification) ; inversement il transforme la recherche d'un objet en requêtes d'extraction de primitives particulières adressées au LLVE (détection descendante).
3. L'expert sur le raisonnement géométrique GRE (Geometric Reasoning Expert) raisonne, en utilisant les informations des classes, sur les relations spatiales entre objets. En mode ascendant, il vérifie la cohérence spatiale des objets détectés (hypothèses) ou prédits par le MSE. Il utilise les liens entre les hypothèses (instances de classes) pour accumuler de l'évidence sur l'existence mutuelle d'hypothèses. En mode descendant, il prédit l'existence d'un objet (génération d'hypothèses par prédiction) sur la base d'une hypothèse existante et lance des requêtes de détection vers le MSE.

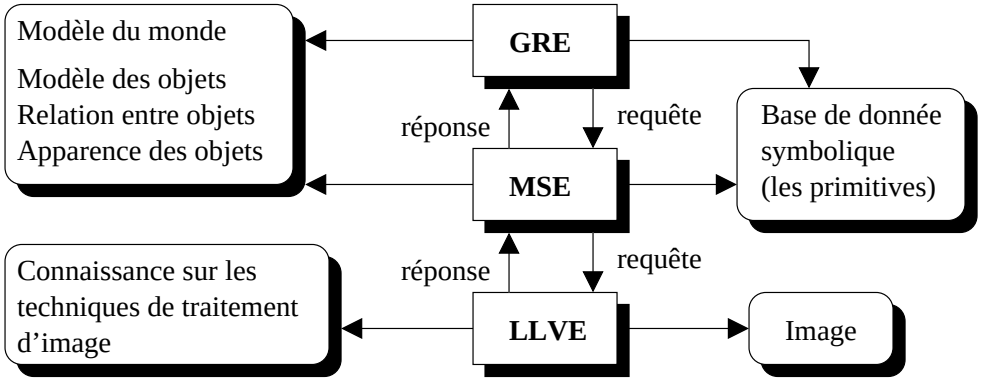


Fig. 5.6 : *Le système SIGMA. D'après [Mat90].*

Cette succession opportuniste de cycles d'interprétations illustre l'aspect hétérarchique de la stratégie d'interprétation de SIGMA.

Le modèle des objets (LTM) est constitué d'un réseau de "classes" représentant chacune une classe d'objets. Une classe du modèle SIGMA est un mélange de la classe, au sens de la programmation objet, et d'un frame. Elle contient donc une connaissance décrivant la forme de l'objet ainsi que des liens de composition, de spécialisation et de vue avec d'autres classes, spécifiant ainsi les relations attendues. Des règles attachées aux classes permettent de prédire l'existence d'un objet voisin si la classe courante est instanciée sous certaines conditions.

Ces règles leur donnent l'apparence d'objets actifs, "sorte d'agents", mais contrairement aux agents, les classes ne sont pas autonomes. En ce sens SIGMA est aussi orienté modèle ou concept.

5.5.1.3 MESSIE-II (*blackboard*)

MESSIE-II [San95] a une décomposition similaire à SIGMA, mais, il est construit sur un blackboard hiérarchique où coopèrent des spécialistes regroupés en trois familles. Chacun des spécialistes dispose des connaissances opératoires de traitement des concepts de son niveau de description. Il y a un spécialiste de la scène gérant les relations spatiales, des spécialistes des objets (détection d'objet et groupement perceptuel) et des spécialistes bas-niveau (paramétrage des opérateurs d'analyse d'image en vue d'extraire des primitives).

Le modèle se présente comme une hiérarchie de composition de classes d'objets et de scènes mises en œuvre à l'aide de frames. Le modèle (LTM) et les hypothèses (STM) sont stockés dans le tableau noir. La stratégie d'interprétation peut être descendante en utilisant le mécanisme de contrôle hiérarchique. La stratégie peut aussi être ascendante grâce à la notion d'événement permettant de déclencher des spécialistes de manière opportuniste.

5.5.2 Les approches centrées modèle

Ces approches soulignent l'importance du modèle hiérarchique des concepts (ou hypothèses), ces derniers étant les unités actives du système. Les concepteurs y ont privilégié une représentation éclatée, modulaire et incrémentale du modèle. Les agents du modèle disposent de la connaissance opératoire pour piloter les tâches (ou procédures visuelles) et pour coopérer suivant les liens existant entre eux.

Organisation : le rôle respectif des agents est prédéfini mais les relations entre agents peuvent varier en fonction des reconnaissances effectuées. L'organisation est donc de type *variable - hiérarchique - prédéfinie.*

La coopération est intégrative : une hypothèse d'un niveau supérieur intègre les résultats de ces sous-parties. La coopération est aussi confrontative : les procédures utilisant les cohérences spatiales mutuelles tendent à confirmer ou à infirmer la certitude associée aux hypothèses.

Type et nombre d'agents : ces architectures sont constituées de quelques dizaines d'agents cognitifs de granularité moyenne à forte[1] (chaque agent étant lui même un blackboard).

Nous allons présenter les systèmes VISION/Schema System, néanmoins la conception de SIGMA et ses frames donnant l'apparence d'objets actifs, SIGMA pourrait presque être classé dans cette catégorie.

5.5.2.1 VISIONS/Schema System

VISIONS [Han78] développé par Hanson et Riseman s'inscrit dans le cadre d'un projet à long terme prolongé par Schema System [Dra89]. Schema System est com-

[1] Draper n'est pas précis à ce propos, mais son modèle contenait en 1989 20 objets

posé d'un modèle, d'un ensemble de spécialistes des tâches visuelles, d'une base de données stockant les primitives et d'un tableau noir général.

Le modèle se présente comme une structure hiérarchique de compositions et de vues dont les éléments (les schémas) sont les concepts à identifier dans l'image. Chaque schéma comporte des connaissances descriptives : attributs géométriques, photométriques, liens éventuels vers ses sous-parties. Il comporte aussi des connaissances opératoires (stratégies) nécessaires à la reconnaissance dans l'image des instances de sa classe. Les stratégies se présentent sous la forme de procédures pilotant l'activation des sous-schémas (suivant les liens *part-of*), ou des spécialistes d'analyse d'image. Lorsqu'un concept est détecté (bottom-up) ou prédit (top-down suivant les *liens part-of*) alors, une hypothèse est générée sous la forme d'une instance de la classe du schéma correspondant au concept. Nous voyons donc que la génération des hypothèses (§4.5.3 p. 70) peut fonctionner en mode ascendant (identification) ou descendant (prédiction) selon une stratégie hétérarchique.

Le système regroupe une vingtaine de spécialistes d'analyse d'image de bas-niveau et de niveau intermédiaire. Les spécialistes de bas-niveau produisent des résultats dans la base de données intermédiaire ISR (§4.3.1 p. 59). On trouve des spécialistes de segmentation région, d'extraction de caractéristiques des régions, d'extraction et de groupement de segments de contours. Les spécialistes du niveau intermédiaire traitent et produisent des données dans l'ISR. On trouve des spécialistes d'identification d'objets selon des critères photométriques, d'appariement de régions similaires, de groupement perceptuel, d'appariement de graphes et d'analyse des relations spatiales entre primitives (région-région ou région-contour). La plupart de ces spécialistes peuvent être utilisés en mode guidé par les données ou en mode requête.

Ces spécialistes sont activés à l'initialisation du système avec des paramètres par défaut, en mode guidé par les données pour produire un premier jeu de primitives dans l'ISR. Cela va permettre de générer quelques hypothèses (les plus simples) c'est-à-dire détecter les classes d'objets que l'on pense être présents dans l'image. Les schémas correspondants sont instanciés et vont désormais orchestrer l'activité du système.

Le contrôle du système est totalement distribué au sein des schémas, le tableau central n'intervient que comme un environnement où les schémas déposent les hypothèses dont ils sont sûrs. Ainsi, le tableau noir permet la détection d'hypothèses conflictuelles (§4.5.4 p. 71).

5.5.2.2 SMA pour la reconnaissance de forme : approche de Yanai & Deguchi

Yanai et Deguchi [Yan98] proposent une modélisation d'un système de reconnaissance sous la forme d'un SMA. Chaque agent dispose d'un module permettant de reconnaître un concept particulier dans l'image. Il dispose aussi d'un module de communication sur lequel va se bâtir une coopération entre agents. Ce module vérifie la consistance des interprétations faites par le module de reconnaissance de l'agent

courant et les interprétations faites par les autres agents. Les modules de communication se tiennent au courant des interprétations par un mécanisme de diffusion de messages (*broadcast*). La coopération prend alors deux formes :

1. En cas de conflit, un mécanisme permet de choisir la meilleure des deux interprétations sur des critères de forme et de respect par l'hypothèse des structures relationnelles de l'objet.
2. En utilisant les hypothèses déjà détectées et les liens relationnels entre agents, alors de nouvelles hypothèses peuvent être générées.

5.5.3 Les agents situés dans l'image

Cette approche, plus récente que les autres, s'applique pour l'instant essentiellement à la vision bas-niveau. L'image y est vue comme un environnement ou territoire analysé par une société d'agents distribués spatialement. Cette approche est novatrice car elle va à l'encontre de la conception classique centralisée et planifiée des tâches d'analyse d'image. Ces architectures mettent en relief la distribution du calcul, l'adaptation locale des traitements et la richesse des coopérations possibles entre les aspects contour et région.

Organisation : le rôle respectif des agents est prédéfini mais les relations entre agents peuvent varier en fonction des rencontres faites dans l'environnement qu'est l'image. L'organisation est donc de type *variable - égalitaire - prédéfinie*.

La coopération est globalement augmentative, les agents travaillant plutôt de façon disjointe. Mais elle peut prendre un caractère confrontatif lorsque les agents accumulent de l'information dans l'environnement et peuvent travailler sur des zones communes. Finalement, elle peut aussi être intégrative lorsque des agents initialisent d'autre agents dans l'image pour aller récolter pour eux de l'information. Dans ce cas, des relations hiérarchiques sont créées.

Type et nombre d'agents. Ces architectures sont constituées d'un grand nombre d'agents faiblement cognitifs pouvant être réactifs et dont la granularité est généralement moyenne ou faible.

5.5.3.1 Agents coopératifs à approche incrémentale

Salotti [Sal94] puis Bellet [Bel98, Bel94] proposent une approche distribuée à base de processus situés dans l'image. Ces processus travaillent de façon incrémentale à partir d'un germe en faisant croître une primitive. Cette primitive peut être une région auquel cas le processus est de type croissance de région, mais elle peut aussi être un contour traité par un processus de type suivi de contour.

À chaque instant, les processus tiennent à jour une liste des pixels candidats à l'agrégation. L'évaluation des pixels candidats se fait selon plusieurs critères. Lorsque l'évaluation des pixels candidats n'est plus satisfaisante, le processus va tâcher de

faire émerger de l'information. Pour cela, il peut coopérer avec des processus topologiquement voisins, ou bien créer de nouveaux processus fils qui devront faire remonter l'information vers le père. Par exemple, un processus de type suivi de contour peut souhaiter faire émerger deux régions de part et d'autre du contour afin de conforter son choix du prochain pixel à ajouter à l'extrémité du contour. Un processus de type croissance de région peut souhaiter faire émerger de l'information contour "devant lui" afin d'obtenir une bonne localisation de l'endroit où il doit s'arrêter. Les processus de même nature seront également amenés à négocier entre eux des fusions, afin de regrouper au sein d'une même entité des morceaux de primitives faisant partie de la même région ou du même contour. En outre, ceci permet de diminuer la charge du système, qui est gérée par un mécanisme de contrôle global (similaire à un système d'exploitation multitâche) d'allocation de ressources.

Cette approche permet d'introduire une coopération opportuniste dans laquelle les différents types de méthodes (contour/région) s'influencent mutuellement. Elle fournit de nombreux points de contrôle, tant dans l'espace de l'image que dans le temps de la croissance de primitive.

Boucher [Bou99] étend les processus à des agents permettant de mieux identifier les structures internes aux agents et leurs interactions. Les agents disposent de quatre comportements :

1. Le comportement de perception explore son environnement et sélectionne des pixels candidats à l'agrégation.
2. Le comportement d'interaction gère les fusions et les échanges de pixels[2] entre agents.
3. Le comportement de différenciation correspond à une interprétation à valeur sémantique. Les agents représentent au départ des primitives région ou contour ; les informations accumulées sur eux-mêmes et dans un environnement commun vont leur permettre de déterminer la nature sémantique de la primitive traitée.
4. Le comportement de reproduction, similaire à celui de Bellet, permet de créer au besoin des agents à certains endroits de l'image. Il conditionne la stratégie d'exploration de cette image et donc le contrôle du système.

Dans les travaux de Germond [Ger99] sur des images IRM du cerveau, différentes coopérations sont définies entre un modèle déformable, un système multi-agents et un détecteur de contour. Le modèle déformable est utilisé comme contour grossier du cerveau. Il permet de positionner des germes agents (croissance de région) qui vont segmenter, selon le processus décrit ci-dessus, la matière blanche et la matière grise du cerveau.

Les contours des régions obtenues vont être affinés par d'autres agents "contour" cherchant un chemin de contour optimum (A*). La fonction de coût prend en compte à la fois le détecteur de gradient et les régions obtenues précédemment. Le nouveau

[2]fonctionnalité encore non implantée dans [Bou99]

contour sert soit à améliorer la segmentation sur la même image, soit à segmenter d'autres images du volume IRM initial.

5.5.3.2 Segmentation par "agents migrateurs" : approche de Liu & Tang

Dans cette approche [Liu99], des agents répartis aléatoirement dans l'image vont migrer (se diffuser) de pixel en pixel jusqu'à ce qu'ils trouvent dans leur environnement local un motif recherché. Une fois fixé sur un pixel, l'agent l'étiquette puis se reproduit en initialisant de nouveaux agents dans son voisinage et devient inactif.

Au cours de sa migration, l'agent va vieillir et peut mourir si le nombre de cycles de recherche excède son temps de vie. Les auteurs ont aussi introduit la notion d'évolution associée aux directions de diffusion et de reproduction. En effet, en cas de succès d'un des comportements dans une direction, la pondération associée à cette direction sera renforcée. Ainsi, les comportements de migration et reproduction de ses descendants favoriseront cette direction.

Les auteurs appliquent leur approche à la détection de segment de contour ; les agents migrent jusqu'à trouver un contour, ils s'y fixent, l'étiquettent et initialisent de nouveaux agents qui auront de fortes chances de trouver un contour recherché dans un voisinage proche.

Deuxième partie

Propositions et expérimentations

Chapitre

6

Problématique, démarche et cadre conceptuel

Lors de ce chapitre transitoire entre l'état de l'art et nos propositions, nous discuterons des trois aspects (adaptation locale, intégration de l'*a priori* , coopération) de la segmentation identifiés lors des chapitres précédents.

Nous pensons qu'il est important qu'une architecture logicielle propose des concepts adaptés à ces trois aspects.

Nous illustrerons notre propos à l'aide d'un problème de segmentation d'une image de scanner du sein. Puis nous étendrons brièvement l'analyse au problème de la vision en général.

Finalement, nous présenterons les principes généraux de nos propositions.

6.1 Problématique de segmentation et ébauches de solutions

L'IAD (chap. 5) est couramment utilisée dans la conception des systèmes de vision pour les raisons évoquées à la section 5.1 p. 84. Cependant, son utilisation en vision bas-niveau sous la forme d'agents situés dans l'image (§5.5.3 p. 100) est plus rare. Nous allons dans cette section (fig. 6.1, p. 107) dégager trois problèmes clés de la segmentation et essayer de montrer en quoi les caractéristiques des SMA énoncées (§5.1 p. 84) peuvent fournir des solutions.

La figure 6.1 précise la démarche évoquée en introduction de la thèse (§1.1 p. 11). Cette démarche consiste à décomposer notre analyse du problème (de la segmentation et plus généralement de la vision) suivant un axe proposant trois niveaux d'abstraction. Pour chaque niveau (excepté celui de l' implémentation matérielle) on dégage une méthodologie permettant de contraindre le niveau inférieur.

Pour l'analyse du niveau conceptuel, nous avons adopté une méthodologie articulée autour de trois points : coopération, adaptation et intégration de l'*a priori* .

Nous allons montrer que ces trois aspects se traduisent au niveau de l'implémentation logicielle par un ensemble de contraintes :

1. modularité et ouverture ;
2. diversité des formalismes ;
3. flexibilité et création dynamique de boucles de contrôle ;
4. focalisation et adaptation ;
5. robustesse des traitements ;
6. calcul distribué.

6.1.1 Un problème particulier : image de scanner du sein

Nous utiliserons des images de scanner de sein (fig. 6.2, p. 108) afin d'illustrer notre propos sur un problème médical. Toutefois, il faut garder à l'esprit que nos travaux se situent sur un **niveau méthodologique** et ne sont donc pas dédiés à une image particulière. L'approche propose un cadre conceptuel enrichi permettant de respecter les six contraintes évoquées ci-dessus.

Comme tout examen radiographique, la mammographie présentée est le reflet de l'absorption des rayons X par les différents tissus.

Les tissus absorbant fortement le rayonnement X apparaîtront en blanc sur l'image et sont dit "opaques" (exemples : le tissu fibreux, glandulaire, tumoral ou calcifié). Par contre, les tissus peu denses comme la graisse apparaîtront en noir et sont dits "radio-transparents".

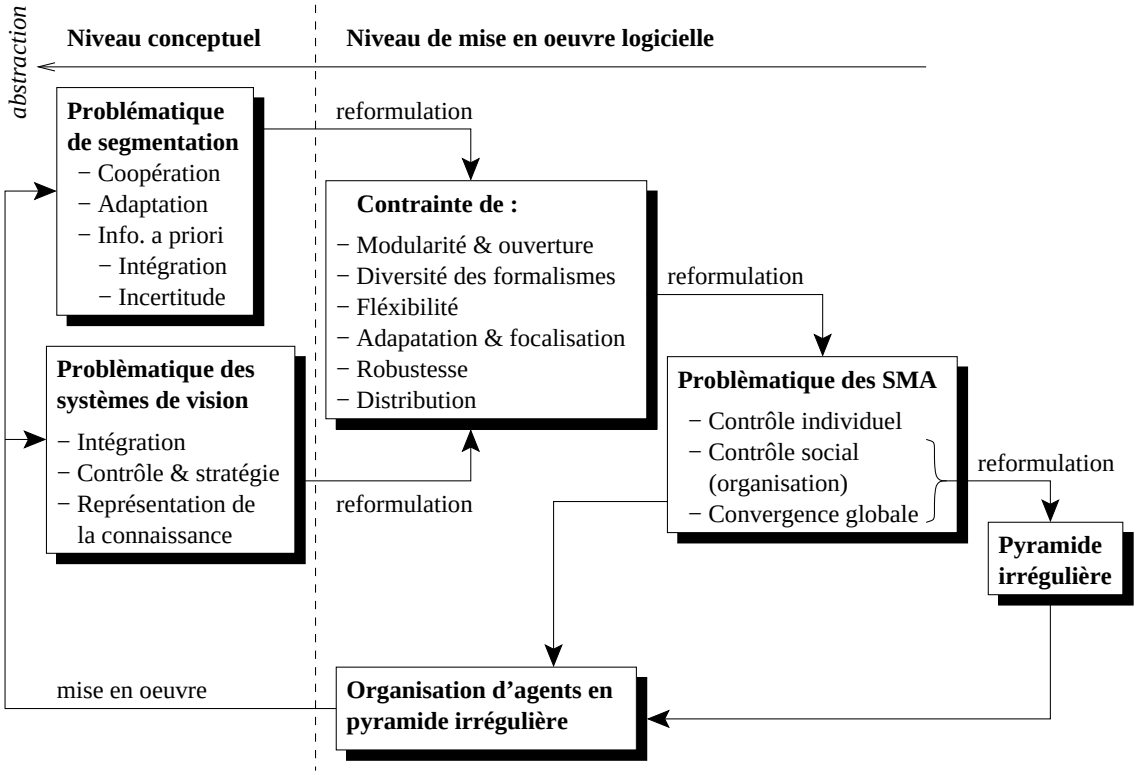


Fig. 6.1 : *Démarche conceptuelle : d'une méthodologie portant sur le niveau conceptuel à une organisation concrète d'agents.*

Les tumeurs sont des tissus hyper-vascularisés se logeant dans la zone de tissu glandulaire. La zone de **Tissu Glandulaire (TG) est donc la région d'intérêt** que nous souhaitons isoler afin de pouvoir évaluer la surface de manière automatique.

Comme le montrent les expérimentations menées au chapitre 10, les approches région ou contour échouent dans une segmentation robuste de cette image.

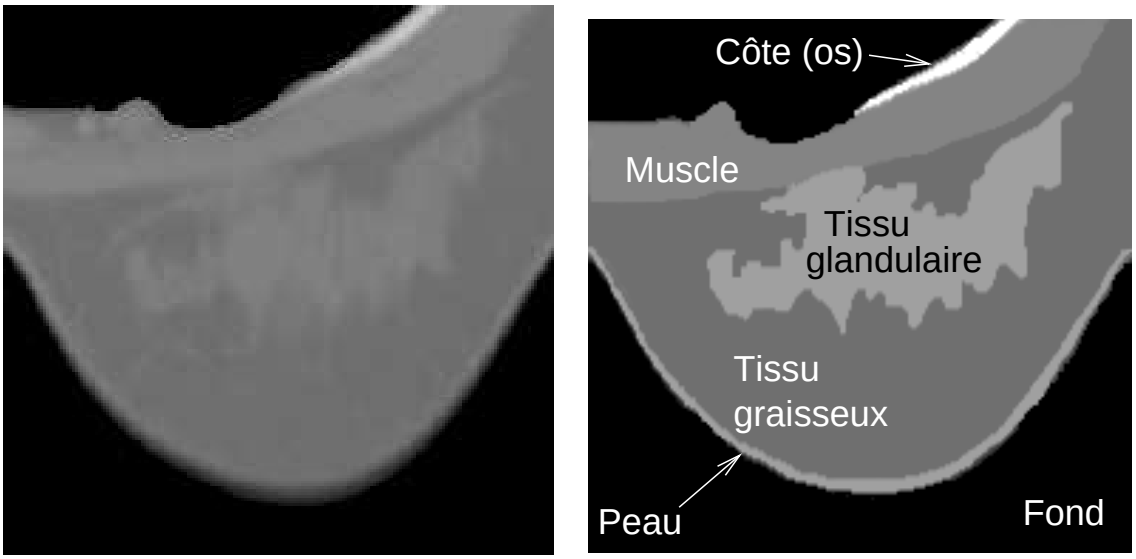
Ceci est dû à la grande similarité de la photométrie entre TG et le muscle :

Niveau de gris / Zone	min	max	moyenne	écart-type
Muscle	128	133	130	1.7
Tissu glandulaire (TG)	124	131	127	2.1

Tab. 6.1 : *Attributs photométriques du muscle et de la zone de tissu glandulaire.*

6.1.2 La nécessité de coopération

Les niveaux de gris de TG varient dans l'intervalle $[124, 131]$ et ceux du muscle dans $[128, 133]$; cela rend inefficace une classification monodimensionnelle ((f), fig. 10.7, p. 219). Du fait de textures similaires, les approches multidimensionnelles ne réussissent guère mieux ((e), fig. 10.7).



(a) (b)

Fig. 6.2 : *(a) Une image de scanner de sein ; (b) une partition idéale faite à la main et la légende des différentes régions dont la zone de Tissu Glandulaire (TG), qui est la région d'intérêt recherchée.*

Un point de contact entre le muscle et TG (fig. 6.3, p. 109) risque d'entraîner une fusion des deux régions par des approches structurales de type croissance ou fusion de régions (fig. 10.17, p. 231). En effet, le profil des niveaux de gris le long d'une ligne traversant un point de contact (fig. 6.3) montre une rupture de faible intensité.

La faiblesse de la transition inférieure du muscle, la présence de points de contacts avec TG et surtout l'irrégularité et la faiblesse des frontières de TG ne permettent pas d'envisager une approche contour ((d), fig. 10.7, p. 219).

Intuitivement, le système de vision humain sépare les deux régions en exploitant la continuité de la frontière basse du muscle. Cette frontière est faible, interrompue aux points de contacts muscle/TG mais elle est étendue. Ainsi, il semble qu'une approche coopérative de type croissance de région contrainte par les contours (§2.5 p. 29) puisse résoudre une partie du problème.

De manière duale, l'utilisation de certaines approches coopératives (§2.5 p. 29) et (§5.4.1 p. 93) permettrait d'améliorer le suivi du contour formant la frontière basse du muscle en s'appuyant sur les régions à sa droite et à sa gauche.

Les entités impliquées dans les coopérations sont donc hétérogènes (point 1 & 2 des caractéristiques évoquées en début de chapitre) posant des problèmes d'intégration ; de plus, les coopérations doivent pouvoir s'installer de manière dynamique (point 3).

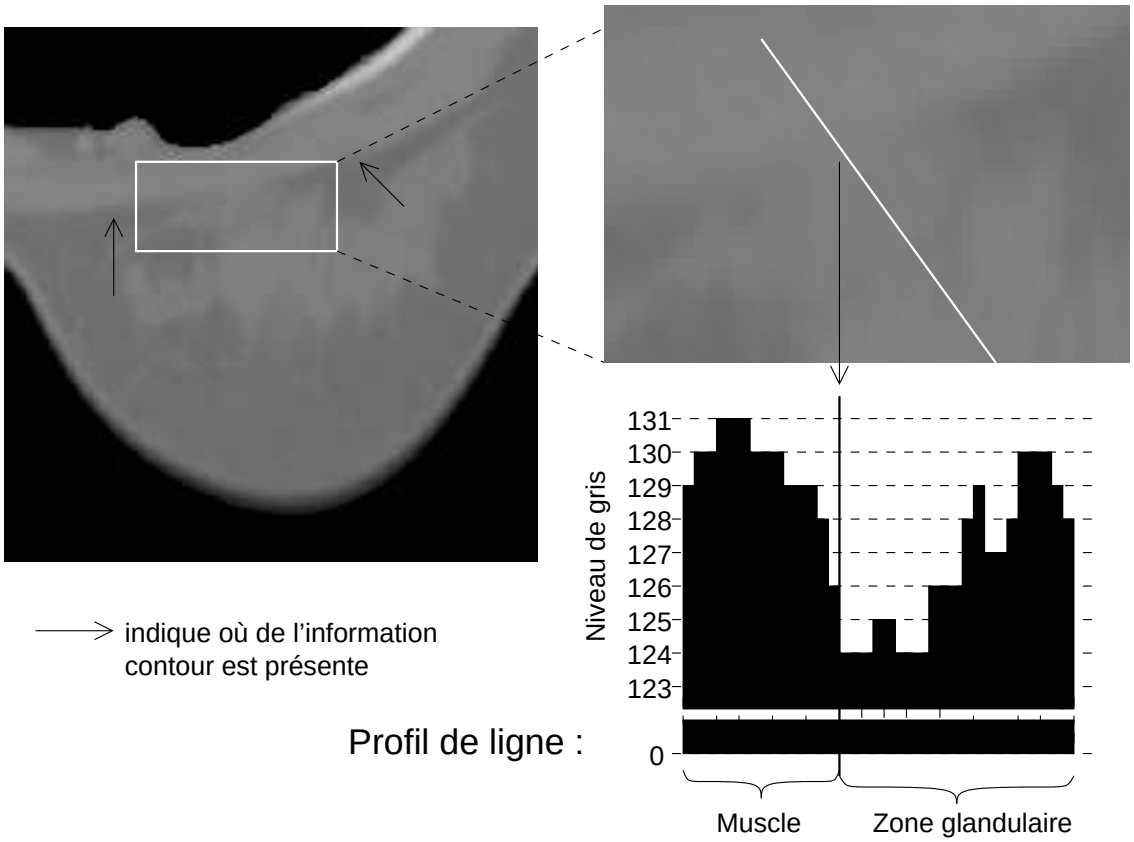


Fig. 6.3 : *Zoom sur la zone de contact entre TG (Tissu Glandulaire) et le muscle.*

6.1.3 La nécessité d'adaptation locale

Cependant, cette coopération n'apparaît possible que par une vision plus globale des formes au sens de l'organisation perceptuelle (§4.4.3 p. 65). En effet, avant de disposer de régions suffisamment grandes pour capter l'information contour présente à gauche et à droite de la zone de contact, la croissance se doit d'être beaucoup plus prudente, et donc d'adapter (point 4) sa tactique d'agrégation ou fusion.

L'adaptation spatiale évoquée ci-dessus fait référence à une modification de seuils d'agrégation ou de fusion. Mais l'adaptation doit aussi permettre une modification en dynamique par rapport au contexte (texture, bruit) des critères de similarité, tant pour les approches de croissance de régions que pour les approches de type suivi de contour.

L'adaptation peut aussi être envisagée comme le choix d'opérateurs adaptés au contexte local de l'image ; dans ce cas, le problème d'adaptation se réduit à un problème de fusion d'informations et, donc, de coopération confrontative entre plusieurs formalismes (point 2) issus de l'application des différents opérateurs.

Ainsi, il semble pertinent de distribuer des entités "intelligentes" capables d'adaptation et de focalisation locale dans l'image. Elles doivent pouvoir coopérer afin de capter des aspects moins locaux et fusionner leurs résultats.

6.1.4 La nécessité de représentation de l'*a priori* et de l'incertitude

Intégration de l'*a priori* . Bien que la prise en compte d'informations liées au domaine lors de la segmentation divise encore les chercheurs, notre opinion est que toute information disponible doit être utilisée au plus tôt afin d'éviter des erreurs difficilement réparables. E. Saund dans [Boy00] utilise les termes : "modèle fort" et "modèle faible" (§4.4.3 p. 65) à propos de l'opposition des deux approches.

L'IA a connu un débat similaire opposant les méthodes faibles et "tout terrain" de type exploration combinatoire[1] aux approches intégrant un maximum de connaissances comme les systèmes experts.

Les solutions finalement retenues (par ex. : les jeux d'échecs [J.M94]) sont comme souvent un compromis entre les deux approches où l'utilisation d'heuristiques très perfectionnées, traduisant la connaissance du domaine, guident la recherche combinatoire.

Il faut donc chercher des formalismes capables de mettre en œuvre des comportements par défaut, robustes et "tout terrain", en l'absence d'information *a priori* , et qui si cette dernière est disponible, peuvent l'intégrer harmonieusement en guidant efficacement le processus de segmentation à la manière des heuristiques en IA.

L'influence concrète du modèle en segmentation se traduit généralement par une focalisation spatiale (§4.5.2 p. 69) ainsi que par une paramétrisation et un séquencement particulier d'opérateurs (§4.5.1 p. 68).

La prise en compte de l'*a priori* se résume donc à établir un lien privilégié entre les éléments du modèle et les modules de segmentation, les premiers adaptant (point 4) le comportement des seconds. Cette adaptation peut être obtenue grâce à une boucle de contrôle établie dynamiquement (point 3) d'une entité du modèle sur une entité de segmentation.

En résumé, l'utilisation des informations du modèle peut se réduire à la mise en place d'interactions (coopération) intelligentes avec le modèle, puis à une adaptation.

Intégration de l'incertitude. L'analyse des niveaux de gris de TG et du muscle ne permet pas de dégager un seuil de similarité global entre régions dans l'image. En effet, si le traiteur d'image spécifie un seuil de ressemblance trop grand (par ex. 7), cela risque de provoquer une fusion des régions muscle et TG, un seuil trop petit (par ex. 2) provoquera une sur-segmentation des deux régions empêchant de calculer la surface de la TG.

[1] dont A* (algo. A.2 p. 257) est un représentant

Il est donc nécessaire de fournir un cadre d'expression explicite de l'incertitude, favorisant sa détection et sa prise en compte par des comportements prudents et/ou coopératifs.

6.2 Problématique étendue à la vision

Notre objectif dans le futur est de fournir un système de vision intégrant l'interprétation de scènes fixes 2D sur la base d'une organisation pyramidale d'agents. Lors de nos travaux sur la segmentation, nous avons structuré notre démarche dans cet objectif, ce que reflète bien l'état de l'art et surtout le chapitre 4.

L'utilisation de l'IAD en système de vision est couramment admise (voir chapitres 4 & 5). Elle fournit en effet des outils de représentation éclatée de la connaissance facilitant ainsi son ingénierie et son intégration. Elle apporte des solutions de contrôle permettant de supporter des stratégies ascendantes, descendantes, mixtes et même hétérarchiques.

Le cloisonnement des disciplines, traitement du signal d'un côté, sciences cognitives de l'autre, a fait apparaître une frontière artificielle entre analyse d'image et interprétation. Certes, il y a plusieurs étapes utilisant des outils différents du traitement de l'information, mais cette dichotomie nuit à une interaction harmonieuse entre le bas et le haut-niveau de la vision. En effet, si l'on représente le module de segmentation comme un bloc "monolithique" qui, après traitement, soumet ses résultats à un autre bloc "monolithique" qu'est le module d'interprétation, l'interaction entre ces deux blocs sera tardive, grossière et brutale. En conséquence, l'interprétation ne peut corriger la segmentation qu'après exécution de celle-ci ; cette correction sera basée sur toute l'image et elle signifiera quelque chose comme "recommence à zéro avec ces nouveaux paramètres".

Nous souhaitons tendre vers une architecture homogène dans laquelle la segmentation n'est qu'un cas particulier de reconnaissance de forme. Ainsi, la confrontation avec le modèle peut se faire dès les premières étapes de la vision, favorisant une utilisation précoce des informations *a priori* se traduisant par une adaptation locale des comportements de fusions de régions.

La similarité entre les schémas de Draper [Dra89] et les agents situés dans l'image (§5.5.3 p. 100) tout comme les travaux de Boucher [Bou99], suggèrent la manière de relier harmonieusement le haut et le bas-niveau de la vision.

En conclusion, nous proposons un formalisme unifié et homogène : des agents situés dans l'espace et dans le modèle, qui interagissent avec leurs voisins afin de faire émerger des formes de plus en plus complexes. Les stratégies ne doivent pas être uniquement ascendantes mais hétérarchiques.

6.3 Cadre conceptuel

Les problèmes de coopération, d'adaptation et prise en compte des connaissances du domaine peuvent être reformulés comme un problème d'interaction entre agents hétérogènes situés dans l'espace et le modèle, et qui respectent un ensemble de contraintes.

Il faut néanmoins appliquer à cette société un modèle d'organisation structurant les relations entre agents et régulant leurs activités, afin de garantir la convergence vers les objectifs globaux du système.

Nous proposons, comme le montre schématiquement la figure 6.4, d'utiliser la structure de la pyramide irrégulière (chapitres 3 & 8) car elle fournit un cadre formel d'organisation assurant une décroissance du nombre d'agents niveau après niveau et donc un contrôle de la combinatoire des stratégies de recherche dans l'espace des segmentations et interprétations possibles.

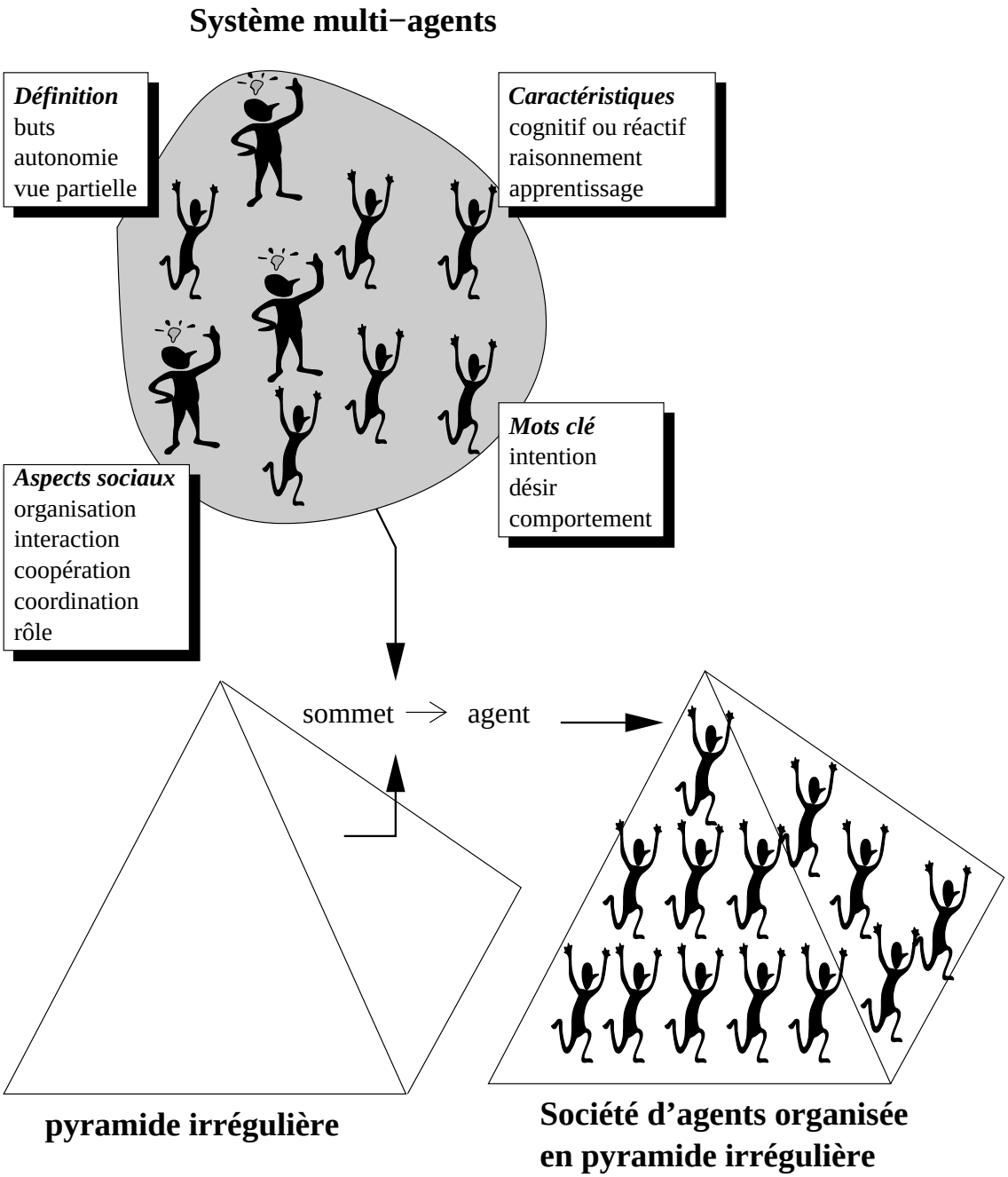


FIG. 6.4 : *Cadre conceptuel de notre approche.*

6.4 Principes généraux

Des sommets aux agents : nous allons au chapitre 8 effectuer une transposition d'une pyramide irrégulière, dont chaque niveau est un graphe d'adjacence, vers une "pyramide irrégulière d'agents" dont chaque niveau est une organisation d'agents.

Les agents fusionnent à travers sept comportements : dans le chapitre 9, nous décrirons en détails les divers comportements des agents et les interactions entre

agents voisins. Ces comportements, présentés schématiquement figure 6.5, sont au nombre de sept :

1. **Un comportement de marquage de territoire** qui consiste à ancrer l'agent dans l'image et à définir la partie (primitive région ou contour) de celle-ci dont il est le représentant.
2. **Un comportement d'exploration** permet aux agents de découvrir leurs voisins dans l'image. C'est avec ses voisins qu'un agent va interagir dans les comportements suivants. Ce comportement est l'équivalent agent de la construction du **graphe d'adjacence** des pyramides irrégulières.
3. **Un comportement de planification de fusion :** l'objectif d'un agent est d'interagir avec ses voisins adjacents afin de déterminer avec lesquels il souhaite fusionner. Il va donc construire un plan de fusion qui est l'équivalent agent du **graphe de similarité** des pyramides irrégulières.
4. **Un comportement coopératif :** le plan de fusion précédemment élaboré est entaché d'ambiguïtés : l'agent ne sait pas pour certains de ses voisins s'il doit fusionner ou non avec eux. Afin de lever ces ambiguïtés, il va mettre en œuvre un comportement coopératif consistant à demander l'avis d'agents présents dans son voisinage. Ces derniers pouvant représenter des primitives de région ou de contour, ce comportement permet une coopération région/région et région/contour. Les modifications entraînées sur le plan de fusion sont l'équivalent agent d'un traitement visant à améliorer le graphe de similarité.
5. **Un comportement de décimation** vise à sélectionner des survivants parmi les agents du niveau courant de la pyramide ; cette sélection est basée sur l'utilité associée au plan de fusion précédemment calculé. Ceci revient, dans le cadre des pyramides irrégulières, à favoriser la survivance des sommets ayant de nombreux arcs sortants de "bonne qualité" dans le graphe de similarité.
6. **Un comportement de rattachement** permet aux agents non-survivants de se rattacher à un survivant afin d'être représentés dans le niveau suivant de la pyramide.
7. **Un comportement de reproduction** permet aux agents survivants de créer un nouvel agent dans le niveau suivant de la pyramide. Ce nouvel agent représentera tous les agents s'étant rattachés au survivant.

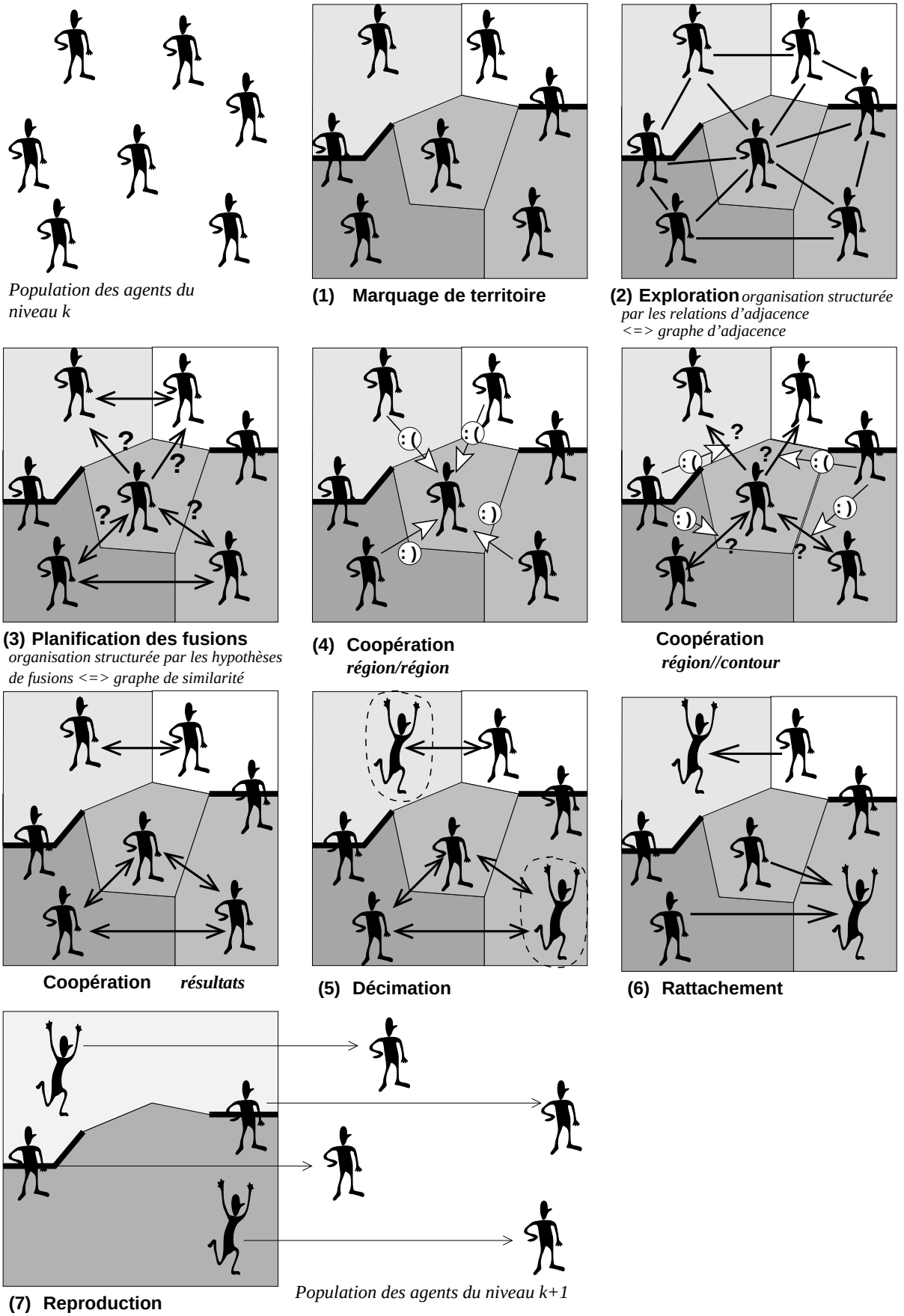


FIG. 6.5 : *Les sept comportements des agents permettant fusion et décimation des agents d'un niveau de la pyramide.*

Chapitre

7

Plate-forme logicielle

Pour des raisons expliquées en introduction de ce chapitre, les plate-formes disponibles au commencement des travaux de thèse n'étaient pas adaptées à nos problèmes. Un important travail de développement logiciel a permis la mise au point d'une plate-forme agent générique, indépendante du domaine applicatif.

Ce chapitre présente les différents concepts de la plate-forme. Nous évoquerons le micro-noyau du système qui fait la jonction avec les couches logicielles basses (Système d'exploitation...) et le matériel. Nous préciserons les principaux composants de l'agent et la manière de modéliser son contrôle.

Ce chapitre n'est pas consacré à l'image mais nous y expliquons le sens de certains termes employés lors des chapitres 8 et 9.

7.1 Introduction

Au début des travaux de thèse en Janvier 1999, nous avons mené une étude sur l'adéquation des différentes plate-formes agents disponibles pour notre problématique. Aucune, à l'époque, ne satisfaisait nos contraintes pour une ou plusieurs des raisons suivantes :

1. Non disponibilité des sources.
2. Chaque agent est associé à un processus entier posant des problèmes de mémoire dans le cas d'une utilisation de plusieurs milliers d'agents.
3. La plate-forme considérée sous-tend un modèle particulier d'agents avec, parfois même, un langage dédié. Or, ce type d'approche pose des contraintes fortes sur la nature des agents applicatifs. Par exemple, les agents devront être forcément de type BDI (§5.3.3.3 p. 92) ne permettant pas une faible granularité pour des agents simples. De plus, cette approche contraint le style de programmation à adopter, ce qui est une qualité pour un langage mûr (ADA, Lisp) mais un inconvénient majeur pour les langages encore immatures issus d'une discipline récente que sont les SMA.
4. La plate-forme fournit des outils inadéquats (par exemple : un système à base de règles d'ordre 0) et permet difficilement l'utilisation d'autres outils, qui nous semblent plus adaptés, sans amputer l'essentiel de l'intérêt de la plate-forme.
5. Le mode d'envoi de messages est inadapté : certaines plate-formes utilisent systématiquement TCP/IP, même si les agents s'exécutent sur la même machine, ce qui ralentit inutilement les performances.

Cependant, depuis quelques temps, certaines plate-formes, plus en adéquation avec nos contraintes, sont apparues, citons par exemple MadKit [Gut00] développée au LIRMM de Montpellier.

L'expérience acquise lors des essais menés durant l'étude a permis de dégager un ensemble de spécifications devant être respectées par une plate-forme agent, ces spécifications ne décrivent pas un modèle d'agents mais plutôt un cadre d'exécution et un ensemble de services devant être fournis par la plate-forme.

Le cadre d'exécution concerne le SMA dans sa totalité et les capacités de parallélisation transparentes lors de l'exécution. À cet effet, nous utilisons une architecture de type micro-noyau (§7.3 p. 122) qui fournit un ensemble de services minimaux permettant la transparence vis-à-vis de l'implémentation sous-jacente.

Le cadre d'exécution porte aussi sur une structure de contrôle minimale, flexible et standard interne à chaque agent, qui permet de fixer un cadre le moins contraignant possible lors de l'intégration de modules et d'outils au sein de l'agent.

Afin de faciliter le développement d'agents, sans pour autant fixer un modèle agent trop contraignant, la plate-forme fournit un ensemble de services se présentant comme une boîte à outils dans laquelle le concepteur des agents applicatifs sélectionne des fonctionnalités plus ou moins complexes, respectant ainsi le principe de granularité variable.

L'utilisation d'un micro-noyau, d'une structure de contrôle minimale interne à l'agent et d'une boîte à outils reflète une préoccupation visant à contraindre le moins possible le concepteur de l'application au risque de déboucher sur une simple application distribuée. Cette orientation nous semble pour l'instant nécessaire en regard du niveau de maturité de la discipline. En effet, autant les notions de fonction ou de boucle sont communément admises pour les langages de programmation, autant la sémantique, ou le type de mise en œuvre attaché à des concepts comme l'intention ou l'environnement, peut varier d'un SMA à l'autre.

7.2 Caractéristiques et vue générale de la plate-forme

Nous allons évoquer les spécifications volontairement bas-niveau et génériques que doivent respecter notre plate-forme :

1. Granularité variable : on doit pouvoir implanter des agents très légers, donc réactifs ou faiblement cognitifs, ou bien des agents plus lourds et fortement cognitifs. Ainsi, nous ne fixons pas *a priori* un mécanisme de raisonnement pour l'agent. Seule une structure de contrôle se voulant générique et flexible est fournie. Celle-ci pose un cadre d'intégration des savoirs et "savoir faire" de l'agent applicatif. L'élaboration de ce dernier est faite en sélectionnant des fonctionnalités dans une boîte à outils, par instanciation, et en ajoutant les fonctionnalités "utilisateur" liées à l'application.

2. Un grand nombre d'agents : plusieurs dizaines de milliers d'agents doivent pouvoir prendre place au sein du système. Ceci exclut l'utilisation d'un processus par agent pour des contraintes de mémoire mais suggère l'utilisation de processus légers (§7.3.4.1 p. 129) ou *threads*. Cette contrainte exclut aussi l'utilisation systématique d'un modèle cognitif de raisonnement agent impliquant une consommation mémoire importante.

3. Un système distribué : étant donnée la grande quantité de calculs à effectuer, on souhaite profiter de l'aspect intrinsèquement distribué des agents pour répartir le calcul simultanément sur des architectures de type MIMD, *Multiple-Instruction Multiple-Data*. Dans les architectures MIMD, plusieurs flux d'instructions traitent de manière indépendante plusieurs flux de données. Les architectures MIMD se décomposent en deux familles [Tan94] :

a. **Architectures à mémoire partagée**(fig. 7.1), les processeurs disposent d'une mémoire commune à laquelle ils peuvent accéder à travers un bus.

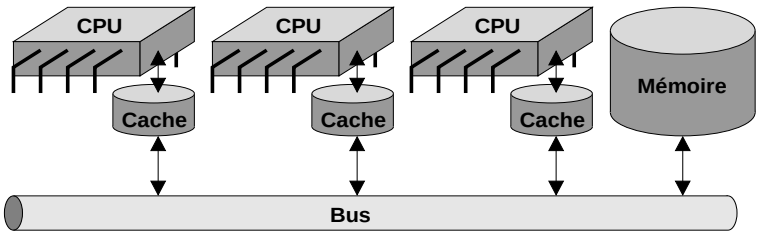


FIG. 7.1 : *Architecture à mémoire partagée.*

Les conflits pour la possession du bus dégradent les performances de ces architectures, on dote alors les processeurs de mémoires caches locales, petites et rapides, dans lesquelles on stocke une portion de la mémoire partagée. Ces mémoires surveillent les données échangées sur le bus afin de maintenir une cohérence globale des données. Ce type d'architecture, désormais relativement courant et moyennement onéreux, se présente pour le grand public sous la forme de plusieurs (souvent 2 ou 4) processeurs généralistes dont les rôles sont symétriques (il n'y a pas de notion de processeur maître ou esclave). On parle alors d'architectures SMP (*Symetrical MutliProcessing*).

b. **Architectures à mémoire distribuée** (fig. 7.2), chaque processeur disposant de sa propre mémoire locale, on obtient un ensemble d'ordinateurs ou de calculateurs à part entière qui possèdent chacun son système d'exploitation, et qui sont reliés via un réseau qui, très souvent, sera local comme Ethernet ou Token ring.

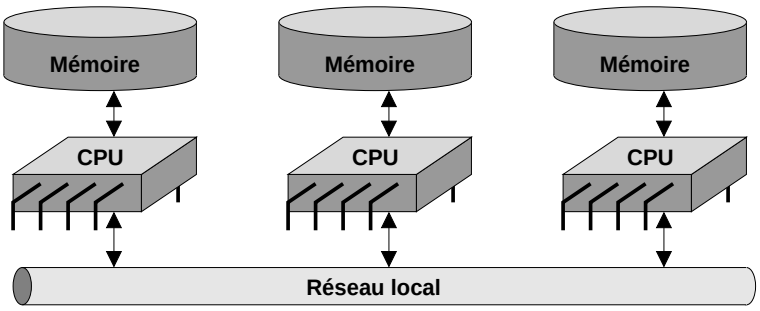


Fig. 7.2 : *Architecture à mémoire distribuée.*

Ce type d'architectures, désormais très courant et peu onéreux, se présente pour le grand public sous la forme de machines (PC ou station de travail) mises en grappes ou *cluster*. Les clusters permettent l'intégration incrémentale de machines hétérogènes et donc une réutilisation des machines plus ou moins obsolètes.

Afin de bénéficier des deux types de distribution, un micro-noyau va fournir une abstraction à l'implémentation sous-jacente permettant aux agents de communiquer d'une manière transparente (fig. 7.3, p. 121).

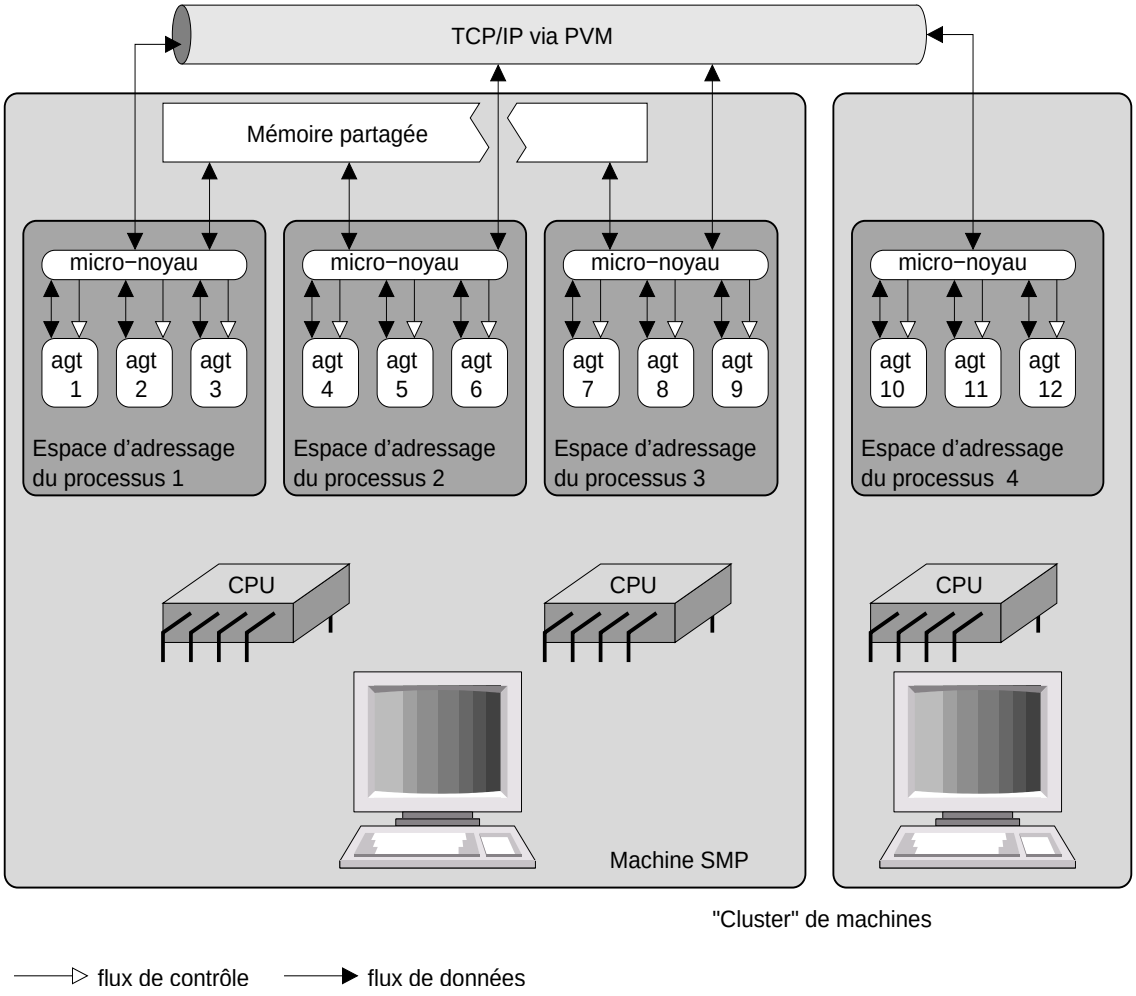


Fig. 7.3 : *Répartition des agents et modes de communication aux sein d'architectures à mémoire partagée (SMP) ; et à mémoire distribuée (cluster).*

4. Mécanismes d'envoi de messages adaptés : il est peu efficace d'utiliser TCP/IP pour faire communiquer deux agents s'exécutant sur la même machine, alors que l'utilisation de la mémoire partagée ou même de l'espace d'adressage commun (si les agents s'exécutent dans le même processus) améliore considérablement les performances. Ainsi, suivant la situation physique respective de deux agents voulant communiquer (fig. 7.3), un mécanisme spécifique d'envoi de messages sera utilisé. C'est une des raisons qui a motivé l'utilisation d'un micro-noyau épaulé d'agents systèmes, comme l'agent postier, au lieu d'une utilisation généralisée d'une implémentation de CORBA[Gei99, COR01], PVM ou MPI.

5. La portabilité permet de déployer le système au sein d'architectures hétérogènes, permettant de faire travailler ensemble des PC sous Windows, Linux ou même des stations de travail. Ceci est un point non négligeable lorsque l'on considère la puissance de calcul potentielle d'une salle de travaux pratiques d'une université. Nous avons donc choisi de développer notre application en Java.

Pourquoi Java ? Java est un langage équilibré, non sans défaut, dont voici les caractéristiques qui ont motivé son utilisation :

- Bien que plus lent (3 à 15 fois) qu'un langage compilé comme C ou C++, l'utilisation de JIT (Just In Time compiler), effectuant une compilation du bytecode java, à la volée au moment de l'exécution, permet de combler une partie de la lenteur inhérente à un langage interprété.
- Java intègre dans le langage des fonctionnalités intéressantes comme les Threads, la sérialisation d'objets et une certaine réflexivité des objets.
- Le JDK (Java Developpment Kit) fournit un ensemble de librairies facilitant le développement. Citons, les structures de données, les librairies réseau ou graphique.
- C'est un langage généraliste, excellent nulle part mais bon partout. Il autorise des traitements bas-niveau (numériques) efficaces grâce à l'existence de types de base (entier, double,...). Il permet aussi une approche plus conceptuelle et exclusivement objet en fournissant des mécanismes standards et spécialisables de manipulation et de comparaison de ces objets (par exemple : traitements sur les chaînes de caractères).
- C'est un langage dynamique, assurant de trouver sur internet des outils répondant à des besoins précis.

7.3 Le micro-noyau et les agents système

Le micro-noyau fournit une abstraction minimale rendant transparente l'implémentation sous-jacente. Le micro-noyau ne présuppose rien sur la nature des agents, par exemple : il n'agit pas comme un environnement observé par les agents. Il ne fait que deux choses :

- fournir un environnement d'exécution aux agents ;
- faire transiter les messages entre agents.

De cette manière, nous respectons la philosophie des micro-noyaux [Tan94], dans le sens où une fonctionnalité supplémentaire prendra la forme d'un nouvel agent inséré dans le système.

Remarque : si d'un point de vue conceptuel, notre approche s'est inspirée des micro-noyaux, les choix technologiques (structures de données, API, etc.) ont été influencés par PVM [Gei94] et le noyau de Linux[1] [Bov01] garantissant ainsi des solutions éprouvées à nos problèmes techniques.

Comme nous le précisons à la section 7.3.4.1 p. 129, l'unité d'exécution d'un agent est le thread. Un processus va donc être une entité regroupant un ensemble d'agents et un micro-noyau (fig. 7.3) supportant l'interaction entre agents et leur fournissant un environnement d'exécution.

[1] Linux étant, bien sûr, un fidèle représentant des noyaux monolithiques, aspect qui a donné lieu à une vive controverse entre Linus Thorvald et Andrew Tannenbaum.

L'objet `MicroKernel` de la figure 7.4 présente les principaux champs et méthodes du micro-noyau. Les méthodes peuvent être comprises comme des appels systèmes exécutés par l'activité de l'agent. Le micro-noyau se doit donc d'être réentrant, plusieurs threads pouvant exécuter simultanément une de ses méthodes. Alors, certaines structures de données sont protégées par un sémaphore booléen, ou *mutex*, garantissant qu'un seul thread à la fois manipule une structure de données protégée.

Les zones de codes critiques se devant être les plus fines possible pour limiter les inter-blocages, il n'y a pas de mutex global portant sur tout l'objet `MicroKernel` mais bien plusieurs mutex verrouillant chacun une structure de données.

```
public final  class MicroKernel{
    /********** Gestion des agents ***********/
     Hashtable localAgents_ht;// table de hachage des agents locaux
     ArrayList localAgents_list;// liste des agents locaux
     public Integer  insertAgent(BaseAgent agent){..}
     public boolean  removeAgent(Integer   agtId){..}
     public boolean  setPriority(BaseAgent agent,int priority){..}

     /********** Gestion des groupes **********/
     Hashtable Groups;// table de hachage des groupes
     public void      joinGroup(Integer   agtId, String groupe){..}
     public void      leaveGroup(Integer  agtId, String groupe){..}
     public ArrayList getAgtOfGroup(String groupe){..}

     /********** Envoi de messages ***********/
     public boolean  sendMsg(Message msg){..}
     public boolean  broadcastMsg(Message msg){..}
     public boolean  broadcastMsg2group(Message msg,String groupe){..}

     /********** Gestion du temps ***********/
     int localTime;
     int cycleActive; //nombre d'agents actifs lors d'un cycle
     public void wakeUpAllActive(){..}
     public void endAction(){..}
     ...
}
```

Fig. 7.4 : *Classe d'implémentation du micro-noyau `MicroKernel` et ses principaux champs et méthodes.*

7.3.1 Gestion des agents et des groupes

Gestion des agents : le micro-noyau tient à jour deux structures de données dans lesquelles il enregistre tous les agents actifs dans le processus courant,

- la table de hachage `localAgents_ht` permet un accès rapide à un agent particulier ;

– la liste `localAgents_list` permet un accès séquentiel rapide à tous les agents. Ainsi, `insertAgent(BaseAgent agent)` permet l'ajout d'un nouvel agent dans le système en lui allouant un nouvel identifiant unique. Le retrait se fait grâce à la méthode `removeAgent(Integer agtId)`.

Gestion des groupes : un groupe est composé d'un nombre arbitraire d'agents, un agent pouvant appartenir à plusieurs groupes. Cette abstraction est utile vis-à-vis du concepteur de l'application en facilitant des opérations : envoi de message, destruction d'agent sur un ensemble d'agents de manière simultanée. Le micro-noyau stocke une table de hachage `Groups` avec, comme clé, le nom du groupe et, comme valeur de retour, la liste des agents appartenant au groupe.

- `joinGroup(Integer agtId, String groupe)` l'agent d'identifant `agtId` se joint au `groupe` ;
- `leaveGroup(Integer agtId, String groupe)` l'agent quitte un groupe ;
- `getAgtOfGroup(String groupe)` retourne la liste des identifants de tous les agents du groupe.

La création d'un nouveau groupe se fait lors du premier appel `joinGroup(...)` sur un groupe n'existant pas déjà.

7.3.2 Les messages

Le format de message (fig. 7.5, p. 124) que nous proposons a comme objectif premier de répondre à des contraintes d'acheminement des messages. On retrouvera des champs identifiant le receveur, l'émetteur, sa date, ainsi qu'une entête pour distinguer les messages système ou utilisateur.

```
public  class Message{
    /* information bas-niveau pour l'acheminement des messages*/
    Integer receiver;//identifiant du receveur
    Integer sender;  //identifiant de l'émetteur
    int     header;  //header du message
    int     date;    //date du message

    /* information niveau applicatif */
    Object  performatif;//performatif applicatif du message
    Object  reply_with; //label à communiquer lors d'une réponse
    Object  in_reply_to;//label communiqué lors d'une réponse
    Object  content;    //contenu du message
    Object  langage;    //langage utilisé dans le contenu du message
     ...
}
```

Fig. 7.5 : *La classe d'implémentation des messages* `Message` *et ses principaux champs.*

Toutefois, notre plate-forme est plus qu'un simple système d'applications distribuées, elle fournit un ensemble de services sous la forme d'une boîte à outils dans laquelle le développeur de l'application peut sélectionner des éléments. Il est donc nécessaire d'étendre sensiblement la sémantique d'un message. Nous avons choisi de calquer les champs relatifs à l'application sur KQML[Fin93], qui est un langage de niveau communication (par opposition au niveau transport) entre agents.

On peut alors envisager à l'avenir un outil prenant la forme d'un spécialiste dont la fonctionnalité est d'interpréter les messages comme étant des messages KQML et KIF [Gen91]. Cependant, en l'absence d'un tel outil, les messages ne sont pas des messages KQML, ils ont le sens que leur donne le développeur de l'application. Le format de message a comme unique rôle de rendre possible le fonctionnement de certains mécanismes (comme les handlers de messages) et de structurer, sans trop contraindre, le développement d'applications.

7.3.3 Envoi de messages

Le micro-noyau est responsable du bon acheminement des messages, la méthode `sendMsg(Message msg)` utilise le champ `receiver` d'un `Message` (fig. 7.5, p. 124) afin de déterminer son destinataire.

- `broadcastMsg(Message msg)` diffuse un message à tous les agents du système ;
- `broadcastMsg2group(Message msg, String groupe)` diffuse un message à tous les agents d'un certain groupe.

Ces deux dernières méthodes utilisent la méthode `sendMsg(Message msg)` sur un ensemble de destinataires qu'elles déterminent.

Remarque : lors de l'appel de `sendMsg(Message msg)`, le compteur `cycleActive` est incrémenté si c'est le premier message reçu par l'agent à ce cycle. Ce compteur est utilisé pour la synchronisation (§7.3.4.2 p. 132), pour savoir combien d'agents seront actifs lors du prochain cycle.

Émission de messages : lors de la distribution d'un message, le micro-noyau regarde si le destinataire est *local*, c'est-à-dire s'il est dans le même processus que l'émetteur. Autrement dit, si l'émetteur et le récepteur du message s'exécutent sur le même micro-noyau, celui-ci peut accéder directement à la boîte aux lettres du récepteur car elle se trouve dans le même espace d'adressage. Si ce n'est pas le cas, le micro-noyau dépose le message dans la boîte aux lettres de l'agent `PostAgent` qui va être responsable de l'envoi de messages vers les agents situés dans des micro-noyaux distants.

```
SendMsg(Message msg)
Agt←localAgents_ht.get( msg.receiver)
si (Agt != null)
– Agt.mailbox.getMsg(msg)
sinon
– PostAgt.mailbox.getMsg(msg)
```

Algo. 7.1 : *L'appel système `MicroKenel.sendMsg(Message msg)` réalisant l'envoi de message.*

L'agent système postier : en respectant la philosophie des micro-noyaux qui consiste à implanter un minimum de fonctionnalités dans le noyau, ces dernières devant être implantées sous forme de processus, la gestion des messages à destination des agents situés dans d'autres micro-noyaux est confiée à un agent `PostAgent` (fig. 7.7, p. 127).

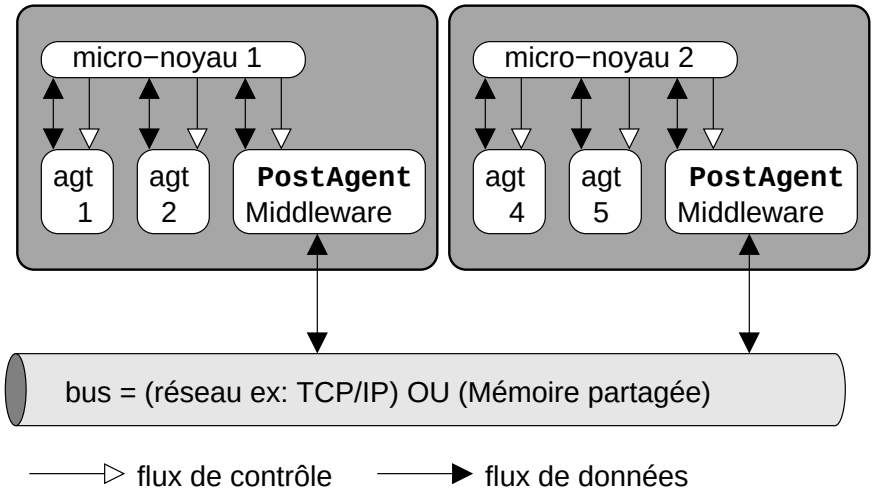


Fig. 7.6 : *L'agent système `PostAgent` : une abstraction permettant de rendre transparents aux agents et au noyau les mécanismes de communication réellement employés.*

L'agent postier joue le rôle d'une "glue" intermédiaire ou *middleware* entre les différents micro-noyaux. Ainsi, le mécanisme d'envoi de messages entre micro-noyaux réellement utilisé est caché derrière l'agent postier, et permet de passer par exemple de PVM à MPI ou CORBA par un simple remplacement de l'agent `PostAgent`.

La boîte aux lettres du `PostAgent` (fig. 7.7, p. 127) & (fig. 7.8, p. 128) est constituée de messages à destination d'agents distants. En la vidant, `PostAgent` répartit les messages dans un tableau de listes de messages `outputMailBox`, dans lequel chaque liste de messages contient les messages à destination d'un micro-noyau particulier. Ce tri est effectué en utilisant une table de routage : `routeTable` pour savoir sur quel micro-noyau s'exécute l'agent destinataire.

```
public  class PostAgent{
 /********** Interaction avec les micro-noyaux distants ***/
 RouteTable routeTable;//table de routage des messages
 ArrayList remoteMK;//table des identifiants des micro-noyaux distants

 Hashtable outputMailBox;//buffers des messages à émettre(classés
                         //par les identifiants des micro-noyaux)
 ArrayList inputMailBox;//buffer des messages reçus

 jPVM      pvmTools;//toolbox PVM
 // sérialisation/désérialisation des messages
 public ArrayList     deserialize(byte[]  messages){....}
 public byte[]        serialize(ArrayList messages){....}
 ...
}
```

Fig. 7.7 : *La classe d'implémentation de l'agent postier* `PostAgent` *et ses principaux champs et méthodes.*

À la fin du cycle courant, l'agent système `PostAgent` va se charger d'envoyer tous les messages présents dans `outputMailBox`. Nous avons choisi d'utiliser PVM pour remplir cette fonctionnalité, mais un autre outil comme MPI ou une implémentation de CORBA peut parfaitement prendre place à cet endroit précis du système sans conséquence sur le reste.

Ainsi, le `PostAgent` sérialise chaque liste de messages (`ArrayList`) présente dans `outputMailBox` grâce à la méthode `serialize(ArrayList messages)`. La sérialisation est un mécanisme permettant de transformer, s'ils ont été prévus à cet effet, des objets ou objets composés en flots d'octets séquentiels, pouvant de cette façon être stockés sur disque ou envoyés par réseau. Java fournit un ensemble d'outils rendant cette tâche aisée.

Une fois chaque liste de messages convertie en autant de flots d'octets, le `PostAgent` va placer chacun de ces flots dans un des buffers PVM destiné à un micro-noyau distant particulier. Cette opération est effectuée à l'aide de l'outil jPVM (le champ `pvmTools`) qui permet une interface entre Java et PVM à l'aide de méthodes natives. En Java, les méthodes natives sont des méthodes écrites dans un langage compilé C ou C++, et présentes dans une librairie dynamique accessible en Java.

Réception des messages : comme nous l'avons évoqué plus haut, les messages locaux sont directement adressés à la boîte aux lettres du destinataire par l'émetteur via l'appel système `sendMsg(Message msg)`. Nous allons donc évoquer la récupération des messages venant de micro-noyaux distants. Le `PostAgent` récupère dans un buffer PVM l'ensemble des messages qui sont destinés aux agents locaux ainsi que certains messages systèmes (de synchronisation) adressés entre agents système. Ces messages se présentent sous la forme de flots d'octets qu'il est nécessaire de

désérialiser à l'aide de la méthode `deserialize(byte[] messages)` qui renvoie une `ArrayList`, c'est-à-dire une liste de messages qui sont stockés dans `inputMailBox`.

`PostAgent` va finalement invoquer l'appel système `sendMsg(Message msg)` sur la liste `inputMailBox` afin de distribuer les messages aux agents locaux du système.

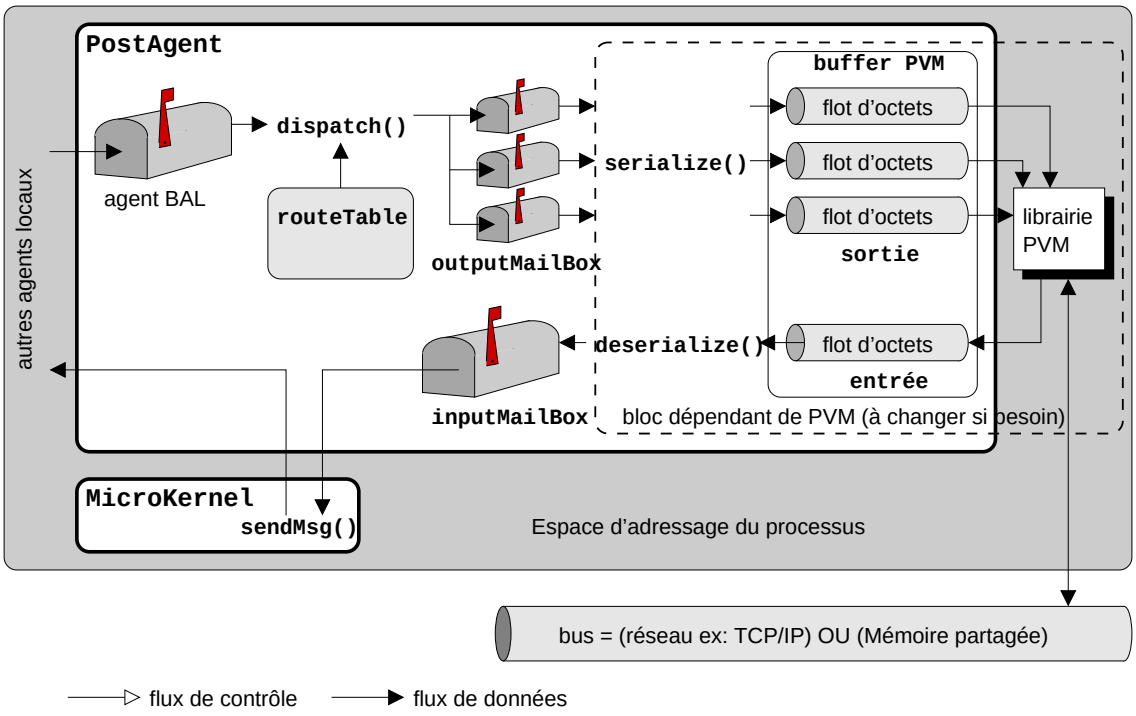


Fig. 7.8 : *Fonctionnement interne de l'agent système `PostAgent` avec une utilisation de PVM.*

Conclusion sur le mécanisme d'envoi de message. Notre approche se justifie par trois points :

1. Lors de la conception d'une application en groupant, dans la mesure du possible, les agents communicant entre eux dans le même processus, on obtient un mécanisme d'envoi de messages très performant car utilisant essentiellement l'espace d'adressage du processus.
2. Le mécanisme d'envoi de messages distants (PVM) est totalement transparent aux agents ; celui-ci peut facilement être remplacé par MPI ou une implémentation de CORBA par modification d'une partie de l'agent système `PostAgent`, voir pointillés (fig. 7.8, p. 128).
3. De même que nous ne voulions pas fournir un modèle type d'agent, nous ne souhaitions pas imposer aux concepteurs d'applications une norme de communication trop structurante comme CORBA et le langage IDL. En effet, il risque d'y avoir une confusion entre les deux niveaux d'abstraction différents que sont un simple service d'échange de messages et la sémantique associée à

ces messages. Cependant, le concepteur d'une application agent reste parfaitement libre d'utiliser une norme comme CORBA et IDL à l'aide de librairies qu'il insère dans ses agents.

7.3.4 Ordonnancement des agents et gestion du temps

Contrairement aux spécialistes des systèmes à base de tableau noir, les agents sont des entités autonomes, ainsi le système d'ordonnancement des agents doit limiter son rôle à répartir les ressources que sont les processeurs, le mieux possible, suivant des consignes (niveau de priorité).

Les agents peuvent être envisagés dans un cadre temps réel, auquel cas, on aura recours à des algorithmes d'ordonnancement adaptés [Cot00]. Dans notre cas, nous allons aborder le problème dans le cadre classique du temps partagé.

7.3.4.1 Processus légers ou threads

Il est important de distinguer deux niveaux conceptuels différents : le niveau concept, ou entité logique, et le niveau de mise en œuvre . Nous avons déjà évoqué cette distinction à propos des messages en opposant le niveau logique de messages entre agents et le niveau mise en œuvre par CORBA ou PVM.

Cette confusion peut aussi apparaître à propos de la notion d'agent (niveau logique) et celle de tâche informatique (niveau mise en œuvre).

En effet, plusieurs agents peuvent parfaitement être exécutés par une même tâche informatique. Toutefois, il faut bien doter les agents d'un moteur d'exécution et il nous semble que la notion de *thread* est bien adaptée car elle permet de respecter les spécifications de granularité variable et de parallélisme d'exécution (voir threads système ci-dessous) énoncées à la section 7.2.

La notion de processus légers ou *threads* est désormais courante dans les noyaux des systèmes d'exploitation modernes, bien que leur mise en œuvre diverge significativement[2]. Un processus est composé d'un espace d'adressage et d'un chemin d'exécution. Voici les principaux éléments de l'espace d'adressage :

1. le segment de code contenant les instructions du programme ;
2. la zone des variables globales ;
3. le "tas" ou zone d'allocation de mémoire dynamique ;
4. la zone contenant des informations système, par ex. : fichiers ouverts, processus fils, etc.

Le contexte ou chemin d'exécution est composé des éléments suivants :

1. les registres du processeur (dont le compteur ordinal qui pointe sur la prochaine instruction à exécuter) ;

[2]Les threads du système Solaris de Sun se distinguent de ceux de Linux.

2. la pile des variables locales des fonctions appelées, contenant aussi les valeurs des compteurs ordinaux avant appel de fonction. Elle sert également de lieu de stockage des registres du processeur lorsque le système d'exploitation préempte le processus.

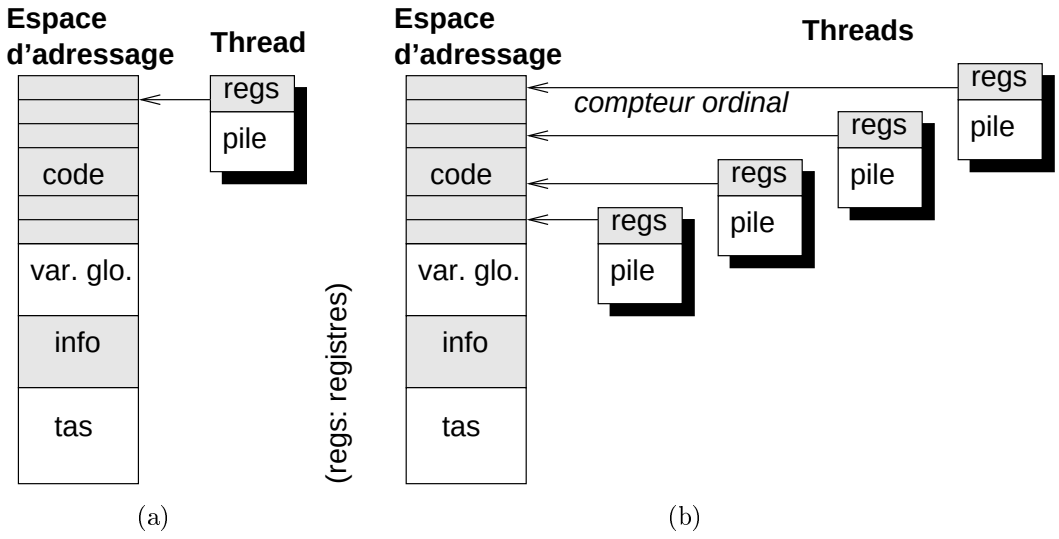


Fig. 7.9 : *(a) Processus classique avec un seul thread ; (b) processus avec plusieurs threads*

On appel "thread" ou "processus léger" (défini par la norme Posix 1003.1c) ce chemin d'exécution. La programmation *multi-threads* est devenue une nécessité avec l'apparition d'applications (comme les serveurs web) nécessitant la création d'un grand nombre d'occurrences du même programme. La duplication des processus, et donc des espaces d'adressage supportant le même programme, consomme inutilement de la mémoire. Ce problème est partiellement résolu avec la capacité des OS (Operating System) de faire partager des pages mémoires communes (segment de code) entre processus. Le changement de contexte entre processus est inutilement coûteux car il nécessite le déchargement et chargement des espaces d'adressages. Les threads d'un processus vont tous partager le même espace d'adressage accélérant le changement de contexte. Ce partage de l'espace d'adressage favorise une communication aisée entre threads sans la nécessité d'avoir recours aux mécanismes de partage de mémoire (IPC Inter Process Communication). Ces deux derniers points font du *thread* un outil adapté à la mise en œuvre du "moteur" des agents. Toutefois, il faut distinguer deux types de threads :

1. Les threads utilisateur sont implantés à l'aide de librairies à l'intérieur d'un processus ne possédant qu'un seul thread système (voir ci-dessous).

L'ordonnanceur (*scheduler*) du système d'exploitation alloue du temps processeur au thread système qui va répartir ce temps, parmi les threads utilisateur, à l'aide d'un ordonnanceur interne s'exécutant en mode utilisateur. Cette approche possède certains avantages :

1. elle offre un moyen pratique au développeur de mettre en œuvre une programmation concurrente entre plusieurs activités ;

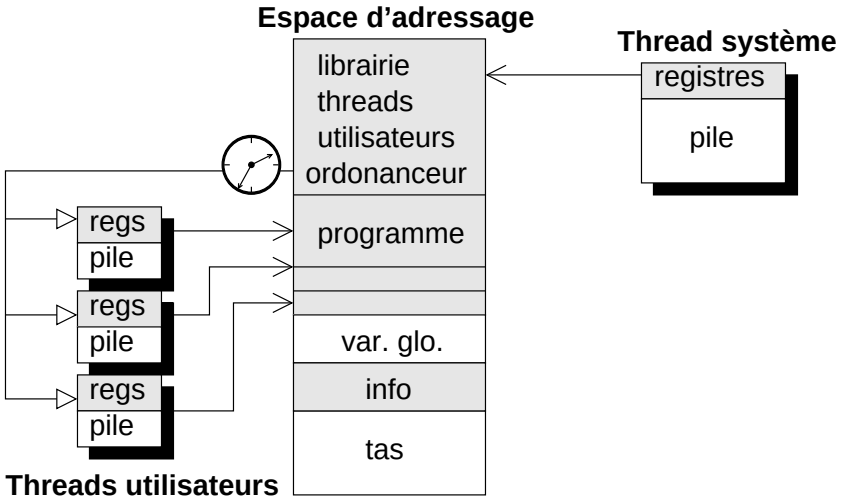


Fig. 7.10 : *Les threads utilisateur.*

2. le temps de commutation de contexte entre threads utilisateur est très court provoquant une augmentation des performances.

Néanmoins, cette approche a des limites :

1. Une E/S (Entrée/Sortie) bloquante lancée par l'un des threads utilisateur va bloquer le thread système, et donc tous les autres threads utilisateur, puisque du point de vue du noyau de l'OS il n'y a qu'un seul thread système.
2. Cette approche ne permet pas de répartir les threads sur plusieurs processeurs puisque, du point de vue de l'OS, il n'y a qu'un seul thread système s'exécutant sur un seul processeur.

2. Les threads système sont répertoriés comme tels au niveau du noyau de l'OS qui prend en charge leur ordonnancement. Une E/S bloquante effectuée par un thread du processus ne bloque pas les autres threads de ce même processus. De plus, deux threads du même processus peuvent s'exécuter de manière physiquement parallèle sur deux processeurs d'une machine SMP (§7.2 p. 119).

Lien avec Java. La classe `Thread` [Oak99] du langage Java fournit un moyen élégant et simple de manipuler les threads en Java. Les possibilités d'actions sur ces threads sont réduites du fait de l'aspect multi-plate-forme du langage empêchant d'utiliser les spécificités d'une plate-forme particulière. La plupart des JVM (Java Virtual Machine) proposent l'utilisation de threads utilisateur appelés *green threads* dont le comportement des threads est plus ou moins similaire [3] quel que soit l'OS sous-jacent. Certaines JVM proposent aussi l'utilisation de threads système appelés *native threads* rendant possible une exécution parallèle des threads dans le cas d'une machine SMP. Dans ce cas, le comportement des threads va varier en fonction de l'OS utilisé, néanmoins une programmation rigoureuse faisant appel à de nombreux

[3] Les spécifications du langage étant très vagues à ce sujet.

points de synchronisation permet d'obtenir un résultat globalement[4] déterministe quel que soit l'OS et la JVM employée.

7.3.4.2 Synchronisation et cycle d'exécution

La gestion du temps est un problème délicat dans un système distribué où les entités peuvent agir les unes sur les autres. Pour résoudre ce problème, on peut avoir recours à un mécanisme de synchronisation globale.

Systèmes globalement synchronisés ou cadencés. Un système cadencé est un système distribué où toutes les entités s'attendent pour faire évoluer un temps commun à l'aide d'une synchronisation globale.

À la date T, toutes les entités consomment leurs messages (ces messages incluent les perceptions faites de l'environnement), elles effectuent un cycle d'exécution incluant le raisonnement (plus ou moins sophistiqué) et le choix des actions à entreprendre qui se traduit par des messages à émettre (ces messages comprennent les actions sur les autres agents et environnement). Ceci fait, tous les messages de toutes les entités sont envoyés et celles-ci passent toutes au même moment, à la date T+1. Si une entité travaille en cadence, elle est assurée que tous les messages envoyés à la date T sont bien partis à cette date et qu'il n'y a pas d'ambiguïté sur la notion de temps. De plus si la transmission de messages est instantanée, à la date T+1, les agents sont certains d'avoir reçu tous les messages partis à la date T.

Systèmes non globalement synchronisés. Dans des architectures non cadencées, les messages sont estampillés avec la date de l'expéditeur. Chaque entité décide à quel moment elle passe à la date suivante et il n'y a qu'elle qui connaît réellement sa date. Une entité peut recevoir des messages passés (elle est en avance sur d'autres), des messages futurs (elle est en retard sur d'autres), des messages présents (elle est à la même date que d'autres). Le problème de ce type d'architecture est de savoir quand effectuer une action. Comment gérer le cas où un message (de date T) essentiel au choix d'une action (menée à l'instant T+1) arrive à l'instant T+2 ? Le message peut être simplement ignoré or cette solution peut mener à certaines incohérences. Imaginons deux agents A_1 et A_2 effectuant chacun une pression simultanée de forces égales et opposées sur un bloc B situé en bord de table. B reçoit à T+1 l'action (datée T) de A_1 le faisant tomber de la table, puis il reçoit à T+2 l'action (datée T) de A_2 qu'il ignore. Logiquement, le bloc aurait dû rester sur la table. Il faut donc pouvoir envisager des retours en arrière, si jamais dans le futur on réceptionne un message passé important. Effectuer ces retours en arrière est très coûteux, surtout si l'entité en a influencé d'autres ; cette approche n'est donc pas viable.

Dans la pratique, les concepteurs de plate-formes SMA proposent très souvent des architectures cadencées avec des messages dont la durée du trajet est considérée comme instantanée. Il est plus facile de réaliser ces systèmes et ils fournissent de

[4]Si ce terme peut paraître choquant à certains il traduit la réalité de l'informatique appliquée qui n'a de formelle que la phase de spécification (laquelle est, dans ce cas, relativement vague).

bonnes propriétés. Nous proposons donc d'intégrer cette cadence et cette instantanéité des messages comme propriétés intrinsèques de notre plate-forme.

Proposition d'un système cadencé : l'agent système `Chronos`. Une plate-forme multi-agents est un système faisant cohabiter et interagir des agents. Comme le signale Ferber [Fer95], ces agents peuvent être de nature purement communicante, situés dans un environnement, ou les deux à la fois. Le cycle d'exécution de la plate-forme est en conséquence celui présenté à l'algorithme (algo. 7.2 p. 134) et (fig. 7.11, p. 133).

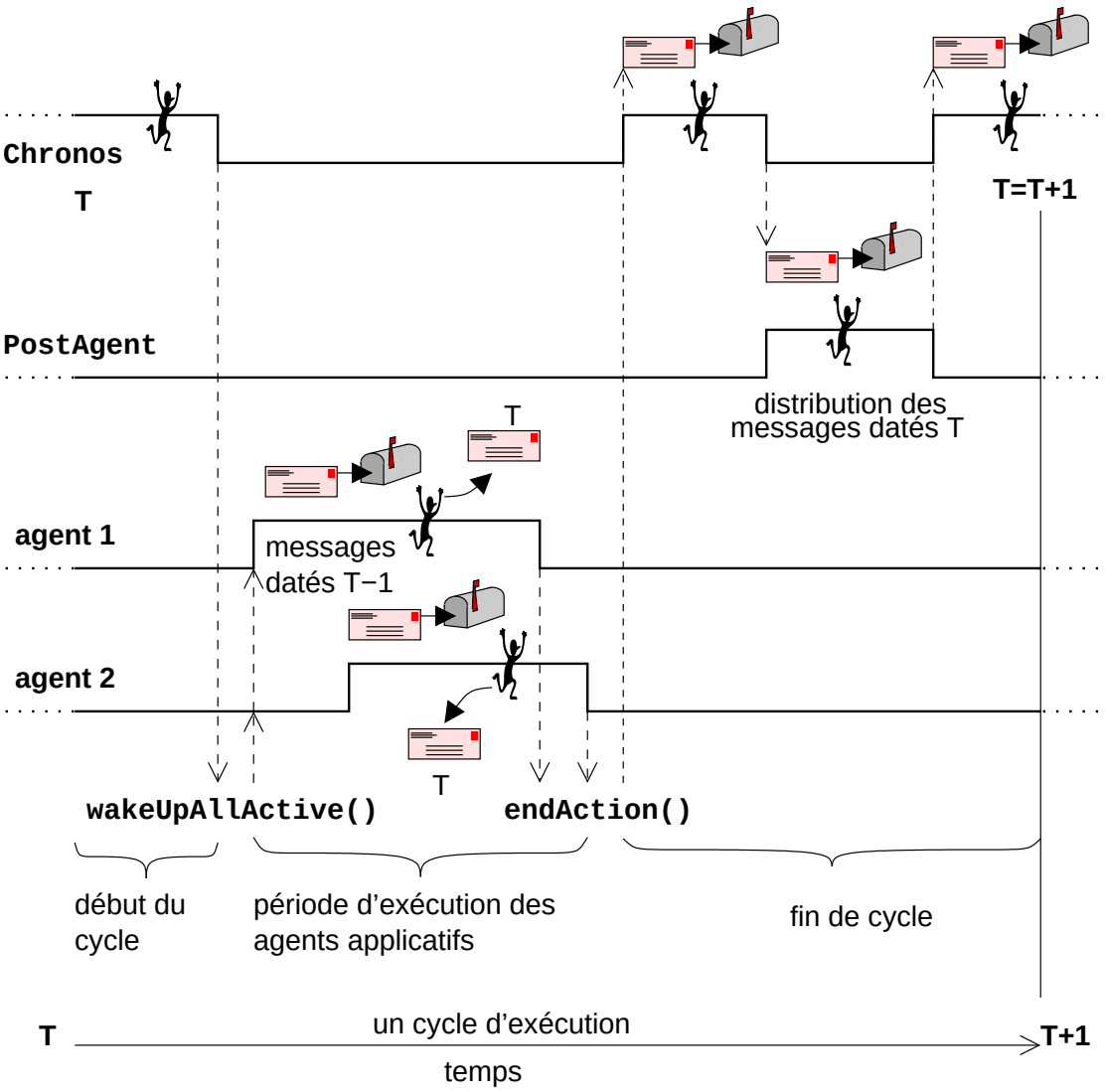


FIG. 7.11 : *Cadencement global des agents : un cycle d'exécution.*

Cycle synchro globale (date T)

1. L'agent `Chronos` réveille tous les agents ayant reçu un message au cycle précédent. Il effectue pour ceci l'appel système `wakeUpAllActive()`, cette méthode parcourt la liste `localAgents_list` du noyau (fig. 7.4, p. 123) et envoie un signal de réveil par la méthode `notify()` à tous les threads dont l'agent a reçu un message (champ `messageInMailBox` de la classe `MailBox` (§7.4.2 p. 144).
2. Les agents consomment leurs messages (datés T-1) incluant les perceptions du(des) environnement(s), ils raisonnent, ils choisissent les comportements à déclencher et les déclenchent, provoquant l'envoi de nouveaux messages datés T. Ces messages incluent les actions menées sur le(les) environnement(s). Puis chaque agent exécute l'appel système `endAction()`, décrémentant à chaque fois le compteur `cycleActive`. Si ce compteur est à 0, un message de contrôle est envoyé à l'agent `Chronos` qui provoquera son réveil.
3. L'agent `Chronos` est réveillé par un message de contrôle indiquant que tous les agents actifs lors de ce cycle ont fini leur exécution. Il va maintenant s'assurer que tous les messages devant être délivrés pour le cycle prochain sont bien arrivés. Pour cela, il envoie un message à l'agent `PostAgent`.
4. `PostAgent` est réveillé par un message lui indiquant qu'il doit vérifier que tous les messages destinés à des agents distants (sur d'autres micro-noyaux) sont bien partis (et arrivés) et que tous les messages en provenance d'agents distants et à destination d'agents locaux sont bien distribués. Ceci fait, l'agent `PostAgent` envoie un message de contrôle à l'agent `Chronos`.
5. `Chronos` va se synchroniser avec les agents `Chronos` des autres micro-noyaux, par envoi de messages et accusé de réception, ceci fait, il passe à la date T+1 (variable `localTime`) et retourne en 1.

Algo. 7.2 : *Un cycle de synchronisation globale mettant en jeu les agents systèmes `Chronos` et `PostAgent`.*

7.3.5 Conclusion sur le micro-noyau et les agents système

Nous proposons une architecture basée sur un micro-noyau aidé de deux agents système, `PostAgent` et `Chronos`, modifiables à souhait qui effectuent les tâches système périphériques. Ces deux agents système sont semblables aux autres agents, à ceci près qu'ils ont une utilisation d'appels système plus intensive et complexe. De plus, le moindre message dans leurs boîtes aux lettres les réveillent, contrairement aux autres agents.

Le modèle d'exécution articulé autour de threads cadencés par `Chronos` permet une distribution physique de l'exécution des agents sur plusieurs processeurs si la JVM supporte bien les threads natifs/système. Dans le cas contraire, on peut toujours utiliser les threads utilisateur qui donneront de meilleures performances sur une machine monoprocesseur.

Lors de la conception de l'architecture, nous avons favorisé l'ouverture et la simplicité au détriment d'une architecture offrant des modèles souvent complexes et parfois trop structurants et contraignants pour le développement d'applications. L'expérience nous a appris qu'à vouloir trop faire, on est tenté de développer des fonctionnalités complexes, mal pensées et peu généralisables débouchant sur des "usines à gaz" impossible à mettre au point et dont on utilise 5% des fonctionnalités. Nous pensons que cette situation évoluera le jour où les SMA auront mûri permettant l'émergence de concepts clairement identifiables.

7.4 L'agent

Comme précisé précédemment, nous n'avons pas souhaité fournir un modèle structurant le mode de raisonnement des agents. En effet, des agents simples et proches du réactif peuvent suffire pour certaines applications tandis que d'autres nécessiteront des agents très cognitifs à forte granularité. De plus, les développeurs ont des visions très différentes des choses. Ainsi, une architecture très structurante peut s'avérer un bon squelette d'assemblage pour son concepteur mais un véritable casse-tête pour un autre développeur qui peinera à y faire entrer ses idées.

Nous proposons, sous la forme d'une classe Java, un squelette d'agent minimal `BaseAgent` établissant une circuiterie entre les messages, les modules de traitements (spécialistes) et la mémoire de l'agent (tableau noir local). Nous proposons des implémentations plus ou moins complexes de ce squelette, tout comme une boîte à outils dans laquelle le développeur peut sélectionner des modules pour façonner son agent d'une manière adaptée à l'application.

7.4.1 Structure de contrôle

Les principaux éléments composants l'agent sont :

1. des modules de traitement que nous nommerons *spécialistes*, qui dépendent de l'application, lesquels sont fournis par le développeur de celle-ci ;
2. une mémoire locale que nous nommerons tableau noir local, qui contient des données relatives à l'application, cette mémoire peut être inexistante pour des agents purement réactifs ;
3. des messages en provenance et en partance.

La structure de contrôle a pour objectif d'établir un squelette supportant et orchestrant l'interaction entre tous ces éléments.

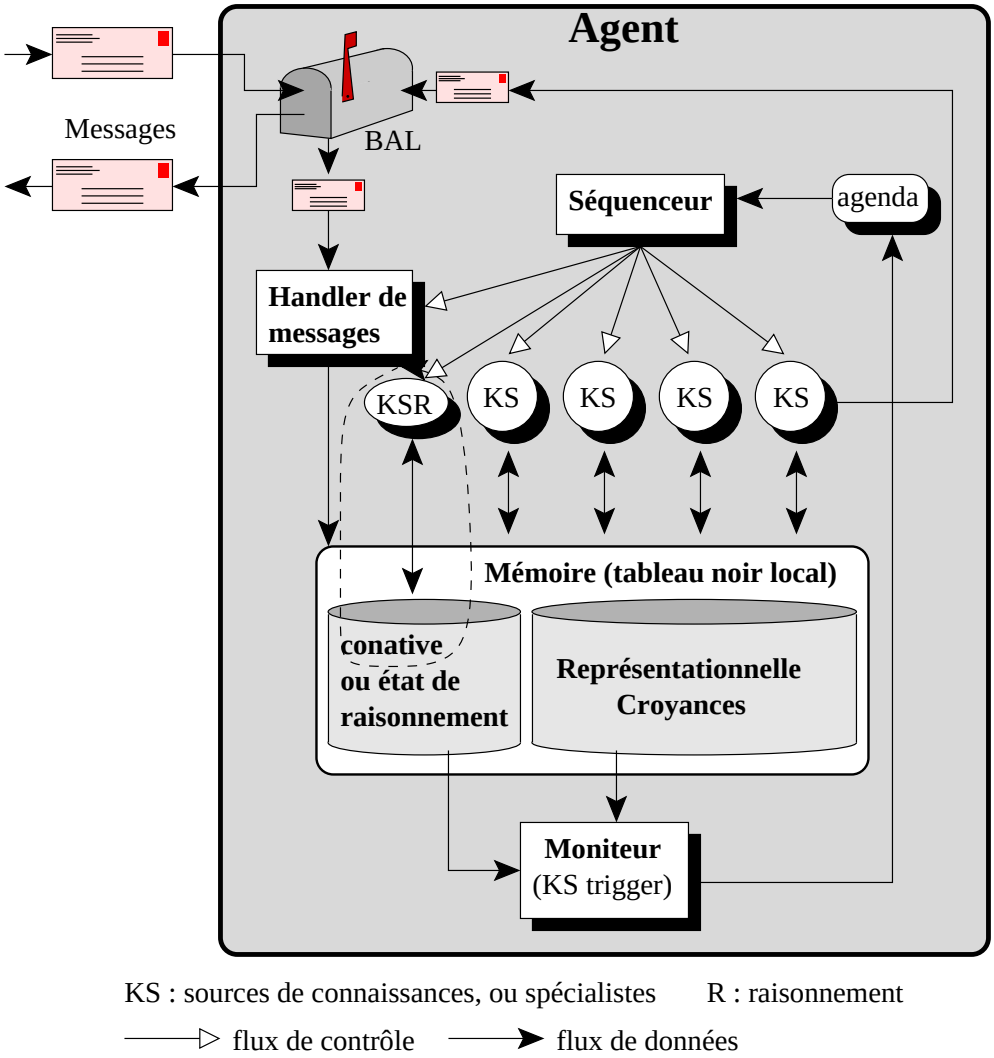


FIG. 7.12 : *Squelette de la structure de contrôle interne de l'agent.*

Nous avons choisi le tableau noir [Erm80] et (§5.2.1 p. 85) comme structure de contrôle de l'agent (fig. 7.12, p. 136). Il permet la mise en œuvre d'agents à granularité et aux capacités de raisonnement variables. En effet, nous souhaitons à court terme intégrer des agents d'interprétation nécessitant des fonctionnalités cognitives plus étendues que celles utilisées par les agents de l'application actuelle (qui se situent dans le bas de l'échelle des agents cognitifs).

Chaque composant de cette structure de contrôle se présente sous la forme d'une classe de base que le développeur des agents applicatifs spécialise par héritage pour obtenir le comportement souhaité. Le squelette générique de l'agent manipule les classes de base des comportements spécialisés grâce au polymorphisme d'héritage. Nous allons évoquer les différents composants intervenant dans cette structure de contrôle.

7.4.1.1 Les *handlers* de messages

La sémantique et le traitement des messages sont des critères distinguant les familles de langages qui y ont recours :

- Pour les langages objet, un message est un ordre traité de manière synchrone ;
- Pour les langages acteur [Agh86], un message est un ordre traité de manière asynchrone ;
- Pour les langages agent, un message est une information à caractère descriptif, conatif, ou autre, que l'agent a le choix de prendre en compte. Si le message est une requête de traitement que l'agent souhaite satisfaire, celle-ci sera effectuée de manière asynchrone.

Notre plate-forme fournit donc naturellement un service de messages asynchrone que les agents peuvent ignorer. Les *handlers* de message sont le premier élément d'une circuiterie reliant messages et traitements asynchrones des messages.

```
public class MessageHandler{
    protected String performatif;
    public void readMessage(Message msg);
}
```

Fig. 7.13 : *La classe `MessageHandler` prend en charge les messages : c'est l'étape de perception de l'agent.*

Pour chaque type de message que l'agent peut traiter, le concepteur de l'application doit créer un handler spécialisant la classe `MessageHandler` (fig. 7.13, p. 137). Pour cela, il précise à l'aide du champ `performatif` les messages qui seront interceptés par ce handler, ce symbole devant correspondre au champ `performatif` de la classe `Message` (fig. 7.5, p. 124).

Il donne aussi une spécialisation de la méthode `readMessage(Message msg)` précisant la manière dont sont traités les messages. Si aucune spécialisation n'est donnée, la méthode `readMessage(Message msg)` fournie par défaut, place ces messages dans une zone particulière du tableau noir local de l'agent.

Les handlers effectuent donc un filtrage de la boîte aux lettres, ils peuvent avoir une fonction de reformulation, de tri sur l'importance ou d'effacement des messages. Cela peut permettre d'éviter à l'agent le traitement de messages inutiles reflétant le "bruit"[5] du système. Les handlers correspondent à la fonction de perception de l'agent, qui, comme nous venons de le voir, peut être attentive ou sensible à certains types de messages, cette sensibilité pouvant varier dynamiquement en fonction de l'état mental de l'agent c'est-à-dire des données présentes dans le tableau noir local.

7.4.1.2 Le moniteur

Le moniteur surveille le tableau noir local : mémoire représentationnelle, conative et états de raisonnement, à l'aide de modules pouvant être procéduraux ou déclara-

[5]Messages considérés comme inutiles pour l'agent.

tifs. En effet, comme nous le verrons (§7.4.4 p. 148), un spécialiste (unité de traitement de l'agent) est composé de deux parties : un module attentif `conditions()` et le corps d'exécution du spécialiste `run()`. Chaque nouveau spécialiste est signalé au moniteur qui va désormais prendre en compte la méthode `condition()` de ce spécialiste. Cette méthode peut être de nature procédurale (souvent suffisante pour des agents à faible granularité), ou composée d'un ensemble de règles d'inférence.

Lorsque la partie condition d'un spécialiste est vérifiée, le moniteur communique à l'agenda du séquenceur le spécialiste déclenchable.

7.4.1.3 L'agenda

L'agenda est une simple liste contenant l'ensemble des spécialistes déclenchables à un moment donné, triés suivant leur degré de priorité. L'agenda fournit deux méthodes : `insert(KnowledgeSource ks)` pour l'insertion d'un nouveau spécialiste suivant sa priorité, et `getFirst()` qui retourne le spécialiste le plus prioritaire. Ce dernier est un comportement par défaut pouvant parfaitement être spécialisé au besoin.

7.4.1.4 Le séquenceur

Le séquenceur est le centre de contrôle de l'agent, il est implémenté dans la classe `BaseAgent` (fig. 7.14, p. 139) qui est la classe de base de tous les agents, fournissant des fonctionnalités par défaut qui peuvent être spécialisées au besoin pour une application.

```
public class BaseAgent{
    protected Integer   agtId;//identifiant de l'agent
    protected Agenda    agenda;
    protected Monitor   monitor;
    protected DataBase  blackboard;
    protected MailBox   mailBox;

    /********** Gestion des spécialistes ******/
    protected KnowledgeSources knowledgeSources;
    public void addKS(KnowledgeSource ks){...}

    /********** Gestion des handlers **********/
    protected MessageHandlers  messageHandlers;
    public void addHandler(MessageHandler mh){...}

    /********** séquenceur ********************/
    public void runAgent(){...}
    public void scheduleKS(){...}
    ...
}
```

Fig. 7.14 : *La classe `BaseAgent` : classe de base des agents applicatifs à spécialiser au besoin.*

Les deux méthodes principales du séquenceur sont :

1. `scheduleKS()` (algo. 7.3 p. 139) : elle est responsable de l'ordonnancement des spécialistes. La méthode par défaut fournie exécute les spécialistes déclenchables par ordre de priorité. Le séquencement s'effectue selon une politique en largeur d'abord, c'est-à-dire que l'on exécute tous les spécialistes déclenchables avant d'interroger de nouveau le moniteur (appel de `monitor.run()`). Cette méthode est réitérée tant qu'il y a des spécialistes déclenchables.

ScheduleKS()
tant que(quelque-chose-à-faire)
- `monitor.run()`
- **si** (`agenda.isEmpty()`)quelque-chose-à-faire←FAUX
- **sinon**
  tant que(`!agenda.isEmpty()`)
  - Spécialiste← `agenda.getFirst()`
  - Spécialiste.`run()`

  fin tant que

fin tant que

Algo. 7.3 : *Ordonnancement des spécialistes suivant les priorités, selon un mode largeur d'abord.*

2. `runAgent()` (algo. 7.4 p. 140) : elle représente la boucle d'exécution principale de l'agent. (1.) Elle commence par suspendre l'agent sur sa boîte aux lettres en attente de nouveaux messages et de l'appel système `wakeUpAllActive()` venant de l'agent `Chronos`. Puis (2.), tous les messages sont filtrés à l'aide des handlers des messages. (3.) elle appelle la méthode implantant l'essentiel de l'activité de l'agent qui va contrôler l'exécution des spécialistes. Enfin, (4.) tous les messages en partance sont envoyés. Finalement (5.), l'agent exécute l'appel système `endAction()` signifiant au noyau qu'il a fini son cycle d'exécution.

RunAgent()
tant que(l'agent est vivant)
1. `mailBox.wait()`
2. `mailBox.flushInputMessages(messageHandlers, defaultHandler)`
3. `scheduleKS()`
4. `mailBox.flushOutputMessages()`
5. `microKernel.endAction()`

fin tant que

Algo. 7.4 : *Boucle principale d'exécution de l'agent.*

7.4.1.5 Exemples de mise en œuvre du contrôle

Nous venons de décrire l'ensemble des éléments composant (fig. 7.12, p. 136) le squelette d'un agent de notre plate-forme. Afin de mettre en œuvre les agents applicatifs pour une application particulière, le développeur procède en deux étapes :

1. spécialiser si besoin les composants (handler, séquenceur, moniteur) afin d'obtenir le contrôle et le mode de raisonnement souhaité.
2. insérer les modules applicatifs sous la forme de spécialistes.

Nous proposons des pistes de conception d'agents utilisant divers formalismes de contrôle. Puis à la section 7.4.1.6 p. 141, nous détaillerons plus particulièrement un modèle de contrôle à base de réseau de Pétri.

Agent piloté par un Automate Fini Déterministe (AFD) : un tel agent est composé :

- d'un ensemble fini d'états stockés en mémoire locale (§7.4.3 p. 146) ;
- d'un ensemble d'événements qui sont des messages traités par les handlers de messages (§7.4.1.1 p. 137) ;
- d'un ensemble de spécialistes (§7.4.4 p. 148) reflétant les actions possibles, la fonction de transition et celle d'action. En effet, l'ensemble des fonctions `conditions()` des spécialistes est inséré dans le moniteur pour former la fonction de transition, et l'ensemble de leurs fonctions `run()` forme la fonction d'activité.

Agent piloté par des règles : un tel agent peut très naturellement être mis en œuvre en dotant chaque spécialiste de règles reflétant les conditions d'activation du spécialiste. Ces règles seront intégrées au moniteur.

Agent à capacité cognitives étendues : si les mécanismes de base ne sont pas assez puissants pour mettre en œuvre une stratégie de raisonnement élaborée, le développeur de l'application utilisera la souplesse de l'architecture à base de tableau noir. Il pourra créer des spécialistes de priorités supérieures aux autres qui seront dédiés au raisonnement (spécialiste KSR de la figure 7.12 page 136).

Si, par exemple, il souhaite implanter un agent BDI (§5.3.3.3 p. 92) : les handlers réalisent la fonction "beliefs-revision" (algo. 5.2 p. 92). Trois spécialistes de raisonnement (de priorité décroissante) sont créés : "select-options", "deliberate" et "planification" réalisant les fonctions du même nom de l'agent BDI . Le dernier spécialiste après création d'un plan d'exécution insère celui-ci directement dans l'agenda du séquenceur.

7.4.1.6 Modélisation à l'aide de réseau de Pétri

Afin de faciliter la lisibilité des chapitres 8 et 9 dans lesquels la modélisation est faite à l'aide de Réseau de Pétri (RdP), nous allons lors de cette section entrer plus en détail sur la modélisation du contrôle de l´agent par RdP.

La modélisation des états et structures de contrôle de l'agent à l'aide d'un RdP est adaptée au parallélisme inhérent aux activités des agents. De plus, les RdP permettent aussi une modélisation aisée des protocoles régissant les interactions entre agents.

Exemple de modélisation d'un spécialiste (cf. définition 5). La figure 7.15 présente la modélisation d'un spécialiste chargé d'envoyer l'agent au lit si un message `heure-tardive` est reçu et que l'agent est fatigué.

Pour le prototypage d'une application il est souhaitable d'utiliser un système à base de règle pour la mise en œuvre d'une telle modélisation. Cette pratique facilite la lisibilité la maintenabilité et l'implémentation du modèle RdP de contrôle de l'agent.

La figure 7.15 présente donc une mise en œuvre du spécialiste `Aller-au-lit` sous la forme d'une règle.

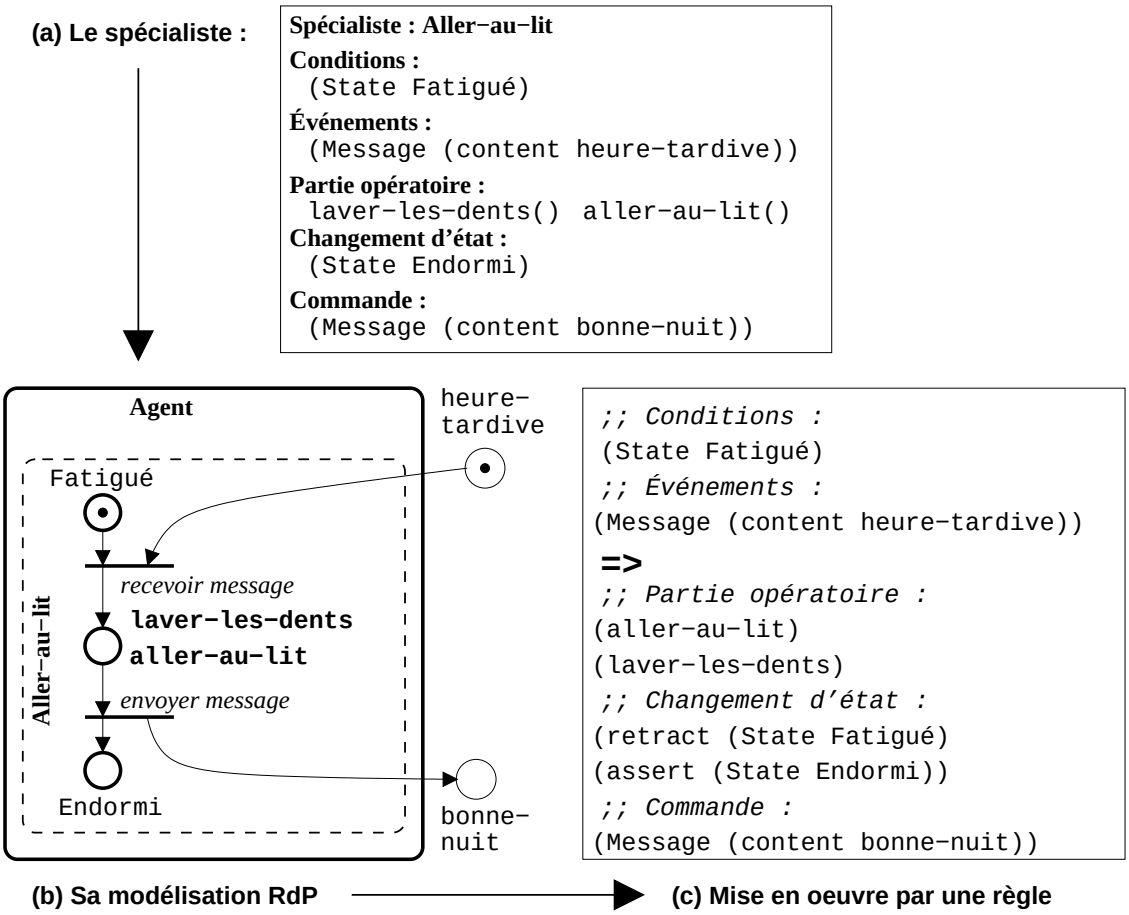


Fig. 7.15 : *(a) Un spécialiste ; (b) la modélisation du contrôle du spécialiste "Aller au lit" par RdP. Nota : les transitions de nos RdP sont toutes tirées (leur fonction de réceptivité est vraie) ; (c) mise en œuvre par une règle.*

Exemple de modélisation d'un comportement (cf. définition 6). Une des activités essentielles d'un agent est d'envoyer et recevoir des messages à ses voisins (accointances). Nous allons modéliser le comportement "aller se coucher" (fig. 7.16, p. 143) dans une version "familiale". Supposons qu'une famille d'agents, tous fatigués, reçoive le message `heure-tardive`. Ils se lavent les dents, vont au lit, puis se souhaitent "bonne nuit". Chacun devant attendre une réponse de tous les membres de la famille avant de s'endormir. Ceci permet d'illustrer une **synchronisation** entre agents.

Cette façon d'aller se coucher nécessite des envois et des réceptions de messages, elle ne peut donc plus être modélisée sous une forme atomique. On va alors utiliser un comportement composé de quatre spécialistes (fig. 7.16, p. 143).

1. Spécialiste : ALLER-AU-LIT
   - **Conditions :**`(State fatigué)`
   - **Événements :**`(Message (content heure-tardive))`
   - **Partie opératoire :**laver-les-dents() ; aller-au-lit() ;
   - **Changement d'état :**`(State couché)`
2. Spécialiste :SOUHAITER-BONNE-NUIT
   - **Conditions :**`(State couché)`
   - **Commande :**$\forall a_i \in$ `m.d.f.` **sendMsg**`(Message (content bonne-nuit))`
   - **Changement d'état :**`(State Écouter-réponse)`
3. Spécialiste :ÉCOUTER-RÉPONSE
   - **Conditions :**`(State Écouter-réponse)`
   - **Événements :**
     $\forall a_i \in$ `m.d.f.` **receiveMsg**`(Message (content bonne-nuit))`
   - **Changement d'état :**`(State réponses-entendues)`
4. Spécialiste :S'ENDORMIR
   - **Conditions :**`(State réponses-entendues)`
   - **Partie opératoire :**éteindre-lumière() ; compter-les-moutons() ;
   - **Changement d'état :**`(State endormi)`

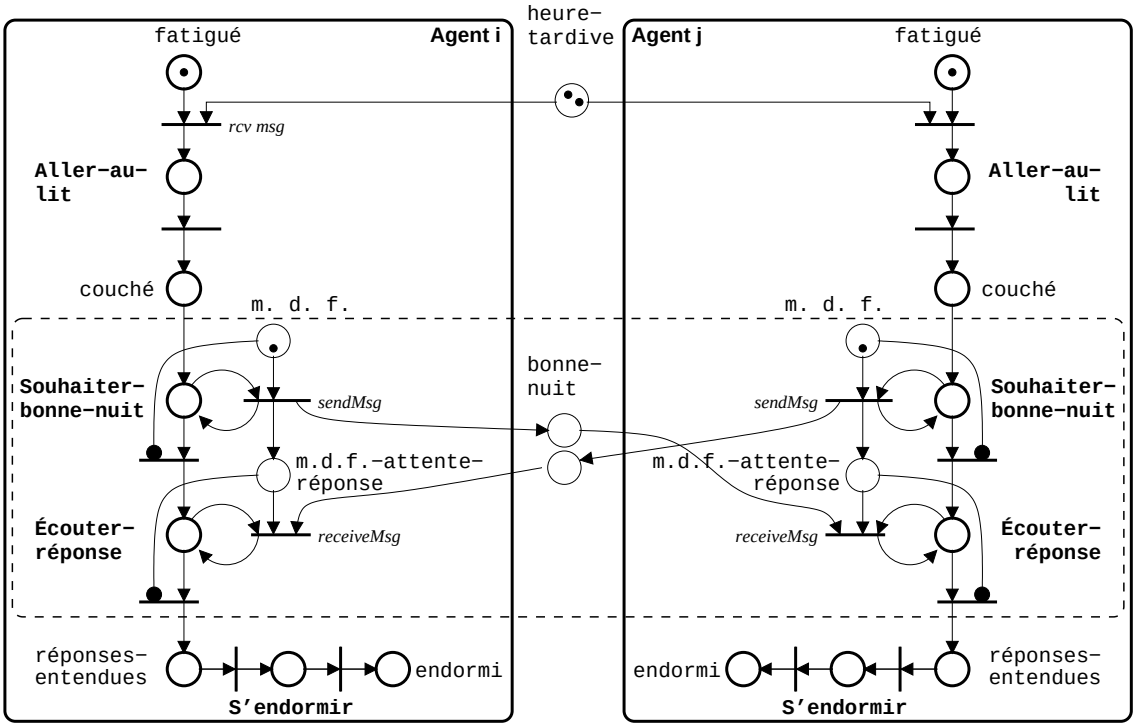


FIG. 7.16 : *Modélisation du comportement "aller se coucher" par RdP. Légende : m.d.f. : les autres membres de la famille (un seul est représenté) ; sendMsg : envoyer un message ; receiveMsg : recevoir un message. En pointillés, les places et transitions qui réalisent la synchronisation entre les membres de la famille.*

Remarques sur la modélisation La partie gauche de la figure 7.17 présente un agent qui envoie trois messages à trois agents, attend et réceptionne trois messages venant de ces trois agents.

Dans la plupart des situations rencontrées dans les chapitres 8 et 9, le nombre de voisins d'un agent est indéterminé à l'avance. Les protocoles d'interaction modélisés par RdP étant identiques pour chaque agent et afin de rendre le RdP lisible, nous aurons recours à un modélisation locale par RdP. Ainsi, seule sera modélisée l'interaction d'un agent avec un seul de ses voisins comme le montre la partie droite de la figure 7.17. Le RdP global du système est obtenu par interconnexion des RdP locaux, et par ajout d'un nombre de jetons dans les places appropriées (ici la place `voisin`) qui correspond au nombre réel de voisins de l'agent.

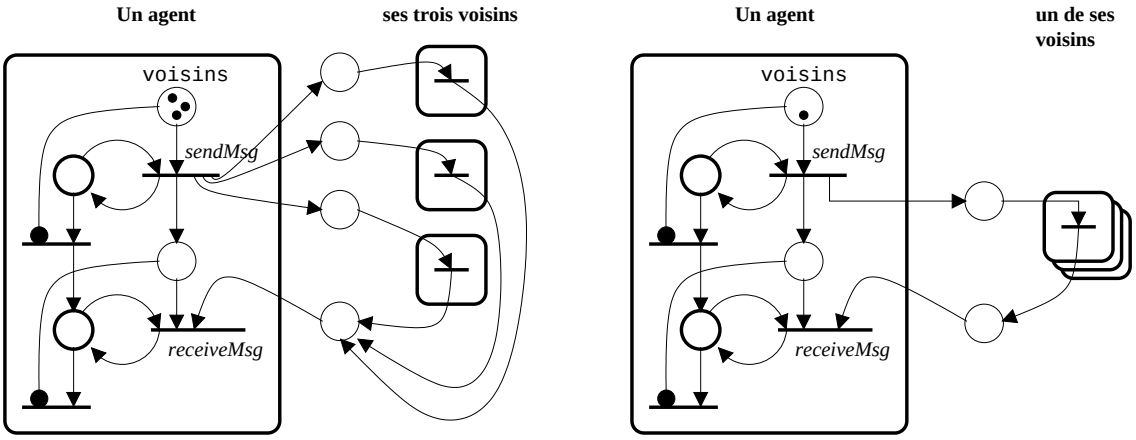


Fig. 7.17 : *À gauche : modélisation globale par RdP ; à droite : RdP réduit à une modélisation locale.*

La modélisation proposée a un **caractère descriptif**, l'objectif est de pouvoir visualiser graphiquement l'évolution de l'état d'un agent en fonction des actions entreprises et des messages reçus. Cette modélisation repose sur deux hypothèses :

- Au niveau application, les agents sont supposés "honnêtes et obéissants" : un agent recevant un message nécessitant une réponse s'exécutera.
- Au niveau transport des messages, le canal de communication est supposé fiable et déterministe : les messages ne se perdent pas.

7.4.2 La boîte aux lettres

La boîte aux lettres est un élément essentiel de l'agent remplissant deux fonctionnalités :

1. envoi et réception des messages ;
2. synchronisation.

L'émission des messages : lorsqu'un spécialiste souhaite envoyer un message, il appelle la méthode `mailbox.sendMsg(Message msg)` ; celle-ci va stocker le message dans le buffer `outputMessages`. Ce tampon (buffer) de message ne sera vidé qu'à la fin du cycle d'exécution de l'agent lorsque le séquenceur appellera la méthode `flushOutputMessages()` (algo. 7.5 p. 145). Cette méthode va invoquer, pour chaque message du buffer, l'appel système `MicroKernel.sendMsg()` (algo. 7.1 p. 126) réalisant effectivement l'envoi du message.

```
public class Mailbox{
  ArrayList inputMessages;
  ArrayList outputMessages;
  boolean   messageInMailBox;
  public synchonized void wait();
  public synchonized void sendMsg(Message msg){..}
  public synchonized void getMsg(Message msg){..}
  public synchonized flushOutputMessages(){..}
  public synchonized flushInputMessages(
                            MessageHandlers messageHandlers,
                            MessageHandler  defaultHandler){..}
}
```

Fig. 7.18 : *La classe d'implémentation de la boîte aux lettres de l'agent.*

flushOutputMessages()
tant que(`!outputMessages.isEmpty()`)
– message← `outputMessages.removeFirst()`
– `MicroKernel.sendMsg(` message)
fin tant que

Algo. 7.5 : *Algorithme d'envoi de tous les messages en partance de l'agent.*

La réception des messages : l'appel système `MicroKernel.sendMsg()` (algo. 7.1 p. 126), avec un message à destination d'un agent local, provoque l'appel de `getMsg(Message msg)` sur la boîte aux lettres de l'agent destinataire. La fonction `getMsg` insère le message dans le buffer `inputMessages` et positionne le booléen `messageInMailBox` à VRAI. Ce booléen va servir à l'appel système `wakeUpAllActive()` invoqué par l'agent système `Chronos` pour savoir si l'agent doit être réveillé car il a des messages en attente.

On remarque sur la figure 7.18 que `getMsg` est une méthode synchronisée ainsi que la plupart des méthodes de la boîte aux lettres. En effet, comme précisé précédemment, la boîte aux lettres joue le rôle de lien avec l'extérieur. Plusieurs threads sont susceptibles d'y accéder de manière concurrente, elle doit donc être protégée par un sémaphore booléen ou mutex.

Lorsque l'agent est réveillé, après la méthode `wait()`, tous les messages datés T-1 à destination de l'agent sont dans la liste `inputMessages`. Le séquenceur appelle donc la méthode `flushInputMessages(...)` qui va procéder au filtrage et à la lecture des messages à l'aide de la table des handlers de messages fournis en paramètres. Si un message ne dispose pas d'un handler correspondant, il est traité par le handler par défaut qui supprime simplement ce message. Ce handler par défaut peut être redéfini au besoin pour effectuer une analyse plus approfondie du message afin de savoir s'il doit être réellement supprimé.

```
flushInputMessages(MessageHandlers messageHandlers,
MessageHandler defaultHandler)
tant que( !inputMessages.isEmpty())
– message← inputMessages.removeFirst()
– handler← messageHandlers.get(message.performatif)
– si (handler != null) handler.readMessage( message )
– sinon defaultHandler.readMessage( message )
fin tant que
```

Algo. 7.6 : *Méthode de perception de l'agent qui filtre à l'aide d'handlers de messages sa boîte aux lettres.*

L'attente des messages : au début de chaque cycle d'exécution (algo. 7.4 p. 140), l'agent se suspend sur sa boîte aux lettres avec la méthode `wait()`. Si l'agent a reçu des messages au cours du cycle précédent (ou qu'il a été initialisé avec un message d'amorce à sa création), le champs `messageInMailBox` est positionné à VRAI, alors il sera réveillé par l'agent système `Chronos` et amorcera son cycle d'exécution.

7.4.3 Tableau noir local

Le tableau noir local joue le rôle de la mémoire de l'agent, dans laquelle il enregistre toutes formes d'informations qu'il juge utiles et qui ont déjà été transformées par la perception c'est-à-dire filtrées par les handlers de messages. Ces informations peuvent être à caractère conatif (intentions, désirs) ou représentationnel (croyance sur l'environnement, et à propos des autres agents).

Les actions sur le tableau noir inspirées par les requêtes SQL des SGBD [Gar01] sont effectuées à travers de quatre familles de méthodes :

- Insertion d'objets :
  `Object insert(`<*Objet*>`)` : ajout d'un nouvel objet inséré dans une table portant le nom de la classe de l'objet[6]. Si aucune table portant le nom de la classe n'existe, une nouvelle table est créée. Cette méthode retourne une clé unique autorisant un accès rapide à l'objet.
- Retrait d'objets :
  - `Object delete(`<*clé*>`)` : retrait rapide d'un objet par sa clé ;
  - `Object delete_from_where(`<*from_name, where_name, where_val*>`)` : retire les objets de classe *from_name* dont le champ *where_name* vaut *where_val*.
- Mise à jour d'objets :
  - `Object uptade(`<*clé,Objet*>`)` : l'objet stocké à *clé* est remplacé par *Objet* ;
  - `Object uptade_set_where(`<*set_name, set_val, where_name, where_val*>`)` : le champ *set_name* des objets dont le champ *where_name* vaut *where_val* est positionné à la valeur *set_val*.
- Récupération d'objets :

[6] On utilise la réflexivité de Java pour obtenir ce nom.

- `Object select(`<*clé*>`)` : récupération rapide d'un objet par sa clé ;
- `Object[] select_from_where(`<*from_ name, where_ name, where_ val*>`)` : récupère les objets de classe *from_ name* dont le champ *where_ name* vaut *where_ val*. Le résultat obtenu (un tableau d'objets) peut être soumis à de nouvelles sélections par une méthode identique prenant en plus comme argument le tableau d'objets issu d'une précédente sélection.

Il n'était pas dans nos objectifs de fournir des fonctionnalités évoluées des SGBD classiques. Nous fournissons un outil simple et pratique permettant aux différents spécialistes de communiquer de manière indirecte.

Comme nous le verrons (§7.4.5 p. 149), il est possible d'ajouter une base de connaissance déclarative à l'agent. Dans ce cas et si l'objet à insérer dans la base de données l'autorise, il sera aussi visible et modifiable depuis la base de connaissance. On obtient ainsi un mécanisme de gestion des données homogène et transparent, utilisable via un formalisme procédural et déclaratif.

7.4.3.1 Les croyances de l'agent

Les croyances de l'agent regroupent les informations à caractère représentationnel à propos de l'agent lui-même, de l'environnement et des autres agents. Ces informations dépendent donc de l'application considérée ; cependant, nous fournissons des structures de base pouvant être spécialisées pour les notions les plus courantes.

Les accointances. Nous allons évoquer ici la structure d'accointance. Cette dernière est la relation abstraite organisationnelle minimale liant deux agents : un agent B est une accointance de A si A connaît B. Ce lien est étiqueté d'un ou plusieurs types spécialisant la nature sémantique de cette relation abstraite.

```
(a) ********************************************************
// une instance de la classe Acquaintance :
public class Acquaintance{
    Integer  id=43;
    String[] types={''collègue'', ''ami''};
    public void addTypes(String rel){...}
}

(b) ********************************************************
;; Et son équivalent déclaratif :
(Acquaintance (id 43) (types collègue ami))

(c) ********************************************************
Un exemple de requête : selection des accointances amis :
Acquaintance[] mes_amis =
select_from_where(``Acquaintance'', ``types'', ``ami'')
```

Fig. 7.19 : *(a) une instance de la classe* `Acquaintance` *du point de vue procédural et en (b) du point de vue déclaratif ; (c) un exemple de requête sur la base de données locale.*

La partie (a) de la figure 7.19 fournit un exemple d'accointance, mais comme nous le verrons (§7.4.5 p. 149), les objets manipulés par l'agent peuvent aussi l'être de manière déclarative. Cette vision des objets est présentée dans la partie (c) de la figure 7.19.

7.4.4 Spécialistes

Les spécialistes (fig. 7.20, p. 148) représentent le savoir faire de l'agent c'est-à-dire sa connaissance opératoire à propos du domaine applicatif.

```
public class KnowledgeSource{
    int priority;
    public boolean conditions(){return false;}
    public String  getConditionsRules(){return null;}
    public Object  run(){return null;}
    public String  getRunRules(){return null;}
}
```

Fig. 7.20 : *La classe* `KnowledgeSource` *définissant l'interface (et des comportements par défaut) des spécialistes de l'agent.*

La plate-forme spécifie simplement l'interface qu'ils doivent respecter afin de pouvoir être intégrés à l'agent. Les spécialistes sont composés de deux méthodes principales :

- Les méthodes `conditions()` et `getConditionsRules()` définissent les conditions d'application du spécialiste. Lorsque l'on déclare un nouveau spécialiste dans l'agent `addKS(KnowledgeSource ks)` de la classe `BaseAgent`, le système intègre la méthode `conditions()` au moniteur, et il regarde si la méthode `getConditionsRules()` retourne des règles ; si tel est le cas et que l'agent dispose d'un système à base de règles, ces règles sont insérées dans le moniteur.
- Les méthodes `run()` et `getRunRules()` définissent le savoir faire du spécialiste qui sera déclenché si les conditions ont été remplies et si le spécialiste est élu par le séquenceur. Si `getRunRules()` retourne des règles, et que l'agent dispose d'un système à base de règles, ces règles sont insérées dans le groupe des règles opératoires de l'agent.

Après avoir abordé la notion de spécialiste d'un point de vue technique et bas-niveau, nous allons préciser la sémantique des deux concepts se référant aux composantes opératoires de l'agent que sont le spécialiste et le comportement.

Il sera fait, lors des chapitres 8 et 9, souvent référence à la notion de spécialiste et de comportement qui représentent, à une échelle différente, les composants actifs de l'agent.

Définition 5 (Spécialiste) *Un spécialiste est la brique logicielle concrète de connaissances opératoires faisant évoluer l'état de l'agent. Cette brique est modélisée sous la forme d'un quintuplet :*

$$(\textit{Conditions, Événements, Partie opératoire, Changement d'état, Commande})$$

Le spécialiste est déclenchable si des conditions portant sur l'état de raisonnement (Conditions) et sur l'apparition d'événement (message, nouvelle croyance) sont réalisées. Le déclenchement du spécialiste provoque l'exécution indivisible de la partie opératoire. Celle-ci peut provoquer un changement d'état et des envois de messages que nous assimilons à des commandes.

Définition 6 (Comportement) *Un comportement est un composant logique abstrait représentant un savoir faire plus ou moins complexe et non atomique de l'agent. Un comportement est donc mis en œuvre par un certain nombre de spécialistes.*

7.4.5 Système à base de connaissance

En fonction des besoins de l'application, le développeur peut, s'il le désire, embarquer dans chaque agent une base de connaissances. Cela permet une représentation déclarative des connaissances à court terme et à long terme de l'agent avec les avantages présentés (§4.6.1.3 p. 73), on peut aussi l'utiliser comme langage de prototypage rapide des agents lors du développement de l'application qui, après validation, sont implantés en procédural.

Les inconvénients doivent cependant être pris en compte, citons : l'augmentation du poids mémoire de l'agent et du temps nécessaire à l'accomplissement d'un cycle de raisonnement. Il est impossible de quantifier exactement ces augmentations tant elles dépendent de l'application. Toutefois, afin de donner un ordre de grandeur, lors

de nos expériences avec des agents faiblement cognitifs pesant 50 Ko de mémoire et exécutant un cycle de raisonnement en 0.1 seconde, l'utilisation d'un système à base de connaissance, comprenant quelques dizaines de règles, triple leur poids (150 Ko) et leur temps (0.3 seconde) d'exécution à fonctionnalités égales.

Jess (Java Expert System Shell) [Jes01] est l'outil que nous avons choisi afin de mettre en œuvre la base de connaissance. Il a été inspiré par CLIPS [Cli01] un autre moteur d'inférence d'ordre 1, la syntaxe des règles des deux outils est d'ailleurs compatible. Comme CLIPS, il utilise l'algorithme Rete (§4.6.1.4 p. 74) & (fig. B.1, p. 260) comme pré-compilation des règles afin d'accélérer leurs déclenchements, il possède de plus de nombreuses fonctionnalités intéressantes :

- Il peut également fonctionner en chaînage arrière.
- Il peut s'appliquer directement sur les objets java, grâce à l'emploi des java beans.
- Il est aisément embarquable dans un application et fournit une API très complète permettant de le contrôler depuis du code java. Il permet aussi très naturellement de faire appel aux méthodes java.
- On peut lui ajouter un module intégrant la logique floue.

Notre plate-forme permet une intégration au besoin de Jess, celle-ci est transparente aux composants ne l'utilisant pas puisqu'ils manipulent toujours les mêmes objets java. Cette intégration permet d'effectuer du *pattern matching* sur les objets de la base de données, pouvant provoquer une modification de celle-ci. Le *pattern matching* est effectué par des paquets de règles fournis par le développeur de l'application à travers les spécialistes, le moniteur ou n'importe quel composant de l'agent.

Chapitre

8

La société d'agents organisée en pyramide irrégulière

Dans le chapitre 6, nous avons justifié l'utilisation des SMA pour la conception de l'architecture logicielle de contrôle d'un système de perception et, plus particulièrement, de segmentation. Il y a en effet adéquation entre les caractéristiques des SMA et certaines propriétés (adaptation locale, coopération...), qu'il semble pertinent de prendre en compte.

De plus, les SMA sont l'expression informatique d'un courant de pensée fournissant de nouvelles méthodologies d'analyse des systèmes, dans lesquelles il semble enrichissant d'inscrire notre approche. Nous proposons donc une méthodologie d'analyse en trois points :

1. Description globale et structurelle de l'organisation regroupant les agents.
2. Description locale et fonctionnelle des agents composant le système (chap. 9).
3. Analyse globale et fonctionnelle dans laquelle l'on vérifie que l'ensemble des interactions locales, réalise bien ce pour quoi le système est conçu (chap. 10).

Cette dernière analyse n'est possible que si l'on a correctement spécifié les deux premiers points. Nous proposons comme élément organisationnel la pyramide irrégulière, qui va imposer sa structure à la population d'agents afin de garantir un comportement globalement contrôlable et convergent des agents.

Dans un premier temps, nous préciserons l'algorithmique des graphes (utilisés par les pyramides irrégulières) dans un cadre d'exécution distribuée (problèmes de synchronisation, etc.).

Puis nous proposons une transposition du graphe vers une organisation d'agents.

Enfin, nous détaillerons les processus d'initialisation de la pyramide, ainsi que le contrôle global qui intervient lors du passage d'un niveau (de la pyramide) à son suivant.

8.1 Introduction

Comme nous l'avons évoqué au chapitre 6, la famille des agents situés dans l'image[1] procédant par coopération augmentative et par influences mutuelles fournit un cadre extrêmement riche et adapté à la mise en œuvre :

- de stratégies coopératives ;
- de focalisation et d'adaptations locales dans l'image ;
- de prise en compte des informations *a priori* et donc du modèle.

On perçoit pour les raisons évoquées ci-dessus l'intérêt de porter la problématique de segmentation dans le cadre des systèmes complexes pour bénéficier des concepts issus des disciplines (§4.2.3 p. 55) qui étudient le holisme, la systémique, l'émergentisme, l'auto-organisation et plus concrètement les SMA (chap. 5 p. 83).

Cependant, la démarche qui consiste à prendre comme cadre d'analyse l'une de ces disciplines en s'interdisant toute approche analytique du problème peut paraître déroutante et n'est pas satisfaisante.

Comme le suggère Marvin Minsky ([Min88]) et (§5.2.2 p. 86), nous effectuons une analyse en trois étapes :

1. Une description de l'organisation liant les agents entre eux et du contrôle social résultant. C'est le problème abordé dans ce chapitre.
2. Une description du fonctionnement de chaque agent considéré individuellement. Cette analyse du contrôle et des comportements de l'agent fait l'objet du chapitre 9.
3. Une analyse plus globale de la combinaison des interactions locales afin de mieux comprendre ce que fait le système. Ce sera l'un des thèmes évoqués lors du chapitre 10.

Ce chapitre va donc décrire une **organisation** fournissant un cadre formel structurant l'activité des agents et permettant, si possible, une exécution distribuée. Les pyramides irrégulières remplissant ces critères, voilà pourquoi nous proposons de les utiliser comme une organisation **structurant et régulant** la société d'agents.

Dans un premier temps, nous allons reformuler l'algorithmique de traitement des pyramides dans un cadre distribué. Dans un deuxième temps, nous nous attacherons à transposer cette base algorithmique formelle dans le cadre d'un système SMA. L'organisation de ce SMA et le contrôle social obéiront aux lois de la pyramide, garantissant un comportement globalement contrôlable et convergent.

[1] Voir typologie des architectures logicielles à base de SMA (§5.5 p. 95).

8.2 Une algorithmique distribuée de traitement des pyramides

L'établissement des éléments de transposition nécessite de préciser les mécanismes du traitement pyramidal. En conséquence, dans cette section, nous reformulons avec précision l'algorithmique des pyramides dans un cadre d'exécution distribuée sur machine à mémoire partagée (§7.2 p. 119). Nous nous appuierons sur les principes énoncés lors du chapitre 3 dédié aux pyramides.

8.2.1 Les structures de données

Toute l'algorithmique des pyramides irrégulières s'articule autour d'un graphe d'adjacence de région : $G^k(S^k, A^k)$ pour chaque niveau k, et de son sous-graphe : le graphe de similarité $Sim^k(S^k, B^k)$. Ces deux graphes utilisent les mêmes sommets, ils diffèrent par les arêtes les reliant.

Ils peuvent être codés simultanément par la structure `sommet` (fig. 8.1, p. 154). Nous utilisons, pour représenter les deux graphes, deux structures de liste d'adjacences [Cor94, p. 458] associées à chaque sommet. Cette représentation favorise une représentation homogène des graphes non-orientés (graphe d'adjacence de région), orientés (graphe de similarité avec seuillage local) et pondérés (graphe de similarité). Cette structure contient :

- les attributs photométriques et géométriques de la région représentée par le sommet ;
- la liste chaînée `adjacents` des arcs sortant dans le graphe d'adjacence $G^k(S^k, A^k)$. Ce graphe étant non-orienté, les arcs sont symétriques pour former des arêtes : $s_j \in \texttt{adjacents}(s_i) \Leftrightarrow s_i \in \texttt{adjacents}(s_j)$;
- la liste chaînée `similaires` des arcs sortant dans le graphe de similarité $Sim^k(S^k, B^k)$;
- un pointeur vers le sommet père `père` ; ainsi qu'une liste des sommets fils `fils` au niveau inférieur ;
- les variables booléennes p et q ;
- un pointeur vers une structure contenant la fenêtre englobante des champs récepteurs et un champ de bits 2D de la taille de cette fenêtre où les bits à 1 représentent les champs récepteurs de la région. Cette représentation se prête naturellement à une utilisation par des opérateurs de traitement d'image (comme ceux fournis par PANDORE [Pan01]) qui peuvent être appliqués avec un masque.

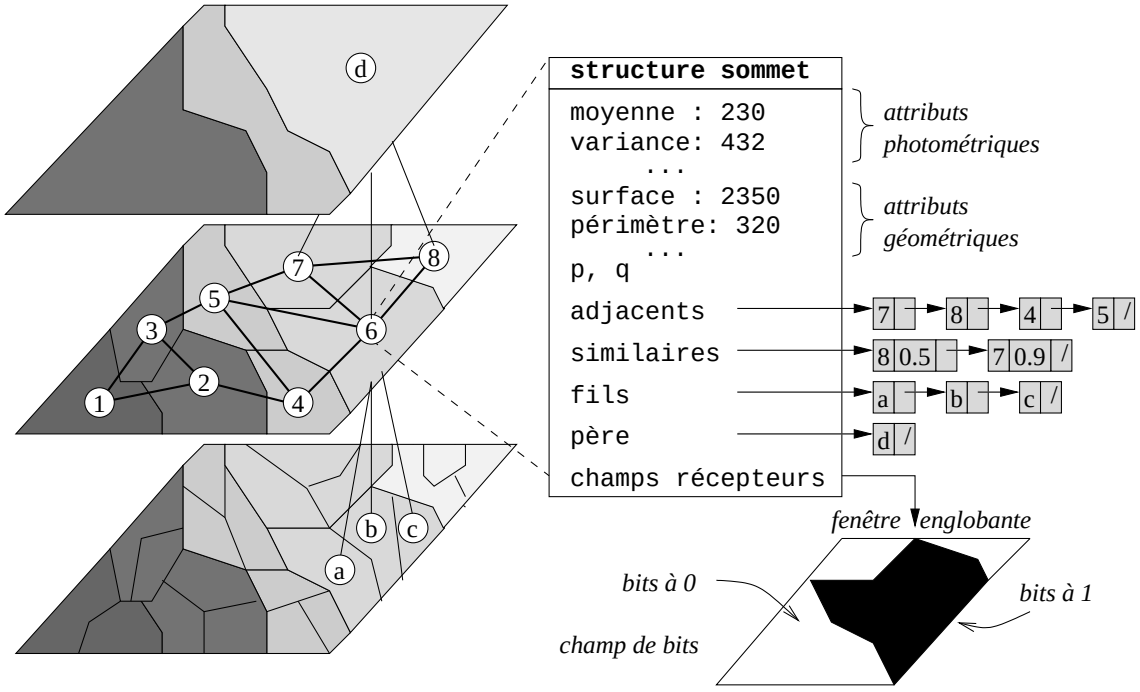


Fig. 8.1 : *La pyramide de graphes d'adjacence de régions (RAG) et la structure d'un sommet du graphe.*

8.2.2 Construction du graphe d'adjacence

Les sommets du niveau 0 (S^0) peuvent être obtenus en associant un sommet à chaque pixel. On peut aussi effectuer une pré-segmentation grossière avec un algorithme de type "split". Pour les autres niveaux, les sommets sont les survivants (§8.2.4 p. 155) de la décimation effectuée au niveau inférieur.

Les sommets étant déterminés, il faut les relier entre eux pour constituer le graphe d'adjacence du niveau k, c'est-à-dire $G^k(S^k, A^k)$. Pour le niveau 0, on supposera que l'on dispose d'une procédure Adjacence selon champs récepteurs(). Pour les niveaux suivants, les sommets adjacents au sommet courant sont les pères des sommets adjacents aux fils du sommet courant.

Construit adjacence(s_i)
si ($k = 0$) Adjacence selon champs récepteurs(`champs récepteurs`)
sinon
pour tout($s_j \in$ `fils`(s_i))
– **pour tout**($s_k \in$ `adjacents`(s_j))
  – *frère* $\leftarrow$ `père`(s_k)
  – **si** (*frère* $\neq s_i$ **et** *frère* $\notin$ `adjacents`(s_i))
    – `adjacents`(s_i)$\leftarrow$`adjacents`(s_i)+*frère*
    fin si
  fin pour tout
fin pour tout

Algo. 8.1 : *Construction des relations d'adjacence d'un sommet.*

8.2.3 Construction du graphe de similarité

Nous allons décrire (algo. 8.2 p. 155) l'algorithme de construction du graphe de similarité $Sim^k(S^k, B^k)$ selon un seuil local $T_l(s_i)$ dont les principes sont expliqués (§3.6.1.2 p. 43).

Rappel : $h(s_i, s_j)$ reflète une distance et donc une similarité entre les sommets s_i et s_j, cette distance va être comparée au seuil de similarité local et pondérer les arcs[2] du graphe de similarité.

Construire similarité(s_i)
calculer $T_m(s_i)$ (voir §3.6.1.2)
$T_l(s_i) = \min(T_m(s_i), T_g)$
pour tout($s_j \in$ `adjacents`(s_i))
– **si** ($h(s_i, s_j) \leq T_l(s_i)$) `similaires`($s_i$)$\leftarrow$`similaires`($s_i$)+($s_j, h(s_i, s_j)$)
  /*les arcs (s_i, s_j) du graphe de similarité sont pondérés par la $h(s_i, s_j)$*/
fin pour tout

Algo. 8.2 : *Construction du graphe de similarité de manière distribuée (à partir de chaque sommet).*

8.2.4 Décimation sur le graphe de similarité

L'algorithme 8.3 p. 157 décrit une procédure itérative de décimation stochastique ou adaptée à l'image. Cette procédure appliquée sur le graphe de similarité respecte les deux règles énoncées (§3.5.2 p. 39). Les sommets survivants ont leur variable booléenne p positionnée à 1.

[2]Le seuil étant local, le graphe est orienté ; c'est pour cette raison que l'on parle d'arcs et non d'arêtes.

Problèmes posés par l'orientation du graphe de similarité $Sim^k(S^k, B^k)$: ce graphe étant calculé à partir de seuillages locaux, il est orienté :

$$(s_i, s_j) \in B^k \nRightarrow (s_j, s_i) \in B^k$$

Or, cette orientation peut poser certains problèmes : il se peut, par exemple, qu'un sommet s_i ne survive pas car son voisin similaire[3] s_j est survivant. Or, il se peut que s_j ne considère pas s_i comme lui étant similaire[4], on dira [Bra95] que s_j ne peut pas absorber s_i. On peut donc se retrouver dans la situation d'un sommet non survivant qui ne peut être absorbé pas aucun de ses voisins survivants.

Braviano [Bra95] propose de relaxer les deux règles 1 & 2 (§3.5.2 p. 39) afin de garantir la convergence du système. Ainsi, les deux règles relaxées deviennent :

Règle 1' Deux sommets ayant une relation d'absorption entre eux ne peuvent pas survivre en même temps :

$$(s_i, s_j) \in B^k \wedge (s_j, s_i) \in B^k \Rightarrow \neg(s_i \in S^{k+1} \wedge s_j \in S^{k+1})$$

Règle 2' Chaque non-survivant s_i de G^k doit avoir un voisin similaire survivant qui puisse l'absorber :

$$\forall s_i \in S^k : s_i \notin S^{k+1} \Rightarrow \Gamma^+_{Sim^k}(s_i) \cap S^{k+1} \neq \emptyset$$

Où $\Gamma^+_{Sim^k}(s_i) = \{s_j \in S^k : (s_i, s_j) \in B^k\}$ est l'ensemble des successeurs de s_i, dans notre cas l'ensemble des sommets similaires à s_i.

Or cette relaxation n'est pas suffisante, dans certains cas particuliers[5] le noyau[6] ne peut être déterminé. Aussi Braviano propose de relaxer encore la règle 1' en autorisant l'ajout de nouveaux sommets.

Nous proposons une approche analogue moins optimale mais qui simplifie grandement l'algorithmique de décimation. Elle part du principe que pour permettre le rattachement d'un sommet s_i à un sommet s_j, les deux sommets doivent se considérer mutuellement similaires.

Cette approche consiste donc à retirer l'orientation du graphe de similarité c'est-à-dire qu'un arc (s_i, s_j) est retiré si l'arc (s_j, s_i) n'existe pas.

$$\exists(s_i, s_j) \in B^k \wedge (s_j, s_i) \notin B^k \Rightarrow B^k = B^k - (s_i, s_j)$$

La conséquence de ce principe "prudent" sera de diminuer le taux de décimation et donc de provoquer la création d'un plus grand nombre de niveaux. Les risques de sur-segmentation existent mais ils peuvent être opposés aux risques de sous-segmentation de l'approche orientée. De plus, ce principe simple facilite le travail, mené par le traiteur d'image, sur l'interprétation des résultats obtenus.

[3] $(s_i, s_j) \in B^k$
[4] $(s_j, s_i) \notin B^k$
[5] Présence de circuit d'ordre impair.
[6] Équivalent du stable maximal des graphes orientés.

La synchronisation entre les tâches. L'aspect distribué de cette procédure impose l'utilisation d'une fonction :
SYNCHRONISE(s_i, `sync-list`(s_i), `S`) dont le rôle est de synchroniser le sommet s_i avec les sommets présents dans `sync-list`(s_i) sur un état `S` donné. Une telle synchronisation est nécessaire à chaque fois que l'activité (thread) associée au sommet courant utilise une variable (x_j, p_j et q_j) d'un sommet voisin. La synchronisation permet de s'assurer que la variable utilisée est bien à jour.

Décimation distribuée (s_i)
- `S` $\leftarrow 0$; $k \leftarrow 1$
- $x_i \leftarrow$ tirage aléatoire suivant une loi uniforme **ou** valeur adaptée à l'image
- SYNCHRONISE(s_i,`adjacents`(s_i),`S++`)
- $\forall s_j \in$`similaires`(s_i)
  si ($s_i \notin$`similaires`(s_j)) `similaires`(s_i)$\leftarrow$`similaires`(s_i)$-s_j$
- `sync-list`(s_i) $\leftarrow$ `similaires`(s_i)
- $p_i^k \leftarrow 1$ **si** $x_i > \max(x_j : s_j \in$ `similaires`$(s_i))$
- SYNCHRONISE(s_i,`sync-list`(s_i),`S++`)
- $q_i^k \leftarrow 1$ **si** $(\forall s_j \in$ `similaires`(s_i) $: p_j^k = 0) \wedge (p_i^k = 0)$
- $k++$
  / Pour les itérations suivantes, pour survivre, le sommet doit être maximal parmi les sommets encore candidats (q=1) */*
- **tant que**$(q_i^{k-1} = 1)$
  - SYNCHRONISE(s_i,`sync-list`(s_i),`S++`)
  - `sync-list`(s_i) $\leftarrow$ `sync-list`(s_i) $-\{s_j : q_j^{k-1} \neq 1\}$
  - $p_i^k \leftarrow 1$ **si** $x_i > \max(x_j : (s_j \in$ `similaires`(s_i) $) \wedge (q_j^{k-1} = 1))$
  - SYNCHRONISE(s_i,`sync-list`(s_i),`S++`)
  - $q_i^k \leftarrow 1$ **si** $(\forall s_j \in$ `similaires`(s_i) $: p_j^k = 0) \wedge (p_i^k = 0)$
  - $k++$

  fin tant que
- **si** $(p_i^k = 1)$ s_i est survivant **fin si**

Algo. 8.3 : *Procédure distribuée et itérative de décimation appliquée au graphe de similarité (cf. §3.5.2 p. 39 pour le rôle de p et q).*

8.2.5 Création du niveau $k+1$ et rattachement des sommets du niveau k

Dans la pratique, chaque sommet survivant ne va pas directement être présent au niveau $k+1$. Au lieu de cela, il va créer (algo. 8.4 p. 158) un nouveau sommet distinct (son père) qui pourra accueillir l'ensemble des structures nécessaires à la représentation de la région issue des rattachements des sommets du niveau k.

CRÉER PÈRE(s_i)
créer $s_{père}$: structure de donnée de type `sommet`, présente au niveau $k+1$
`fils`($s_{père}$)$\leftarrow s_i$
`père`(s_i) $\leftarrow s_{père}$
retourne $s_{père}$

ALGO. 8.4 : *Création des structures de données associées aux sommets survivants dans* G^{k+1}.

Les sommets du niveau k vont devoir se rattacher à un sommet du niveau $k+1$. Les survivants se rattachent naturellement à leur père ; quant aux non-survivants, ils doivent se rattacher au père d'un sommet survivant dans leur voisinage du graphe de similarité. Or, selon la règle 2'(§8.2.4 p. 155), il existe au moins un tel sommet dans leur voisinage (du graphe de similarité).

RATTACHEMENT(s_i)
si (s_i est non survivant)
– $s_j \leftarrow \arg\min h(s_i, s_j)$ tel que (s_j est survivant) $\wedge$ ($s_j \in$ `similaires`(s_i))
– `père`(s_i)$\leftarrow$ `père`(s_j)
– LOCK MUTEX(`fils`(`père`(s_i)))*//verrouille la structure de données*
– `fils`(`père`(s_i)) $\leftarrow$ `fils`(`père`(s_i)) + s_i
– UNLOCK MUTEX(`fils`(`père`(s_i)))
fin si

ALGO. 8.5 : *Rattachement d'un sommet non-survivant du niveau k à un sommet survivant du niveau $k+1$ et construction des liens père-fils.*

Cette procédure va être exécutée en parallèle sur chaque sommet. La structure `fils`(.) des sommets survivants risque donc d'être accédée de manière concurrente : il est nécessaire de la protéger. Nous proposons l'emploi d'un "mutex" [Tan94, p. 576] pour "mutual exclusion" bien adapté à un accès concurrent sur machine à mémoire partagée exécutant des tâches, travaillant dans le même espace d'adressage (processus légers ou thread)(§7.3.4.1 p. 129). Un autre environnement d'exécution nécessitera une technique de protection adaptée. Le mutex peut être assimilé à un sémaphore binaire sur lequel deux opérations atomiques LOCK MUTEX(data) et UNLOCK MUTEX(data) peuvent être exécutées, permettant de verrouiller et déverrouiller la structure de données en argument.

8.2.6 Synthèse

Nous disposons d'un ensemble de traitements locaux ne portant que sur le sommet et son voisinage dans les graphes d'adjacence et de similarité. Ces traitements sont assemblés au sein de la procédure TRAITEMENT PARALLÈLE SOMMET (algo. 8.6 p. 159) qui modélise le traitement distribué effectué sur chaque sommet.

Traitement parallèle sommet($s_i, images, S^{k+1}$)
Construit adjacence(s_i)
Mise à jour des attributs région(s_i,`fils`(s_i),images)
synchronise(s_i,`adjacents`, attributs_regions_ok)
Construire similarité(s_i)
Décimation distribuée (s_i)
si (s_i est survivant)
– $s_{père}$← Créer père(s_i)
– synchronise(s_i,`adjacents`,décimation_ok)
– lock mutex(S^{k+1})
– $S^{k+1} \leftarrow S^{k+1}+s_{père}$
– unlock mutex(S^{k+1})
sinon /* *s_i n'est pas survivant*/*
– synchronise(s_i,`adjacents`,décimation_ok)
– Rattachement(s_i)
fin si

Algo. 8.6 : *Procédure de traitement locale d'un sommet.*

Finalement, la procédure Construction pyramide (algo. 8.7 p. 159) contrôle et séquence la construction de la pyramide ; construction qui est effectuée niveau par niveau tant que la contraction d'un niveau à l'autre est significative : un seuil fixé par l'utilisateur valant "contraction minimale". Au niveau k, card(S^k) threads sont créés et lancés de façon concurrente avec la procédure Traitement parallèle sommet. On attend ensuite leur fin avec la procédure Join avant de passer au niveau supérieur.

Construction pyramide(*images*)
partition ← Split(images)
$k \leftarrow 0$
$G^k \leftarrow$ Construire graphe d'adjacence(partition)
tant que($(k = 0)$ **ou** (card(S^{k-1})-card(S^k)≥contraction minimale))
– **pour tout**($s_i \in S^k$)
  – thread←créer une nouvelle tâche (thread)
  – thread.Run(Traitement parallèle sommet(s_i, S^{k+1}))
  fin pour tout
– Join()/* *attend la fin de l'exécution des tâches lancées* */
– $k++$
fin tant que
retourne S^{k-1}

Algo. 8.7 : *Construction de la pyramide de graphe, avec un traitement concurrent des sommets à un niveau donné.*

On remarque, comme le montre la figure 8.2 page 160, que le traitement est concurrent et distribué entre sommets pour un niveau donné, et séquentiel et centralisé de niveau en niveau.

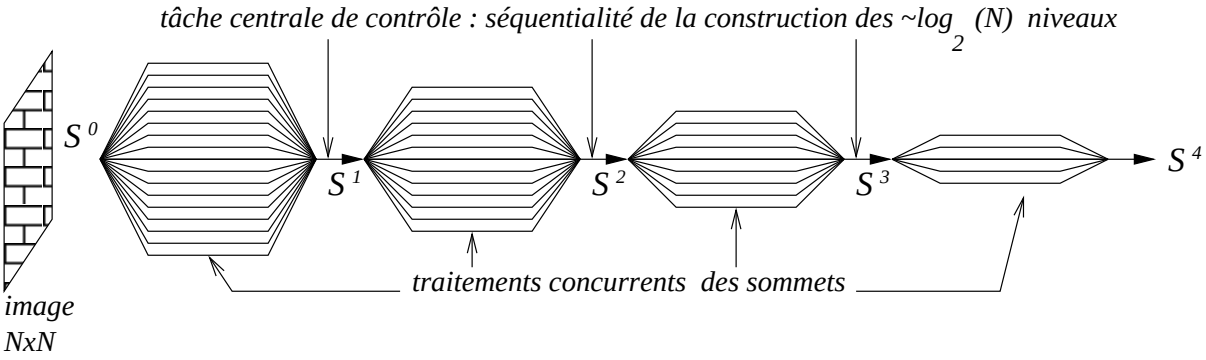


Fig. 8.2 : *Schéma d'exécution du traitement dans la pyramide.*

8.3 La pyramide irrégulière : une organisation de contrôle social

Les systèmes multi-agents proposent d'éclater la complexité des traitements au sein d'entités coopérantes permettant dans le cas de la segmentation d'image, de bénéficier des avantages évoqués en introduction de ce chapitre. Une conséquence de cette démarche est de rendre essentielle la notion d'organisation dans un système multi-agents afin de garantir la cohérence globale du système. Or, les différentes approches de segmentation d'image à l'aide d'agents situés dans l'image (§5.5.3 p. 100) n'identifient pas clairement une telle organisation.

L'organisation définit les rôles respectifs des agents et les relations abstraites les liant. Ainsi, il devient possible d'appréhender le système dans sa **globalité** durant les deux étapes suivantes :

1. l'étape de conception où l'on définit les rôles, les relations et les **interactions** entre agents ;
2. l'étape de validation où l'on vérifie que la somme des interactions locales réalise bien les objectifs globaux du système. Concrètement, lorsque l'on constate un dysfonctionnement global, l'organisation fournit un schéma de lecture du système rendant possible la localisation ce dysfonctionnement.

La pyramide organise les agents : à la manière d'une société qui se dote d'une constitution et d'un ensemble de lois, nous avons choisi la pyramide irrégulière comme organisation imposant sa structure aux agents situés dans l'image. Ce choix n'exclut pas d'autres approches, en particulier la pyramide duale ; cependant, la pyramide irrégulière possèdent certaines propriétés la rendant particulièrement adaptée à notre problème, à savoir :

1. La capacité de mettre en œuvre des traitements en parallèle, chaque sommet n'interagissant qu'avec ses voisins. On peut ainsi profiter de l'aspect intrinsèquement distribué des agents.
2. Le graphe d'adjacence fournit des informations sur la topologie des régions, ce qui en fait un bon support pour des agents situés dans l'image.
3. L'approche "graphe" fournit un premier niveau d'abstraction vis-à-vis de l'image : on retrouve des notions d'adjacence, ou de similarité, voisines des notions d'accointance, ou d'affinité, courantes dans les systèmes multi-agents.
4. Chaque graphe de la pyramide se présente comme un résultat intermédiaire, on peut donc stopper à tout moment le processus, en disposant d'un résultat incomplet mais utilisable. De plus, cette pile de graphes fournit un historique de l'évolution du système autorisant une évaluation *a posteriori* du comportement des agents.
5. Outre l'aspect multirésolution que procure la pyramide, elle permet une construction **progressive** des primitives de l'image donnant pour chaque niveau, à l'aide du graphe d'adjacence, les relations topologiques entre primitives. Ces aspects sont utiles pour la vision intermédiaire et haut-niveau (chap. 4 p. 51) où l'on a recours à des algorithmes de propagation d'informations, de contraintes ou d'appariement de graphes. Ces procédures permettent, dans le cadre d'une stratégie ascendante, de franchir progressivement les différents niveaux de description à l'instar de la construction des niveaux de la pyramide. En conséquence, comme nous le verrons au chapitre 11, nous proposons l'utilisation de la pyramide d'agents comme un cadre favorisant une mise en œuvre homogène et progressive de toutes les étapes de la vision, de la segmentation à l'interprétation.

Les agents enrichissent l'approche pyramidale : l'utilisation d'agents au lieu de simples sommets enrichit le cadre décisionnel de l'approche pyramidale. En effet, l'approche pyramidale n'est pas sans inconvénient (chap. 3 p. 33), une manière d'y remédier serait de pouvoir mettre en œuvre des stratégies locales, coopératives, et d'intégrer au mieux l'information *a priori* . Les agents fournissent une abstraction particulièrement adaptée à l'enrichissement du cadre décisionnel tant pour la mise en œuvre de stratégies locales et coopératives que pour l'intégration d'informations *a priori* .

La pyramide irrégulière d'agents se présente comme une pyramide irrégulière dans laquelle chaque sommet va être remplacé par un agent (fig. 6.4, p. 113).

La délocalisation et un enrichissement du cadre décisionnel favorisent la mise en œuvre de stratégies coopératives, localement adaptées, et de prise en compte de l'*a priori* .

Les algorithmes de graphes définis à la section précédente vont s'appliquer à chaque niveau de la pyramide. Pour cela, ils sont traduits en protocoles d'interactions locales au sein des agents. Ainsi, la pyramide irrégulière impose sa structure aux agents agissant comme une "Loi" qu'ils doivent respecter afin d'assurer un contrôle social et la convergence du système.

8.3.1 Agents région & agents contour

Chaque agent va représenter une primitive de l'image. Afin de mettre en œuvre la coopération région/contour évoquée au chapitre 6, certains agents représenteront une primitive région tandis que d'autres représenteront une primitive contour.

Les agents sont des entités autonomes mues par un thread propre à chaque agent, disposant chacun de sept comportements et de croyances propres sur lesquels nous reviendrons plus en détails au chapitre 9.

Les agents région disposent d'un ensemble de données locales (leurs croyances) semblables à la structure de la figure 8.1 page 154. Ces données décrivent la région que l'agent représente. Ils disposent aussi de sept comportements correspondant, entre autres, à la mise en œuvre de l'algorithmique distribuée décrite à la section 8.2 p. 153. Les agents ont comme but de fusionner entre agents similaires afin d'effectuer une décimation sur la population d'agents du niveau k pour obtenir une nouvelle population d'agents générée par les agents survivants au niveau $k+1$. On notera P^k_{reg}, la population d'agents région du niveau k.

Les agents contour disposent d'un ensemble de données locales (leurs croyances) décrivant une chaîne de segments de contour. L'objectif à moyen terme est de pouvoir fusionner les agents contour et donc d'appliquer une décimation semblable à celle effectuée sur les agents région. Les fusions sont choisies par des règles s'inspirant du système de Nazif et Levine (§4.6.1.4 p. 74) & [Naz84]. Les développements menés dans ce sens n'ont pas acquis la maturité nécessaire pour être opérationnels ; en conséquence, les agents contour dans la version actuelle du système se répètent à l'identique niveau après niveau, intervenant comme support de fusion des agents région, via un comportement coopératif.

Nous envisageons une autre possibilité dans laquelle des contours actifs remplacent les chaînes de segments de contour.

On notera P^k_{edge}, la population d'agents contour du niveau k.

8.3.2 Transposition d'un espace opératoire à un espace comportemental

Une société d'agents à chaque niveau de la pyramide. Chaque niveau k de la pyramide est composé d'une population d'agents P^k qui est l'union de la population des agents région et contour pour ce niveau :

$$P^k = P^k_{reg} \bigcup P^k_{edge}$$

Les relations d'adjacence à chaque niveau de la pyramide. Lorsque deux agents sont adjacents dans l'image, une relation d'accointance (§7.4.3.1 p. 147) de type "adjacent" va être établie entre eux. Cette relation prend la forme d'une instanciation d'une classe `Acquaintance` (fig. 7.19, p. 148) représentée dans le formalisme déclaratif :

`(Acquaintance (id 43) (types adjacent))`

L'union des relations d'accointance étiquetées "adjacent" forme donc les relations d'adjacence dans l'image. Si l'on note $\texttt{adjacents}(a_i)$, l'ensemble des accointances de l'agent a_i étiquetées "adjacent" ; alors, A^k_{adj} l'ensemble des relations d'accointance de type adjacent du niveau k est formé par :

$$A^k_{adj} = \bigcup_{a_i \in P^k} \texttt{adjacents}(a_i)$$

Chaque niveau de la pyramide d'agent est une organisation. Chaque niveau k de la pyramide est une organisation d'agents adjacents O^k_{adj}, structurée par le réseau d'accointances d'adjacence A^k_{adj} et composée de la population des agents du niveau courant P^k :

$$O^k_{adj}(P^k, A^k_{adj})$$

Cette organisation représente l'information région et contour, et sa topologie dans l'image à une certaine résolution.

Cette formalisation effectuée, on peut la mettre en relation avec le graphe d'adjacence évoqué au chapitre 3. Pour un niveau k donné, l'ensemble des sommets S^k est transposé dans l'espace de la population d'agents P^k :

$$S^k \Leftrightarrow P^k$$

L'ensemble des arêtes joignant les sommets A^k est transposé dans l'espace des relations d'accointance de type adjacent A^k_{adj} :

$$A^k \Leftrightarrow A^k_{adj}$$

Finalement, le graphe d'adjacence $G^k(S^k, A^k)$ est transposé en une organisation d'agents $O^k_{adj}(P^k, A^k_{adj})$:

$$G^k(S^k, A^k) \Leftrightarrow O^k_{adj}(P^k, A^k_{adj})$$

Cette équivalence définit une transposition d'un espace opératoire travaillant sur des structures de graphes où les sommets représentent des primitives (régions ou contours) et les arcs des relations d'adjacence dans l'image, vers un espace comportemental où des agents actifs et autonomes remplacent les sommets du graphe. La notion d'accointance entre agents représente une adjacence dans l'environnement représenté par l'image.

Cette transposition permet d'enrichir le cadre décisionnel des approches pyramidales en distribuant l'intelligence dans l'image à travers l'abstraction que représente l'agent. Cette intelligence, à l'origine présente sous la forme d'une connaissance procédurale et algorithmique, est désormais encapsulée dans l'agent, qui est une abstraction prédisposée à la mise en œuvre de stratégies coopératives, localement adaptées et intelligentes, capables d'utiliser un savoir *a priori* ou un modèle décrit dans un langage déclaratif.

8.3.3 Caractéristiques de l'organisation

L'organisation des agents adjacents d'un niveau donné est de type variable-égalitaire-prédéfinie. En effet, le rôle respectif des agents est prédéfini mais les relations entre agents peuvent varier en fonction : (i) des rencontres faites dans l'environnement (représentant l'image) ; (ii) des arrangements négociés lors des comportements de décimation (§9.9 p. 202) et de rattachement (§9.10 p. 206). Comme les agents d'un niveau interviennent d'une manière uniforme sans notion hiérarchique, l'organisation est dite égalitaire.

La coopération entre agents peut prendre deux formes :

1. Les agents travaillent sur des portions quasi[7] disjointes de l'image. Ainsi, une forme de coopération augmentative résulte de la somme des travaux "quasi" indépendants des agents. Cette coopération n'existe que du point de vue de l'observateur, qui constate *a posteriori* un travail effectué par des entités.
2. Cependant, les agents disposent d'un comportement de coopération intentionnel (§9.8 p. 192) résultant d'interactions locales entre agents adjacents. Cette coopération peut être qualifiée d'intégrative car certains agents intègrent les points de vue de leurs voisins dans leur processus de décision.

Type et nombre d'agents. Les agents sont de granularité moyenne, ils disposent d'une mémoire et de mécanismes simples de raisonnement, leur poids mémoire avoisine les 50 ko. Leur nombre est important : quelques milliers pour le premier niveau de la pyramide.

Des agents situés dans l'image. Chaque agent est localisé dans l'image par rapport à la primitive qu'il représente. L'organisation structurée par les accointances d'adjacence, fait de chaque niveau de la pyramide une carte topologique de l'information présente dans l'image à un certain niveau de résolution. Notre architecture répond donc à tous les critères de la famille des agents situés dans l'image que nous avons évoqués à la section 5.5.3 p. 100.

8.4 Les agents interface et environnement

Afin de réaliser l'interface avec l'utilisateur, le système est doté d'un agent particulier : `IHMAgt`. Ce dernier récupère un ensemble de paramètres fournis par l'utilisateur, qu'il communique (fig. 8.3, p. 165) à un autre agent particulier `MonitorAgt` (§8.5 p. 166) qui prend en charge l'initialisation du système et surveille la construction de la pyramide. Un dernier agent `EnvAgt` va représenter l'environnement partagé par les agents de la pyramide.

[7] Les agents région travaillent sur des régions disjointes, mais les agents contour travaillent sur des segments de contour chevauchant des régions.

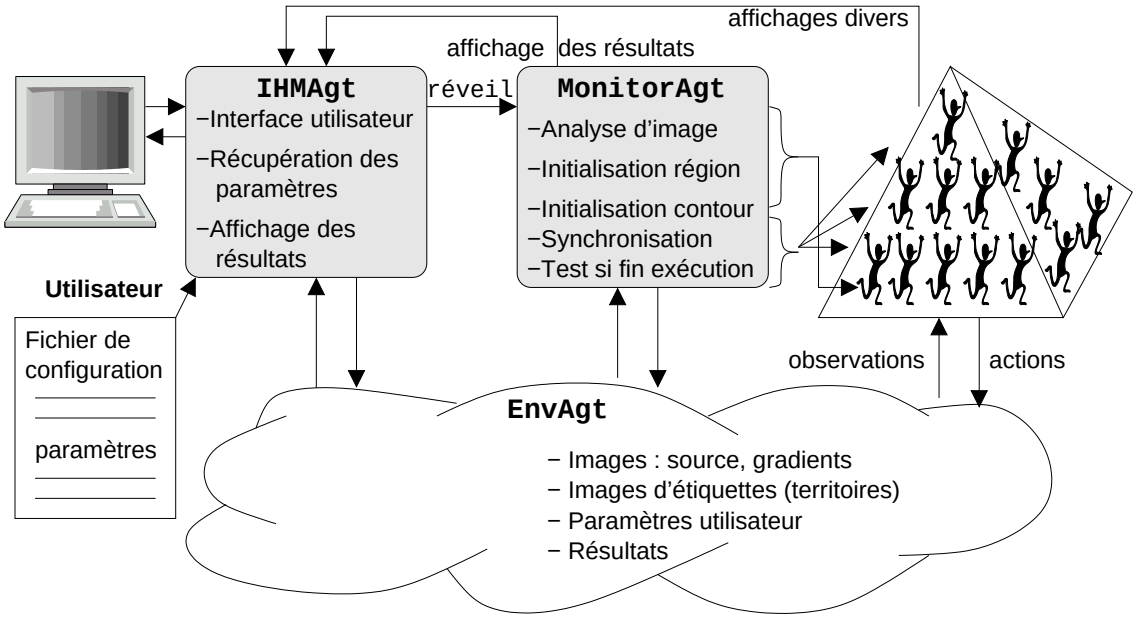


Fig. 8.3 : *La communication au sein du système : l'agent* `IHMAgt` *responsable de l'interface utilisateur ; l'agent* `MonitorAgt` *initialise la pyramide ; et l'agent* `EnvAgt` *sert d'environnement aux agents de la pyramide.*

8.4.1 L'agent environnement (`EnvAgt`)

Cet agent va disposer de toutes les informations partagées par les agents du système c'est-à-dire :

- les paramètres fournis par l'utilisateur ;
- l'image source ou image à segmenter ;
- des images de normes de gradient et d'orientations de gradient ;
- des images d'étiquettes pour chaque niveau de la pyramide, correspondant chacune à la segmentation obtenue pour le niveau donné. Une étiquette correspond à l'identifiant de l'agent responsable de la primitive.
  Ainsi, pour chaque niveau k, il y a une image de labels des agents région `reg_map(`k`)` et une image de labels des agents contour `edge_map(`k`)`. Ce sont ces images que les agents de la pyramide explorent (§9.5 p. 181) pour rencontrer de nouveaux agents qui leur sont adjacents.

8.4.2 L'agent interface utilisateur (`IHMAgt`)

Cet agent effectue le lien avec l'utilisateur du système, il propose sous la forme d'un ensemble de menus déroulant le choix de l'image à traiter, et la modification des paramètres du système. Il commence par lire un fichier contenant les paramètres par défaut du système que l'utilisateur peut modifier à l'aide de l'interface graphique.

Les paramètres étant déterminés, l'utilisateur lance l'exécution de la pyramide. L'agent `IHMAgt` récupère l'ordre et les paramètres qui sont transmis à l'agent `MonitorAgt`. Tout agent peut demander à `IHMAgt` d'afficher des résultats, ou des fenêtres, en lui

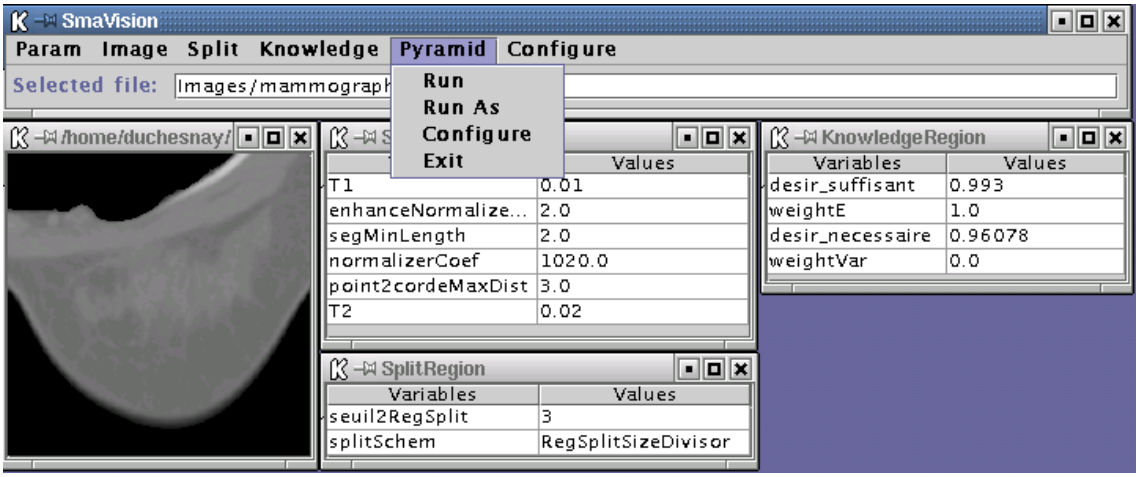


Fig. 8.4 : *Le menu principal de l'interface graphique et quelques panneaux de configuration.*

envoyant simplement un message d'affichage. On obtient ainsi une architecture modulaire, souple et distribuable, l'agent `IHMAgt` pouvant s'exécuter sur une machine différente[8] de celle où s'exécute les agents de la pyramide.

Nous traiterons plus en détails les paramètres du système ainsi que l'affichage des résultats au chapitre 10.

8.5 Initialisation : de l'image à l'organisation d'agents

L'étape d'initialisation permet de construire, à partir de l'image, le premier niveau de la pyramide. Cette initialisation est menée par un agent particulier : `MonitorAgt`.

Cet agent va aussi jouer le rôle **d'état civil** des agents présents dans la pyramide en constatant les naissances (création d'agents dans le prochain niveau de la pyramide) et les "décès" (fin d'exécution d'agents dans le niveau courant).

L'idée originale était à l'instar des pyramides irrégulières d'associer un agent à chaque pixel. Or, pour une image de 1000x1000 pixels, chaque agent pesant 50ko de mémoire, cela nécessiterait 5Go de mémoire pour traiter le premier niveau.

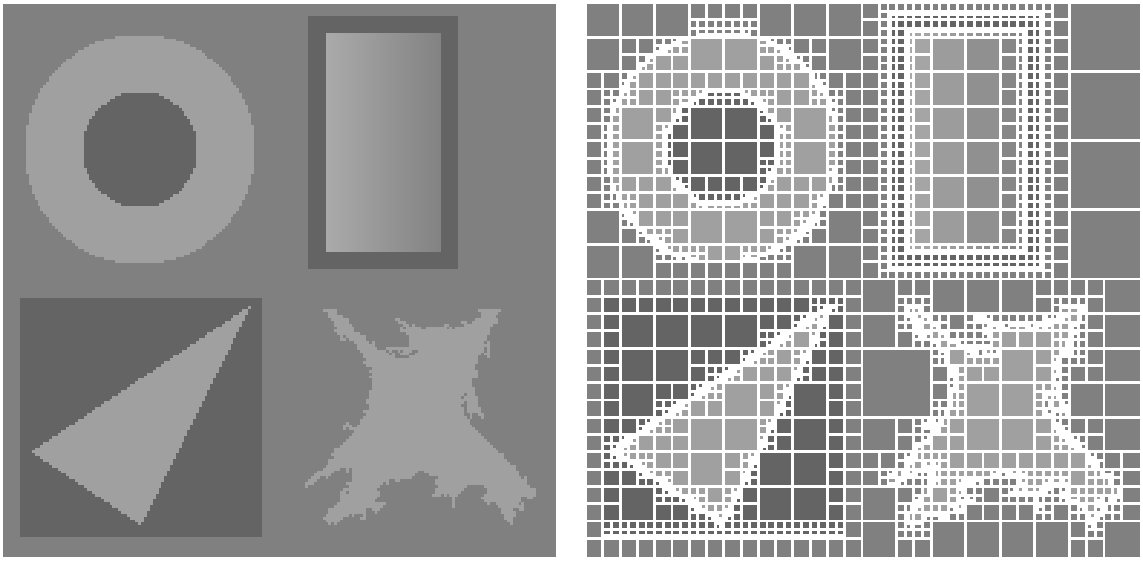
De plus, les images sont couramment constituées de grandes zones homogènes faciles à segmenter avec des approches classiques. Ainsi, des spécialistes dédiés à l'analyse d'image incorporés à l'agent `MonitorAgt` vont effectuer une première analyse de l'image dont le résultat est une sur-segmentation grossière qui réduit considérablement le nombre de primitives initiales et donc le nombre d'agents. De plus, il semble pertinent d'utiliser une méthode simple et rapide pour les zones de l'image qui s'y prêtent.

[8]Pour l'affichage, `IHMAgt` agit à la manière d'un serveur X-Window, communicant avec ses clients sur TCP/IP.

8.5.1 Initialisation des agents région

La figure 8.8 page 173, propose une modélisation sous forme de RdP des états de l'agent `MonitorAgt`. La réception d'un message `réveil` provenant de l'agent interface graphique provoque le déclenchement du spécialiste `InitRégions`.

Ce spécialiste d'analyse d'image de l'agent produit une première sur-segmentation de l'image. Il emploie pour cela un algorithme de division récursive (§2.4.1 p. 27) basée sur une structure d'arbre quaternaire. Un agent région est créé pour chaque région issue de cette sur-segmentation. Cette pré-segmentation permet de limiter le nombre d'agents région et la charge du système, tout en concentrant un grand nombre d'agents et donc la capacité calculatoire du système, sur les zones de transition ou texturées.



(a) (b)

Fig. 8.5 : *(a) Image tirée de savoise (GDR-PRC-ISIS) ; (b) sur-segmentation obtenue par un algorithme de division récursive.*

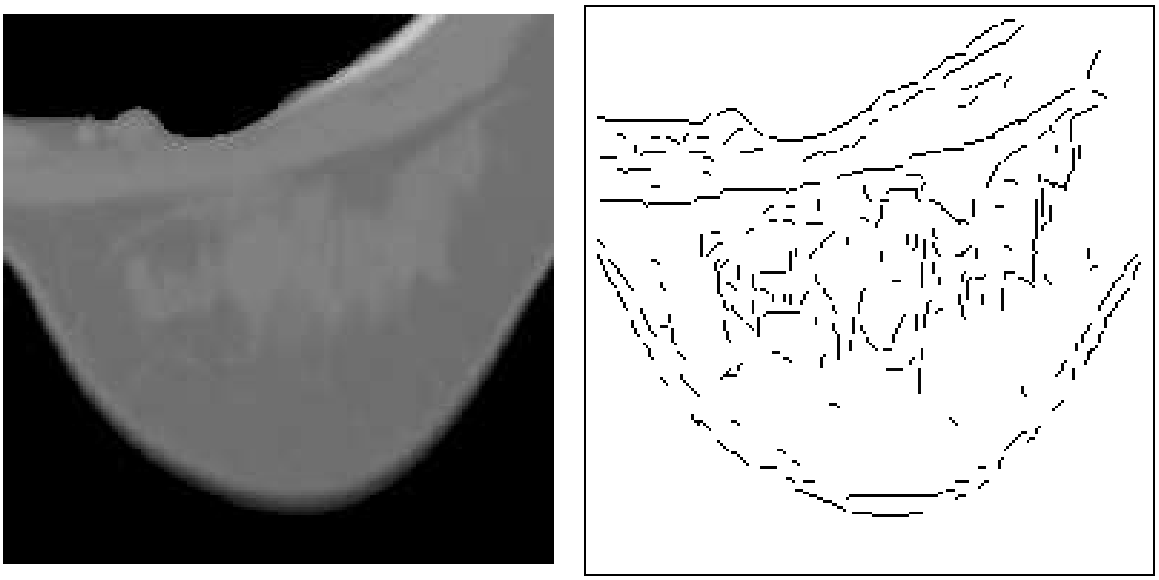
Ainsi, comme le montre la figure 8.5, la pré-segmentation obtenue sur une image 192x192 (issue de savoise du GDR-PRC-ISI) génère 2083 régions et autant d'agents région dans le premier niveau de la pyramide. Chaque agent région (chap. 9 p. 175) est initialisé avec des structures de données (placées dans les croyances de l'agent (§9.3 p. 178)) décrivant la primitive région dont l'agent est le représentant.

Pour le codage de la région de chaque agent région, on utilise une structure similaire à celle présentée (fig. 8.1, p. 154). Pour les agents du premier niveau, les champs récepteurs de cette structure sont directement issus de la région obtenue par l'algorithme de division récursive de l'image.

8.5.2 Initialisation des agents contour

Un second spécialiste `InitContours` va appliquer une séquence d'opérateurs permettant d'extraire des chaînes de segments de contour robustes mais fragmentées. Cette séquence d'opérateurs a comme objectif les critères de Canny (§2.3.1 p. 22). La séquence d'opérateurs est la suivante :

1. Un détecteur de contours dérivatif du premier ordre (Sobel) (§2.3.1 p. 22). Ce détecteur génère deux images : une image de la norme des gradients et une image de leurs orientations.
2. Un opérateur de rehaussement de contours (pour la bonne détection) est appliqué afin d'améliorer la qualité des transitions qui parfois sont en pente douce.
3. Un opérateur de suppression des non-maximas locaux (§2.3.2 p. 23) permet d'obtenir une bonne localisation et une faible multiplicité des réponses.
4. Un opérateur de suivi de contours (§2.3.3 p. 24) extrait des chaînes de points où le gradient est important. Nous avons adopté un suivi de ligne de crête du même type que celui évoqué (§2.3.3 p. 24).
5. Finalement, un opérateur de type approximation polygonale [Pav77] transforme des chaînes de points de contour en chaînes de segments de contour.



(a) (b)

Fig. 8.6 : *(a) Une image de scanner du sein ; (b) segments de contour extraits.*

Cette séquence d'opérateurs appliquée sur l'image d'un scanner de sein (fig. 8.6, p. 168) génère 255 chaînes de segments de contour et donc autant d'agents contour. Les agents contour (chap. 9 p. 175) sont initialisés avec des structures de données (placées dans les croyances de l'agent (§9.3 p. 178)) décrivant la primitive contour qu'ils représentent.

8.5.3 Construction des relations d'accointance

La construction des relations d'accointance est mise en œuvre par les agents eux-mêmes grâce à deux comportements `MarquageTerritoire` et `Exploration` décrits respectivement (§9.4 p. 179) et (§9.5 p. 181). Le premier comportement marque, dans l'environnement, le territoire (région, chaîne de segments de contour) représenté par l'agent. Le deuxième comportement permet aux agents de se rencontrer dans cet environnement (constitué de cartes d'identifiants).

Une rencontre se traduit par l'établissement d'une relation d'accointance de type "adjacent", on y ajoute aussi la nature région ou contour de l'agent rencontré.

Les rencontres ainsi faites deviennent les voisins de l'agent, avec qui il va interagir et peut-être fusionner.

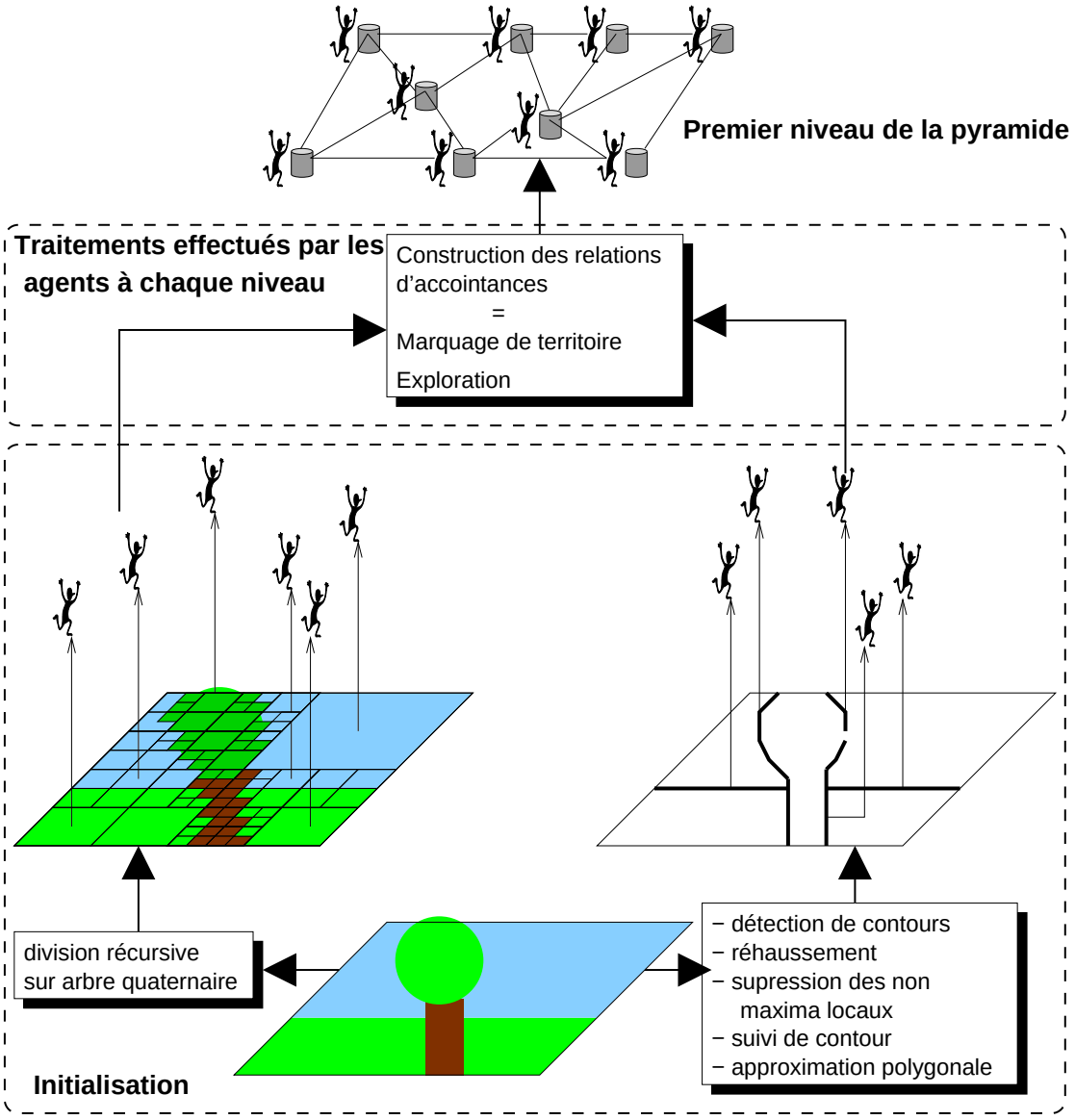


Fig. 8.7 : *Récapitulatif des étapes de l'initialisation, suivi de l'étape de reconstruction des relations d'accointance, cette dernière étant effectuée par tous les agents (de manière concurrente) et ce, pour chaque niveau.*

8.6 Construction de la pyramide

Les liens entre agents étant tissés, chaque agent va interagir localement avec ses voisins afin de déterminer ceux avec lesquels il souhaite fusionner. La construction concurrente de ces plans de fusion, leur réalisation et la gestion des conflits résultants sont mis en œuvre au sein de chaque agent par sept comportements présentés au chapitre 9 et résumés au chapitre 6 (fig. 6.5, p. 115).

8.6.1 Synchronisation et passage d'un niveau à l'autre

L'agent `MonitorAgt` va intervenir en trois points pour synchroniser l'activité des agents de la pyramide :

1. Comme nous le verrons (§9.10 p. 206), le territoire d'un agent du niveau $k+1$ dépend des agents du niveau k se rattachant à lui. Il est donc nécessaire d'attendre la fin de l'exécution de tous les agents du niveau k avant qu'un agent du niveau $k+1$ ne débute le marquage de son territoire.
2. Un agent doit, avant d'explorer son environnement être certain que tous les agents du même niveau ont bien fini de marquer leurs territoires respectifs.
3. L'exploration du territoire par un agent débouche sur la découverte de nouveaux voisins auprès desquels il doit se faire connaître. Il est donc nécessaire, avant de passer aux étapes de traitements ultérieures, de s'assurer que les démarches de prises de contact entre agents ont bien toutes abouti.

Nous allons voir, dans les paragraphes suivants, comment l'agent `MonitorAgt` est utilisé pour ces trois synchronisations globales. Ces dernières sont présentées sous la forme d'un RdP (fig. 8.8, p. 173). Le contrôle de l'agent ainsi modélisé répond au modèle proposé (§7.4.1.6 p. 141).

L'agent `MonitorAgt` répertorie les agents de chaque niveau : afin de pouvoir contrôler la croissance de la pyramide, l'agent `MonitorAgt` joue le rôle d'état civil répertoriant les créations de nouveaux agents (sur le prochain niveau de la pyramide) et les "décès" (fin d'exécution) des agents du niveau courant.

Deux situations peuvent générer de nouveaux agents :

1. Création des agents du premier niveau au moment de l'initialisation (voir `CréerAgents` 8.8 page 173).
2. Reproduction d'un agent survivant ((§9.11 p. 209)) lors des niveaux suivants. Dans ce cas, l'agent survivant signale au moniteur l'existence du nouvel agent créé par un message `nouvel-agent`.

   Le spécialiste `RepertorierNouvelAgent` (fig. 8.8) traite ce message et rajoute un jeton dans la place `agents-du-niveau-courant`.

Les jetons présents dans cette place vont transiter dans les places `agents-du-niveau-courant` et `agents-en-attente` au fur et à mesure des trois synchronisations.

Les trois synchronisations globales : l'agent `MonitorAgt` est dans l'état `S1` (fig. 8.8) et envoie un message de réveil (`réveil`) à tous les agents du niveau courant. Ce message permet aux agents d'engager leur comportement de marquage du territoire.

Puis, lorsque la place `agents-du-niveau-courant` est vide, il peut passer dans l'état `W1` d'attente de la fin du comportement de marquage de tous les agents du niveau courant de la pyramide.

Ces derniers signifient au moniteur la fin de leur comportement de marquage de territoire par un message `fin-marquage`. Lorsque le moniteur a reçu autant de messages `fin-marquage` qu'il y avait de jetons en `agents-en-attente`, il passe dans l'état `S2` pour débuter la seconde synchronisation.

Les deux synchronisations suivantes s'effectuent suivant le même principe que la première.

Passage au niveau suivant : la dernière synchronisation se termine lorsque l'agent moniteur a reçu autant de messages `fin-exécution` qu'il y a de jetons dans la place `agents-en-attente`. Autrement dit, tous les agents du niveau courant ont fini de fusionner les uns avec les autres et les survivants ont fini de se reproduire.

Alors, l'agent moniteur va évaluer à l'aide du spécialiste `NiveauSuivant` (fig. 8.8) s'il doit passer au niveau suivant ou s'il doit arrêter la croissance de la pyramide. Si le critère d'arrêt n'est pas rempli (§8.6.2), l'agent incrémente la place des compteurs de niveau et dépose le jeton d'état dans la place `S1` pour recommencer un nouveau niveau de la pyramide. Si les conditions sont vérifiées, il passe dans l'état `End` en demandant à l'agent responsable de l'interface graphique l'affichage des résultats.

8.6.2 Conditions sur la fin d'exécution

Outre les trois points de synchronisation globale, l'agent `MonitorAgt` surveille les conditions d'arrêt de croissance de la pyramide.

La convergence du système est assurée car les agents respectent les "lois" des pyramides irrégulières. La décimation des agents de chaque niveau est équivalente à une décimation sur le graphe de similarité (§3.6.1 p. 42).

Ce dernier est composé d'un ensemble de composantes connexes (représentant chacune une région homogène de l'image)(§3.6.1 p. 42) groupant les sommets similaires deux à deux. Or, si il n'y a plus de couple de sommets jugés similaires, chaque composante connexe se réduit à un singleton : il n'y a donc plus de décimation possible et la procédure de construction de la pyramide s'arrête (§8.2.6 p. 158).

Comme nous le verrons dans le chapitre 10, une forte proportion des décimations se produit (cela va dépendre de l'image) dès les premiers niveaux de la pyramide. Lors des derniers niveaux, la diminution du nombre de sommets[9] peut être très faible et n'apporter que très peu d'améliorations à la qualité de la segmentation. Ainsi, il peut être pertinent d'arrêter la croissance de la pyramide avant son terme.

[9]Nous rappelons que la décimation s'effectue dans le graphe de similarité et non dans le graphe d'adjacence.

Pour cela, nous utilisons un critère d'arrêt qui évalue si la décimation d'un niveau à l'autre est toujours significative. Il suffit d'effectuer un test comparant le nombre de sommets au niveau courant et au niveau précédent. Bertolino [Ber95] propose d'évaluer le rapport entre le nombre de sommets aux deux niveaux.

Nous suggérons un critère équivalent à celui proposé par [Ber95] : la différence entre le nombre d'agents de deux niveaux consécutifs doit être supérieure à un seuil fixé par l'utilisateur.

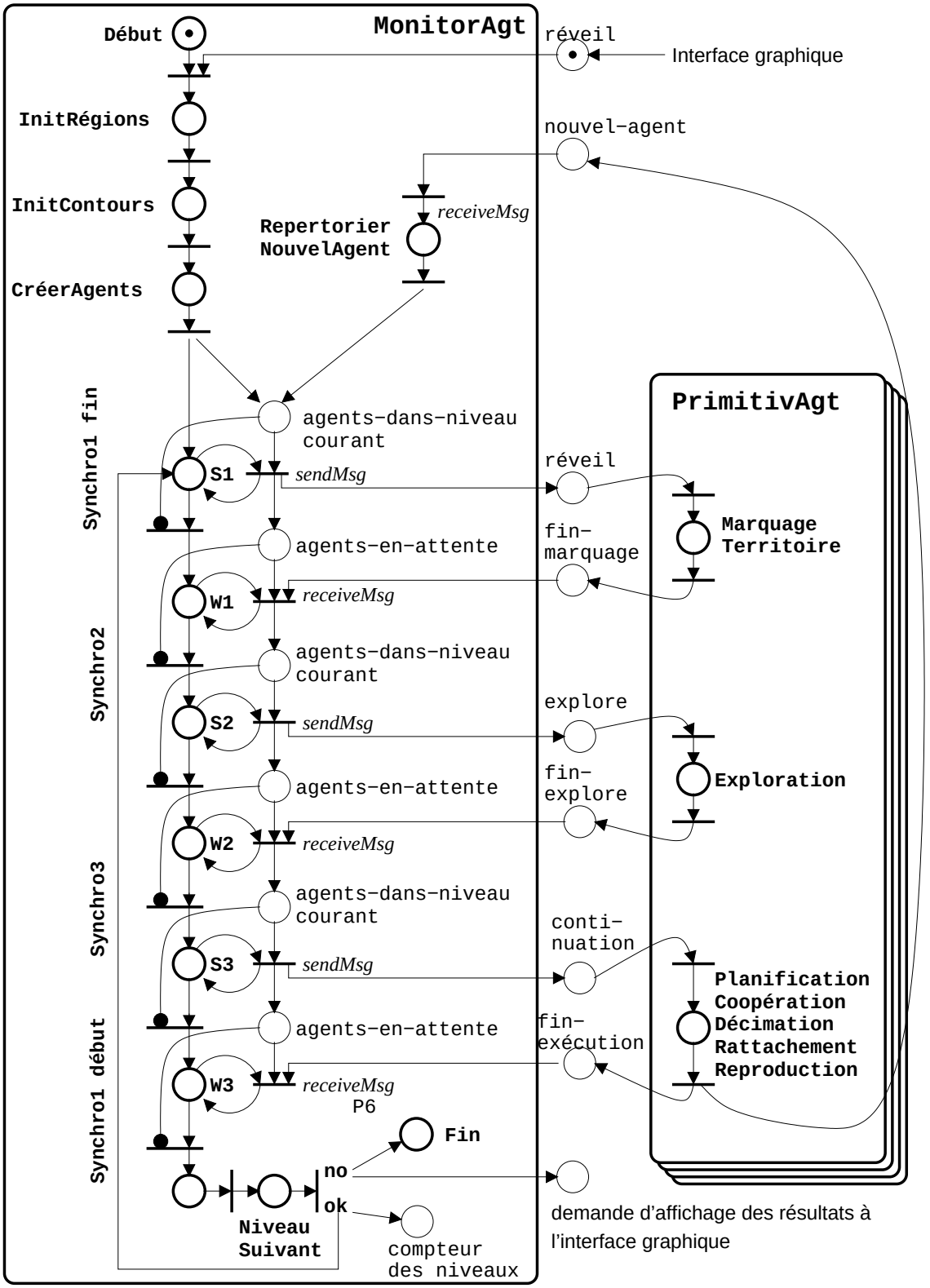


FIG. 8.8 : *RdP modélisant l'agent* `MonitorAgt` *: l'étape d'initialisation et les trois synchronisations globales. Légende :* `S?` *début d'une Synchronisation ;* `W?` *: (Wait) état d'attente.*

Chapitre

9

L'agent et ses comportements

Nous abordons dans ce chapitre la deuxième étape de la description de l'architecture de contrôle.

1. Description globale et structurelle de l'organisation regroupant les agents (chap. 8).
2. Description locale et fonctionnelle des agents composant le système.
3. Analyse globale et fonctionnelle où l'on vérifie que l'ensemble des interactions locales réalise bien ce pour quoi le système est conçu (chap. 10).

Au cours de ce chapitre, nous allons donner une description individuelle de l'agent qui portera sur ses sept comportements. Nous préciserons aussi les mécanismes de contrôle et les interactions locales de l'agent avec ses voisins dans l'organisation définie au chapitre 8.

Deux types d'agents, qui traduisent les primitives région et contour, coexistent et interagissent au sein de la pyramide. Nous proposons une mise en œuvre particulière des aspects concernant l'adaptation locale, la coopération (région/région, région/contour) et l'intégration de l'incertitude dans la connaissance *a priori* .

Néanmoins, on peut parfaitement envisager d'autres types de propositions dans la mesure où les règles imposées par la pyramide irrégulière sont respectées. Nous pensons en particulier à une plus importante contribution des agents traduisant les primitives contour.

9.1 Introduction

Après avoir abordé l'aspect global du système c'est-à-dire son organisation, nous allons étudier le fonctionnement de chaque agent et les interactions locales entre agents.

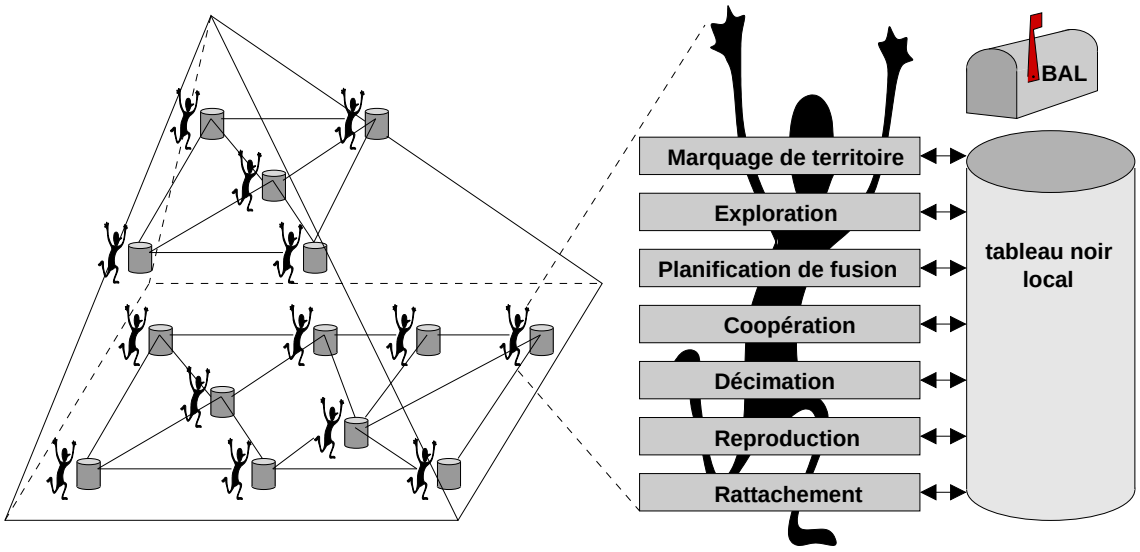


FIG. 9.1 : *Un agent de la pyramide (`PrimitivAgt`) et ses sept comportements.*

Un agent de la pyramide représente une primitive région ou contour de l'image. Un agent va tâcher de compléter cette primitive en fusionnant avec d'autres agents au moyen de **sept comportements** (fig. 6.5, p. 115) :

1. **Le comportement de marquage de territoire** définit le territoire de l'agent dans l'environnement partagé par tous. L'environnement dans ce cas est concrétisé par des cartes de labels.
2. **Le comportement d'exploration** de l'environnement permet à l'agent de détecter ses voisins dans l'image. Un agent va ainsi construire ses relations de voisinage reflétant l'adjacence spatiale dans l'image. Ces relations structurent la population d'agents du niveau courant en une organisation qui est l'équivalent agent du graphe d'adjacence.
3. **Le comportement de planification des fusions.** Chaque agent région doit décider pour chacun de ses voisins région s'il souhaite ou non fusionner avec lui. L'agent doit donc dresser un plan des actions qu'il souhaite entreprendre avec ses voisins. Ce plan est constitué de trois listes :
   (a) Une liste contenant les voisins avec qui il souhaite fusionner (cette liste est le plan de fusion).
   (b) Une liste contenant les voisins avec qui il ne souhaite pas fusionner ;
   (c) Une liste contenant les voisins avec qui il n'est pas certain de vouloir fusionner.

   Cette incertitude reflète celle du traiteur d'image, les agents décideront de la nécessité des fusions lors du comportement suivant.

4. **Le comportement coopératif** permet à chaque agent de lever les ambiguïtés (fusions incertaines) à l'aide d'un processus visant à collecter puis fusionner le plus d'informations possibles à propos des hypothèses incertaines. Cette collecte d'informations s'effectue à l'aide de demandes d'avis, faites aux agents voisins qu'ils soient de type région ou contour.
5. **Le comportement de décimation** vise à sélectionner des survivants parmi les agents du niveau courant de la pyramide. Cette sélection est basée sur l'utilité associée au plan de fusion précédemment défini.
6. **Le comportement de rattachement** autour des agents survivants est engagé entre survivants et non-survivants.
7. **Le comportement de reproduction** est engagé par les agents survivants afin de créer un nouvel agent dans le niveau suivant de la pyramide.

Remarque : comme nous l'avions déjà mentionné au chapitre 8, les agents contour se répètent à l'identique, niveau après niveau. Ils interviennent en complément d'informations contour via un comportement coopératif, pour influencer les fusions entre agents région.

L'objectif dans le futur étant de favoriser une meilleure prise en compte de l'information contour en utilisant par exemple des contours actifs à la place de simples chaînes de segments de contour.

9.2 Structure de contrôle de l'agent

La figure 9.2 page 178, présente les principaux états de raisonnement de l'agent, conditionnant l'activation de ses sept comportements. Chaque comportement étant lui-même modélisé par un sous-réseau de Pétri qui est présenté dans la section qui lui est consacrée.

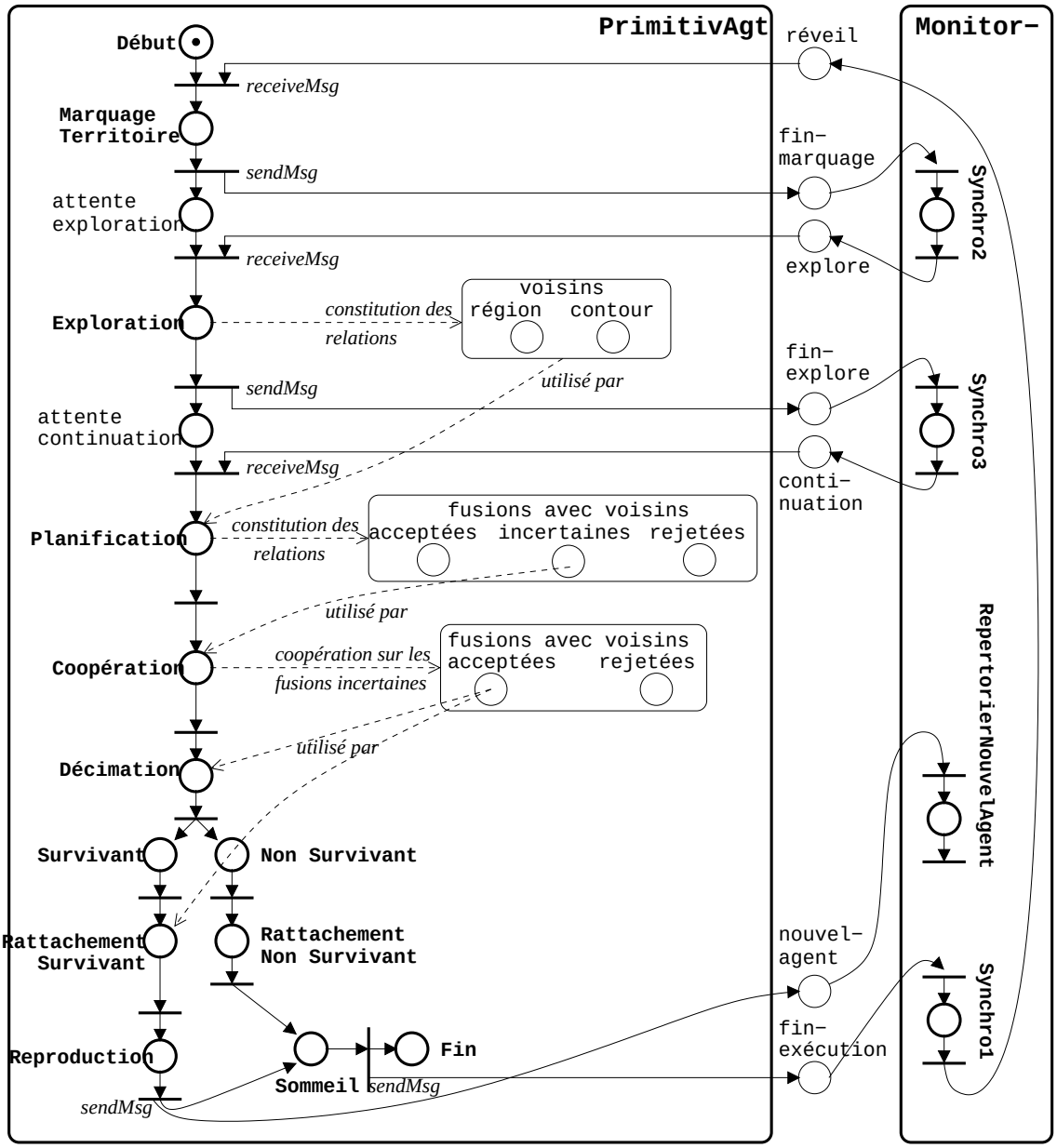


FIG. 9.2 : *Modèle RdP du contrôle des comportements des agents de la pyramide (`PrimitivAgt`).*

9.3 Les croyances de l'agent

Les accointances. Les agents, qu'ils soient de type région ou contour, vont lors du comportement d'exploration (§9.5 p. 181) enregistrer dans leur tableau noir local l'ensemble de leurs voisins (adjacent) dans l'image. L'enregistrement de ces liens prend la forme d'une relation d'accointance(§7.4.3.1 p. 147), étiquetée par le type (région, contour) du voisin. Si par exemple, un agent a_i est adjacent à l'agent région d'identifiant 43, a_i stockera l'information suivante :

```
(Acquaintance (id 43) (types adjacent region))
```

Les croyances des agents région contiennent, en plus des accointances, les attributs photométriques et géométriques caractérisant la région représentée par l'agent. L'agent a_s stocke ces attributs dans une structure de données contenant :

- μ_s : moyenne des niveaux de gris normalisés, c'est-à-dire (moyenne des niveaux de gris de la région) / 255 ;
- σ_s : écart-type des niveaux de gris normalisés ;
- h_s : histogramme ;
- S_s : surface (nombre de pixels) ;
- G_s : centre de gravité ;
- cr_s : les champs récepteurs de la région. La représentation des champs récepteurs est faite à l'aide d'un masque de bits évoqué (§8.2.1 p. 153).

Les croyances des agents contour contiennent la chaîne de segments de contour que l'agent représente. L'agent a_s dispose, pour chaque segment de contour sc_s^k, d'une structure de donnée codant les paramètres du segment de contour. À savoir :

- ω_s^k : orientation (coordonnées polaires) ;
- r_s^k : position par rapport à l'origine ;
- l_s^k : longueur ;
- g_s^k : gradient moyen normalisé le long du segment.

9.4 Comportement de marquage de territoire

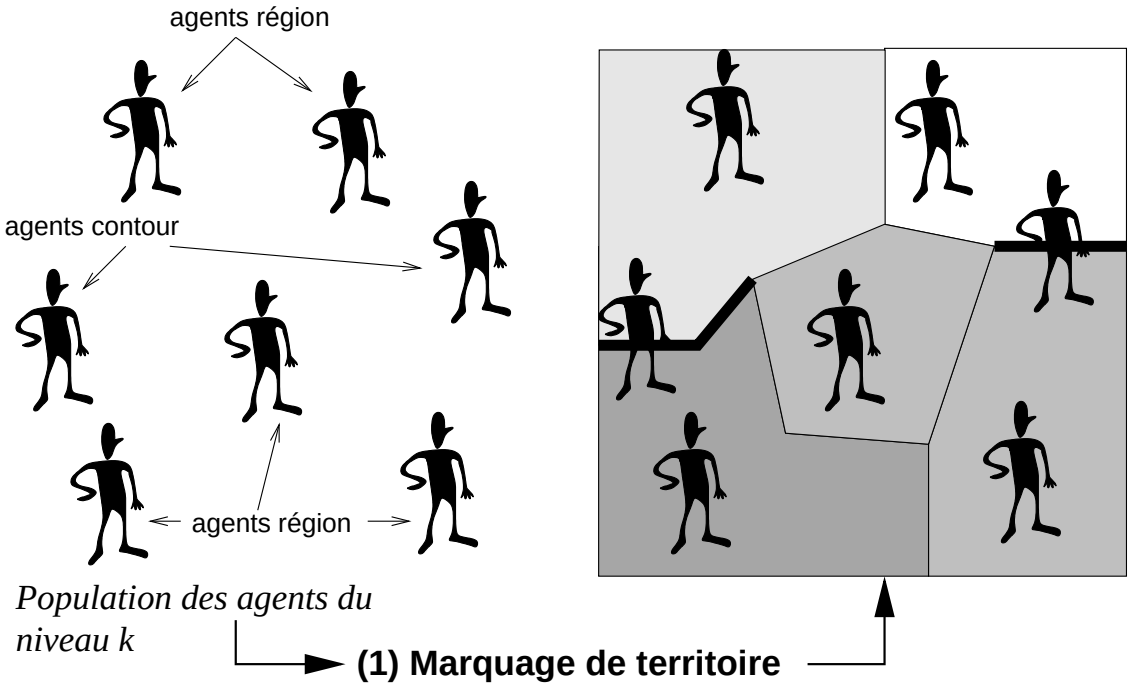


Fig. 9.3 : *Vision schématique du comportement de marquage du territoire.*

Chaque agent représente une primitive région ou contour présente dans l'image. Afin de pouvoir relier les agents représentant des primitives adjacentes, il est tout

d'abord nécessaire que les agents marquent de leur identifiant la zone de l'environnement correspondant à leur primitive.

Cependant, il faut attendre que tous les agents du niveau précédent aient fini de se rattacher et de se reproduire. Lorsque l'activité sur le niveau précédent a cessé, l'agent qui sommeille, en est averti par la réception du message `réveil` provenant de l'agent moniteur (fig. 8.8, p. 173) & (fig. 9.2, p. 178).

Détails sur le contrôle portant sur ce spécialiste. Le comportement de marquage de territoire est implémenté par un seul spécialiste. La figure 9.4 présente la structure de contrôle du spécialiste correspondant au RdP (fig. 9.2, p. 178). Par la suite nous ne présenterons que le modèle de contrôle RdP des spécialistes, la mise en œuvre à base de règle étant implicite.

MARQUAGE DE TERRITOIRE
Conditions :
`(State Début)`
Événements :
`(Message (content réveil))`
Partie opératoire :
`(marquage-territoire)`
Changement d'état :
`(retract (State Début))`
`(assert (State attente-exploration))`
Commande :
`(sendMsg (Message ...`
`(content fin-marquage)))`

(a)

```
;; Conditions :
(State Début)
;; Événements :
(Message (content réveil))
 =>
Partie opératoire :
(marquage-territoire)
;; Changement d'état :
(retract (State Début))
(assert
     (State attente-exploration))
;; Commande :
(sendMsg (Message ...
         (content fin-marquage)))
```

(b)

FIG. 9.4 : *(a) Le spécialiste de marquage du territoire, à mettre en relation avec la figure 9.2 page 178 ; (b) la mise en œuvre du RdP sous forme de règle.*

Marquage de territoire des agents région. Une fois réveillé (fig. 9.2, p. 178), un agent région va marquer un territoire correspondant à sa région. Cette région est l'union des régions correspondantes aux agents qui se sont rattachés au survivant dont il est le père (voir rattachement figure 6.5 page 115). Le marquage s'effectue dans l'agent environnement sur la carte d'identifiants du niveau courant `reg_map(`k`)`, où k est le niveau courant de la pyramide.

Marquage de territoire des agents contour. De même, une fois réveillé, un agent contour va marquer un territoire correspondant à sa chaîne des segments de contour. Le marquage s'effectue dans l'agent environnement sur la carte d'identifiants du niveau courant `edge_map(`k`)`.

Chaque agent va signaler à l'agent moniteur qu'il a bien fini son marquage de territoire à l'aide d'un message `fin-marquage` (fig. 9.2, p. 178).

Ce marquage de territoire dans un environnement commun à tous les agents du niveau courant, ancre les agents dans l'image et permet de délimiter la zone de l'image dont l'agent (région/contour) est le représentant. Cette identification de territoires dans l'environnement va permettre aux agents de se rencontrer.

9.5 Comportement d'exploration

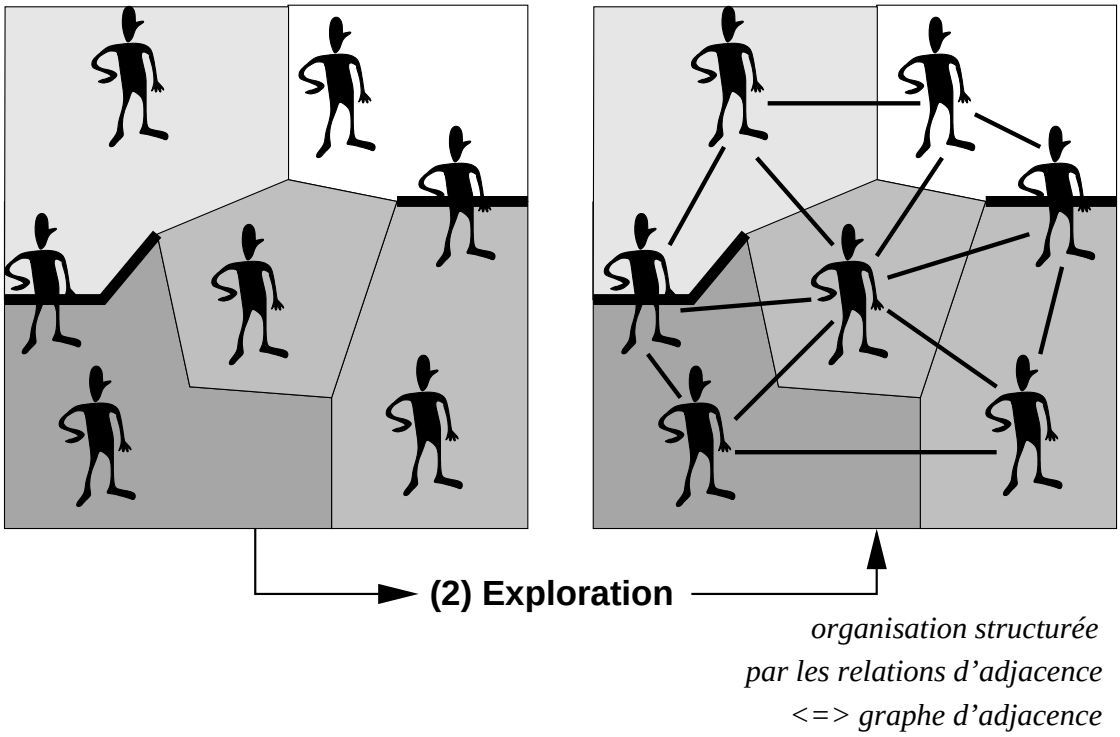


Fig. 9.5 : *Vision schématique du comportement d'exploration.*

Dans l'objectif de constituer les agents en une organisation structurée par les relations d'adjacence dans l'image, les agents vont explorer les alentours de leurs territoires pour y rencontrer leurs voisins dans l'image.

Toutefois, il faut attendre que tous les agents du niveau courant de la pyramide aient bien fini de marquer leurs territoires respectifs. Cette synchronisation globale (la deuxième) est menée par l'agent moniteur (fig. 8.8, p. 173), qui réveille tous les agents du niveau courant à l'aide d'un message `explore`(fig. 9.2, p. 178).

Exploration des agents contour. Un agent contour examine les sites des cartes `reg_map(`k`)` et `edge_map(`k`)`[1] qui appartiennent à des zones de focalisation. L'agent

[1]Rappelons que ces cartes représentent les territoires des agents région et contour.

contour construit une zone de focalisation elliptique (fig. 9.6, p. 182) pour chacun de ses segments de contour. Les rayons de l'ellipse sont fonctions de la longueur du segment.

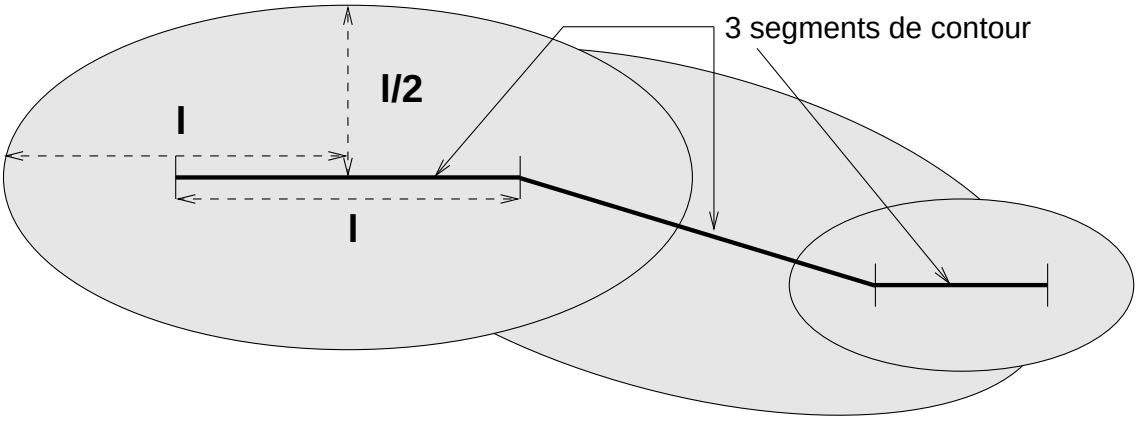


FIG. 9.6 : *Les zones de focalisation d'un agent contour comportant une chaîne de trois segments de contour.*

Chaque nouvel identifiant rencontré dans `edge_map(`k`)` correspond à un agent contour voisin et débouche sur la création d'une relation d'accointance étiquetée "contour" :

```
(Acquaintance (id ?voisin-id) (types adjacent edge))
```

Tandis que les nouveaux identifiants rencontrés dans `reg_map(`k`)` correspondent à des agents région voisins.

L'agent contour prend contact. Les agents contour doivent se faire connaître[2] des agents région dont ils ont rencontré l'identifiant dans l'environnement. Cette prise de contact prend la forme d'un message :

```
(Message (performatif tell) (sender ?s) (receiver ?r)
         (content (Acquaintance (id ?s) (types adjacent edge)))
)
```

Les zones de focalisation étant différentes pour chaque agent contour, un agent contour c_1 peut connaître un agent contour c_2 sans que la réciproque soit vraie (fig. 9.7).

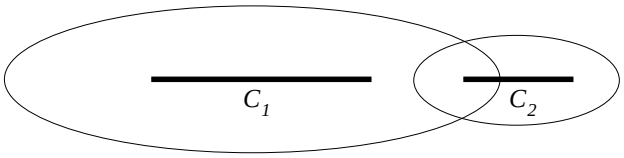


FIG. 9.7 : *Deux segments c_1, c_2 et leurs zones de focalisation respectives.*

[2]Un agent A connaît un agent B s'il dispose avec B d'une relation d'accointance. Celle-ci est créée soit par une rencontre de l'identifiant dans l'environnement (rencontre indirecte), soit par une prise de contact (rencontre directe).

Ainsi, un agent contour va adresser un message de présentation (voir ci-dessus) aux agents contour qu'il connaît par rencontre indirecte.

Cette démarche de présentation assure de disposer de relations de connaissance symétriques. La connaissance dans le cas présent est porteuse d'une sémantique particulière : l'adjacence dans l'image.

Exploration des agents région. Les agents région vont explorer les sites de `reg_map(`k`)` pour y découvrir leurs voisins de type région. La zone explorée correspond aux sites adjacents le long des sites frontières de la région. Ils ne cherchent pas à rencontrer les agents contour, ces derniers se chargeront de prendre contact avec eux.

Chaque agent va ensuite signaler à l'agent moniteur qu'il a bien fini son exploration à l'aide d'un message `fin-explore` (fig. 9.2, p. 178).

Une organisation d'agents selon l'adjacence dans l'image. Cette exploration a permis aux agents de découvrir leurs voisins dans l'image et ainsi de construire les relations d'accointance régions/régions, régions/contours, contours/contours. La population des agents d'un niveau est désormais organisée par les relations d'adjacence dans l'image.

Cette organisation $O^k_{adj}(P^k, A^k_{adj})$, (§8.3.2 p. 162) définit pour chaque agent les voisins avec lesquels il va interagir, pour coopérer et parfois fusionner.

9.6 Comportement de planification des fusions

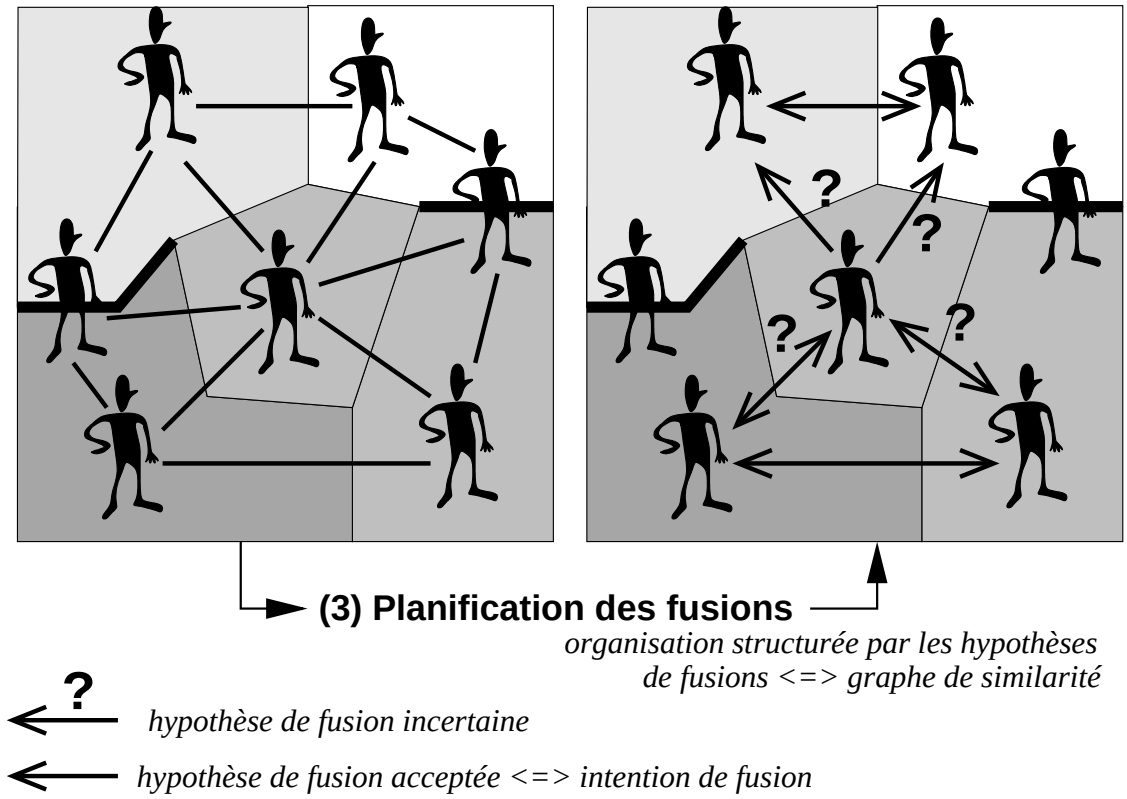


FIG. 9.8 : *Vision schématique du comportement de planification des fusions.*

Les liens reflétant l'adjacence spatiale entre agents d'un même niveau étant tissés, les agents vont tâcher de déterminer les voisins avec lesquels ils souhaitent fusionner.

Un agent a_s doit donc dresser un **plan des actions** qu'il souhaite entreprendre avec ses voisins. Ce plan est constitué de trois listes dans lesquelles l'agent classe ses voisins de même nature[3] :

1. `rejet`(a_s) : les voisins avec lesquels l'agent a_s ne veut pas fusionner.
2. `incertitude`(a_s) : les voisins avec lesquels l'agent a_s n'est pas certain s'il doit fusionner ou non.
3. `accept`(a_s) : les voisins avec lesquels l'agent a_s souhaite fusionner. Cette liste est appelée **plan des fusions** de l'agent a_s.

Étant donné que pour l'instant seuls les agents région fusionnent entre eux, ce comportement ne concerne pas les agents contour. Mais les principes et protocoles énoncés peuvent s'appliquer à des agents fusionnant des chaînes de segments de contour. Dans ce cas, seuls les attributs et les critères d'évaluation du désir (similarité) changent.

[3]Un agent région va chercher à fusionner avec d'autres agents région.

La construction de ce plan va s'effectuer en deux temps :

1. Récupérer les attributs des agents voisins (de même nature) et déduire, pour chacun de ses voisins, un vecteur **d'affinités** fonction de ses attributs propres et de ceux du voisin.
2. Déduire du vecteur d'affinité un **désir** de fusion pour chaque voisin, puis classer les voisins, en fonction du désir qui leur est associé, dans les trois listes d'acceptation, de refus, ou d'incertitude vis-à-vis de la fusion. Comme nous le verrons (§9.6.5 p. 189) et (§9.6.4 p. 188), nous proposons une méthode **d'adaptation locale** et de prise en compte de **l'incertitude** du traiteur d'image à propos des seuils à utiliser pour classer les voisins.

Ces listes représentent une première ébauche d'une planification des actions qu'un agent souhaite effectuer avec ses voisins.

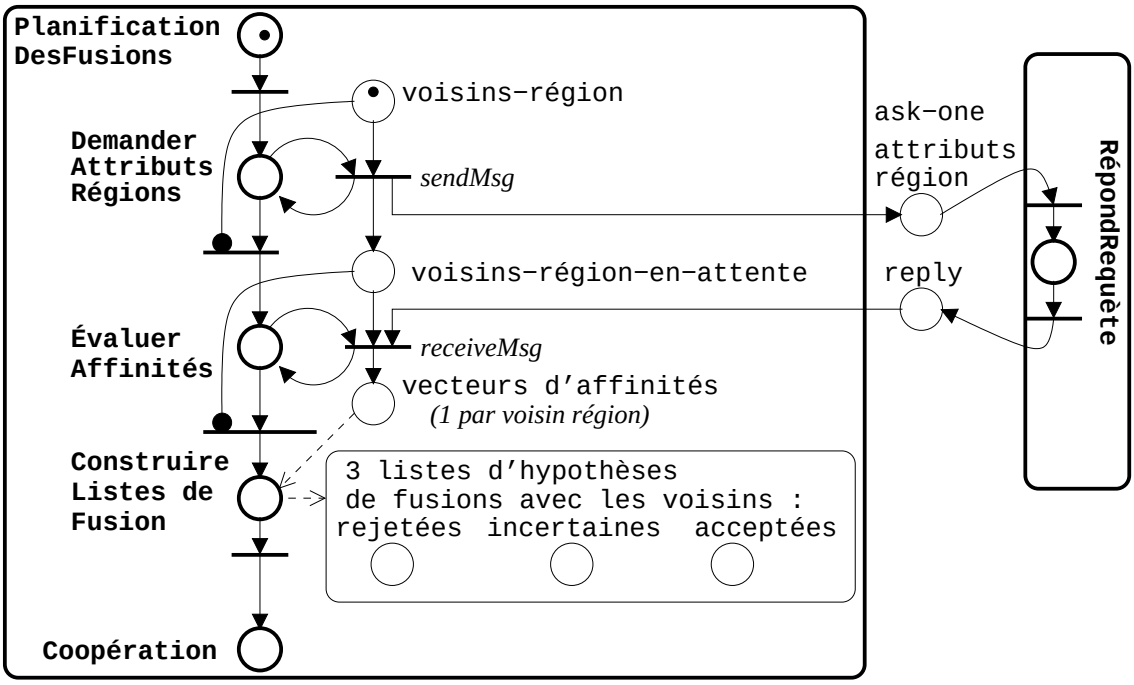


Fig. 9.9 : *Le comportement de planification des actions. Les voisins région sont classés dans une des trois listes en fonction du désir (similarité) éprouvé par l'agent courant.*

9.6.1 Calcul des affinités associées à chaque voisin

La première étape du comportement de planification consiste pour un agent région à récupérer les attributs région de ses voisins. Ainsi, chaque agent région va envoyer à ses voisins région (fig. 9.9, p. 185) un message du type :

```
(Message (performatif ask-one)(sender ?s)(receiver ?r)
         (reply-with attributs-région-from ?r)
         (content (attributs-région))
)
```

Le performatif `ask-one` (issu de la syntaxe KQML) spécifie la sémantique interrogative du message. Le champ `reply-with` permet d'identifier la réponse.

Ces requêtes envoyées, l'agent passe dans l'état `ÉvaluerAffinités` (fig. 9.9, p. 185). Dans cet état, l'agent attend les réponses et, à chaque réponse reçue, il évalue les affinités pour le voisin lui ayant répondu.

Définition 7 (affinités) *Les affinités d'un agent a_s pour un agent a_i se présentent sous la forme d'un vecteur a_{si}, représentant les similarités entre les deux agents, évaluées suivant plusieurs critères dans l'espace des attributs région.*

$$a_{si} = \Big(|\Delta\mu_{si}|, |\Delta\sigma_{si}|, fl_{si}, cont_{si}, r_{si}\Big)$$

Où :
- $|\Delta\mu_{si}| = |\mu_s - \mu_i|$ est la différence des moyennes (en niveau de gris) normalisées des régions ;
- $|\Delta\sigma_{si}| = |\sigma_s - \sigma_i|$ est la différence des écart-types (en niveau de gris) normalisés des régions ;
- fl_{si} est la proportion de la longueur de frontière avec a_i par rapport à la la longueur de frontière totale de a_s ;
- $cont_{si}$ est le contraste normalisé (divisé par 255) entre les deux régions (§2.4 p. 26) ;
- $r_{si} = \frac{h_s h_i}{\|h_s\|\|h_i\|}$ est la corrélation des histogrammes des deux régions.

9.6.2 Le désir de fusion

Définition 8 (Désir de fusion) *Le désir de fusion d'un agent a_s pour un agent a_i se présente sous la forme d'un scalaire $d_{si} \in [0, 1]$ évalué à partir des affinités. Ce désir de fusion représente l'impression générale de a_s sur la qualité d'une fusion avec a_i. Il peut être assimilé à l'inverse d'une distance dans l'espace des attributs région ou à la similarité entre les deux régions.*

En se basant sur le vecteur d'affinités, l'agent va calculer son désir de fusion pour chacun de ses voisins région.

$$d_{si} = \textsc{Eval\_désir}(a_{si}, (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)) \tag{9.1}$$

$$d_{si} = 1 - \Big[\alpha_1|\Delta\mu_{si}| + \alpha_2|\Delta\sigma_{si}| + \alpha_3(1 - fl_{si}) + \alpha_4 cont_{si} + \alpha_5(1 - r_{si})\Big] \tag{9.2}$$

Où les pondérations α_1, α_2, α_3,α_4, α_5, peuvent être fixées *a priori* par le traiteur d'image, ou adaptées dynamiquement et localement à l'aide d'une analyse en composante principale (§2.2.2 p. 20) permettant de favoriser les attributs localement les plus discriminants.

Définition 9 (Hypothèse de fusion) *L'hypothèse de fusion h_{si} d'un agent a_s pour un agent a_i est une structure de données qui rassemble toutes les informations dont*

dispose a_s à propos d'une fusion (hypothétique) avec a_i. Elle est structurée comme suit :

$$h_{si} \ := \texttt{(merge (agent } a_s\texttt{)(with } a_i\texttt{)(desire 0.93))}$$

Signifiant, dans ce cas, que le désir de fusion de l'agent a_s avec l'agent a_i est de 0.93, la valeur du désir étant comprise entre 0 (aucun désir) et 1 (désir maximum).

L'agent va constituer pour chacun de ses voisins une hypothèse de fusion.

9.6.3 Adaptation locale de l'importance des différentes affinités dans l'évaluation du désir

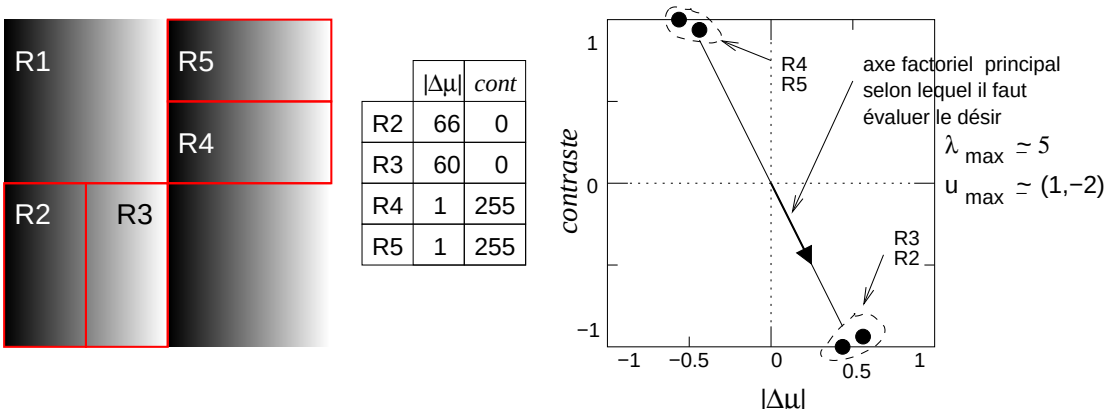


Fig. 9.10 : *À gauche : une image composée de deux zones de dégradé, avec cinq régions. Au centre : les valeurs non normalisées des affinités de R1 $|\Delta\mu|$ et contraste à la frontière. À droite : ces mêmes affinités centrées et réduites (pour ACP), ainsi que l'axe factoriel principal ; on remarque le fort pouvoir de discrimination de l'axe des contrastes à la frontière.*

Prenons l'exemple de la figure 9.10 composée de deux dégradés et dont la partition idéale finale doit représenter deux régions, une pour le dégradé de gauche et une pour celui de droite. Si l'agent représentant la région R1 utilise $|\Delta\mu|$ comme critère de calcul de désir, c'est-à-dire $\alpha_1 = 1$, $\alpha_j = 0, \forall j \in [2,5]$; alors il commettra une erreur en donnant un fort désir de fusion à R4 et R5 et un plus faible à R2 et R3.

Intuitivement, dans ce genre de situation, le traiteur d'image préconiserait l'utilisation des contrastes à la frontière au détriment de $|\Delta\mu|$, car les deux gradients ont la même moyenne et seul le contraste à la frontière les distingue. Cette intuition est confirmée par la lecture du graphe (fig. 9.10, p. 187) qui montre que le contraste est bien plus discriminant que $|\Delta\mu|$. En effet, une analyse en composante principale permet de dégager un axe factoriel principal s'étirant essentiellement dans l'axe des contrastes. Ainsi, une ACP (§2.2.2 p. 20) permet de déterminer automatiquement un coefficient de pondération à apporter à chaque affinité pour calculer le désir selon l'axe factoriel principal.

L'agent fait ainsi évoluer dynamiquement et localement ses critères d'évaluation du désir, afin de séparer au mieux les agents qui lui ressemble des autres. Nous ferons remarquer les conditions particulières de l'utilisation de l'ACP avec un faible nombre de points. Ces conditions traduisent la réalité du nombre de voisins moyen par agent (environ 5).

Cette fonctionnalité n'est encore qu'en phase de développement, si les résultats sont ceux attendus, il est nécessaire de compléter les expérimentations cf. chapitre 10.

9.6.4 L'incertitude

Dans les approches classiques de segmentation par pyramide irrégulière (chap. 3 p. 33), le traiteur d'image fixe un seuil de similarité pour construire le graphe de similarité sur lequel va s'effectuer la décimation.

Un cadre pour l'expression de l'incertitude. Le modèle standard des pyramides irrégulières ne favorise pas la prise en compte de l'incertitude du traiteur d'image sur le degré de similarité nécessaire pour autoriser une fusion entre deux sommets. Sur l'image du scanner du sein (fig. 10.2, p. 214), cette approche contraint le traiteur d'image à fixer un seuil. Or, le seuil nécessaire à une séparation correcte du muscle et du tissu glandulaire est extrêmement délicat à choisir et nécessite une analyse approfondie de l'image.

Les pyramides irrégulières classiques, sur ce type d'image, requièrent donc une information *a priori* très précise. Elles ne sont pas robustes (§10.2.4 p. 230) vis-à-vis d'une légère variation des seuils.

Or, cette variation traduit l'incertitude du traiteur d'image vis-à-vis des seuils à fournir afin de bien la segmenter.

Le système proposé fournit un cadre d'expression de l'incertitude du traiteur d'image à travers une formulation inspirée de la théorie de Dempster-Shafer (§4.6.2.2 p. 78), dans laquelle un intervalle $[D_n, D_s]$, traduit l'incertitude du traiteur d'image sur la similarité nécessaire à une fusion entre deux régions.

Permettre la fusion d'informations. Comme dans Dempster-Shafer, le cadre d'expression de l'incertitude du système fournit simultanément un cadre de fusion de l'information. Or, cette fusion d'informations mise en œuvre par des comportements coopératifs région/région et région/contour favorise la robustesse de l'approche vis-à-vis de légères variations dans les images à segmenter. L'approche agent réussit à segmenter, pour un même paramétrage, deux images similaires là où une pyramide irrégulière classique requiert une intervention humaine pour chacune des images (fig. 10.7, p. 219) & (fig. 10.9, p. 222).

Le traiteur d'image va donc fournir deux seuils de désir :
- D_n : le **désir nécessaire** est le seuil à partir duquel une fusion est plausible ;
- D_s : le **désir suffisant** est le seuil à partir duquel une fusion est crédible, avec $0 \leq D_n \leq D_s \leq 1$.

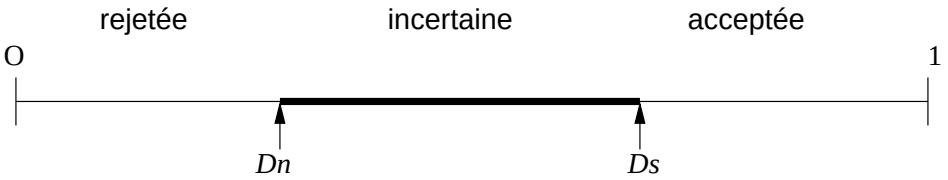


Fig. 9.11 : *Les deux seuils de désir bornant l'intervalle d'incertitude vis-à-vis d'une fusion.*

L'agent dispose de trois listes dans lesquelles il va classer les hypothèses en fonction du désir calculé et des seuils.

- `rejet` est une liste dans laquelle l'agent insérera les hypothèses et donc les accointances avec lesquelles il ne veut pas fusionner ;
- `incertitude` est une liste dans laquelle il mettra les hypothèses et donc les accointances avec lesquelles il ne sait pas encore s'il doit fusionner ou non, **l'ambiguïté sera levée à l'aide d'un comportement coopératif** (§9.8 p. 192) ;
- `accept` contiendra finalement les hypothèses et donc les accointances avec lesquelles il veut fusionner. Ces hypothèses de fusion deviennent des **intentions de fusion** (déf. 10 p. 191).

Soit h_{si} l'hypothèse de fusion de l'agent courant s avec l'agent i et d_{si} le désir associé à cette hypothèse. Le classement des hypothèses s'effectue suivant la règle décrite à l'algorithme 9.1.

Classement_hypothèse(h_{si}, d_{si}, D_n, D_s)
- **Si** $(d_{si} < D_n)$ **Alors** `rejet`$(a_s) \leftarrow h_{si}$
- **Sinon Si** $(d_{si} < D_s)$ **Alors** `incertitude`$(a_s) \leftarrow h_{si}$
- **Sinon** `accept`$(a_s) \leftarrow h_{si}$

Algo. 9.1 : *Règle de classement des hypothèses dans les trois listes en fonction du désir.*

Afin de permettre un formalisme déclaratif, on ajoute à la structure hypothèse de fusion (déf. 9 p. 186) un dernier champ précisant la liste d'appartenance de l'hypothèse, par exemple :

$h_{si} \in$ `rejet`$(a_s) \Leftrightarrow$ `(merge (agent` a_s `)(with` a_i `)(desire 0.34) (list rejet))`

9.6.5 Adaptation locale du désir nécessaire

Un des inconvénients des pyramides irrégulières est l'apparition de fausses transitions et la disparition de certaines frontières. Ceci est causé par la dérive progressive des statistiques des régions. Cette dérive est provoquée par des fusions de qualité moyenne, respectant de justesse le seuil de similarité. On peut, pour limiter cet effet, effectuer les meilleures fusions en premier, retardant et empêchant peut-être la dérive des statistiques au fil des niveaux.

La solution proposée par Montanvert et al. [Mon91] consiste à calculer un seuil de similarité local plus restrictif que le seuil global. Nous appliquons cette technique au seuil D_n afin de le rehausser et ainsi rejeter les moins bonnes hypothèses. En prenant en compte les hypothèses dans les listes `incertitude` et `accept`, nous appliquons l'algorithme proposé dans [Mon91]. On cherche le seuil donnant la partition en deux classes des deux listes confondues, maximisant un critère de variance interclasses (§3.6.1.2 p. 43).

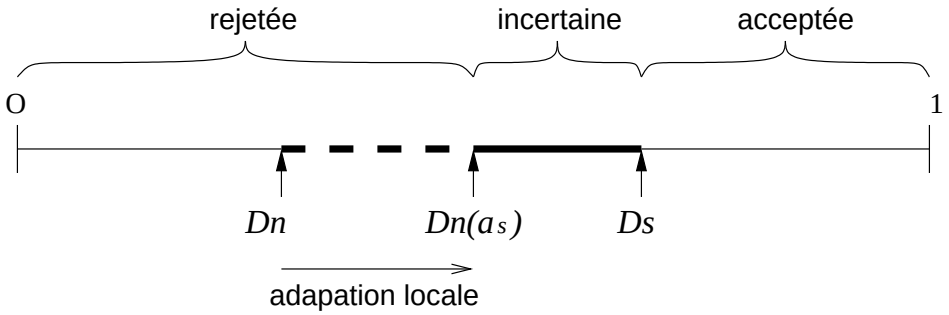


FIG. 9.12 : *Adaptation locale du seuil de désir nécessaire.*

D_n est remplacé par ce seuil local à l'agent a_s : $D_n(a_s)$ (fig. 9.12).

Remarque : si $D_n(a_s) \geq D_s$, nous avons opté pour une solution prudente qui consiste à supprimer D_s et donc l'intervalle d'incertitude, en ne laissant qu'un seul seuil $D_n(a_s)$ qui dans ce cas délimite les hypothèses rejetées des hypothèses acceptées. Une autre solution consisterait à fixer $D_n(a_s)$ à la valeur de D_s.

Les deux listes `incertitude` et `accept` sont filtrées à l'aide de ce nouveau seuil local : les hypothèses dont le désir associé est plus petit que le seuil local $D_n(a_s)$ sont transférées de la liste `incertitude` vers la liste `rejet`. La liste `accept` n'est modifiée que dans le cas évoqué ci-dessus où $D_n(a_s) > D_s$.

9.6.6 Synthèse sur la planification des fusions

Nous allons récapituler l'ensemble des étapes intervenant dans la construction des trois listes d'hypothèses de fusion :

1. l'agent débute ce comportement en demandant les attributs région de chacun de ses voisins région : `DemanderAttributsRegion` (fig. 9.9, p. 185) ;
2. l'agent attend les réponses, et calcule un vecteur d'affinités pour chaque réponse reçue : `ÉvaluerAffinités` (fig. 9.9, p. 185) ;
3. il va ensuite pouvoir constituer les trois listes d'hypothèses de fusion : `ConstruireListesDeFusion` (fig. 9.9, p. 185).

Une synthèse sur la construction des trois listes d'hypothèses de fusion est donnée (algo. 9.2 p. 191), nous allons en commenter les étapes :

1. calculer le vecteur de pondération des affinités ;

2. génèrer la liste `hyp_list` composée des hypothèses h_{si} de fusion ;
3. classer les hypothèses dans les trois listes `rejet`, `incertitude` et `accept` en fonction du désir d_{si} associé à l'hypothèse et des seuils globaux : D_n, D_s ;
4. adapter localement le seuil de désir nécessaire ;
6. reclasser les hypothèses en fonction du nouveau seuil ;
7. passer à l'état `Coopération`.

ConstruireListesDeFusion
Conditions :
`(State ÉvaluerAffinités) (vide (voisins-régions-en-attente))`
Partie opératoire :
/* *aff_list : liste des vecteurs d'affinités* */
/* *hyp_list : liste des hypothèses de fusion* */
1. $\overrightarrow{\omega} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5) \leftarrow$ Adapt_pondération_ACP(`aff_list`)
2. $\forall a_{si} \in$ `aff_list`,
   h_{si} := `(merge (agent` a_s `)(with` a_i `)(desire` d_{si} =Eval_désir$(a_{si}, \overrightarrow{\omega}))$)
3. $\forall h_{si} \in$ `hyp_list`, Classement_hypothèse(h_{si},d_{si}, D_n, D_s)
4. $D_n(a_s) \leftarrow$Adapt_désir_nécessaire(`aff_list`)
5. **si** $(D_n(a_s) \geq D_s)$ $D_s \leftarrow D_n(a_s)$
6. $\forall h_{si} \in$ `accept` $\cup$ `incertitude`,Classement_hypothèse(h_{si},d_{si},$D_n(a_s)$,D_s)
Changement d'état :
7.`(retract (State ÉvaluerAffinités))`
`(assert (State Coopération))`// *changement d'état*

Algo. 9.2 : *Spécialiste qui détermine le plan des actions constitué de trois listes* `rejet`, `incertitude` *et* `accept` *contenant les hypothèses de fusion rejetées, incertaines ou acceptées.*

9.7 Du graphe de similarité aux intentions de fusions

Les intentions de fusion organisent les agents. Pour chaque relation d'accointance de type adjacent entre agent de même nature, il existe une hypothèse de fusion qui caractérise la relation d'accointance par la liste à laquelle elle appartient.

Autrement dit, tous les voisins (de même nature) d'un agent a_s sont classés dans une des trois listes : `rejet`(a_s), `incertitude`(a_s) ou `accept`(a_s) qui est la liste des intentions de fusion.

Définition 10 (Intention de fusion) *L'intention de fusion IF_{si} de l'agent a_s pour a_i est une hypothèse de fusion que l'agent a_s considère souhaitable. Autrement dit, c'est une hypothèse de fusion h_{si} classée dans la liste* `accept`(a_s)

$$h_{si} \in \texttt{accept}(a_s) \Rightarrow IF_{si}$$

Définition 11 (Plan de fusion) *L'ensemble de ces intentions de fusion* `accept`(a_s) *d'un agent a_s forme son plan de fusion.*

L'union de tous ces plans de fusion parmi la population P^k des agents du niveau k forme l'ensemble des **relations des intentions de fusion** : A^k_{IF}.

$$A^k_{IF} = \bigcup_{a_s \in P^k} \texttt{accept}(a_s)$$

Ces relations structurent la population P^k en une organisation O^k_{IF} des agents ayant l'intention de fusionner les uns avec les autres :

$$O^k_{IF}(P^k, A^k_{IF})$$

Équivalence avec le graphe de similarité. Chaque intention de fusion IF_{si} d'un agent a_s correspond à un arc (s, i) reliant deux sommets similaires du graphe d'adjacence du niveau k. Ainsi, l'ensemble des arcs joignant les sommets de B^k (arcs du graphe de similarité) (§3.6.1 p. 42) est transposé dans l'espace des intentions de fusion A^k_{IF} :

$$B^k \Leftrightarrow A^k_{IF}$$

Finalement, le graphe de similarité $Sim^k(S^k, B^k)$ est transposé en une organisation d'agent $O^k_{IF}(P^k, A^k_{IF})$:

$$O^k_{IF}(P^k, A^k_{IF}) \Leftrightarrow Sim^k(S^k, B^k)$$

9.8 Comportement de coopération

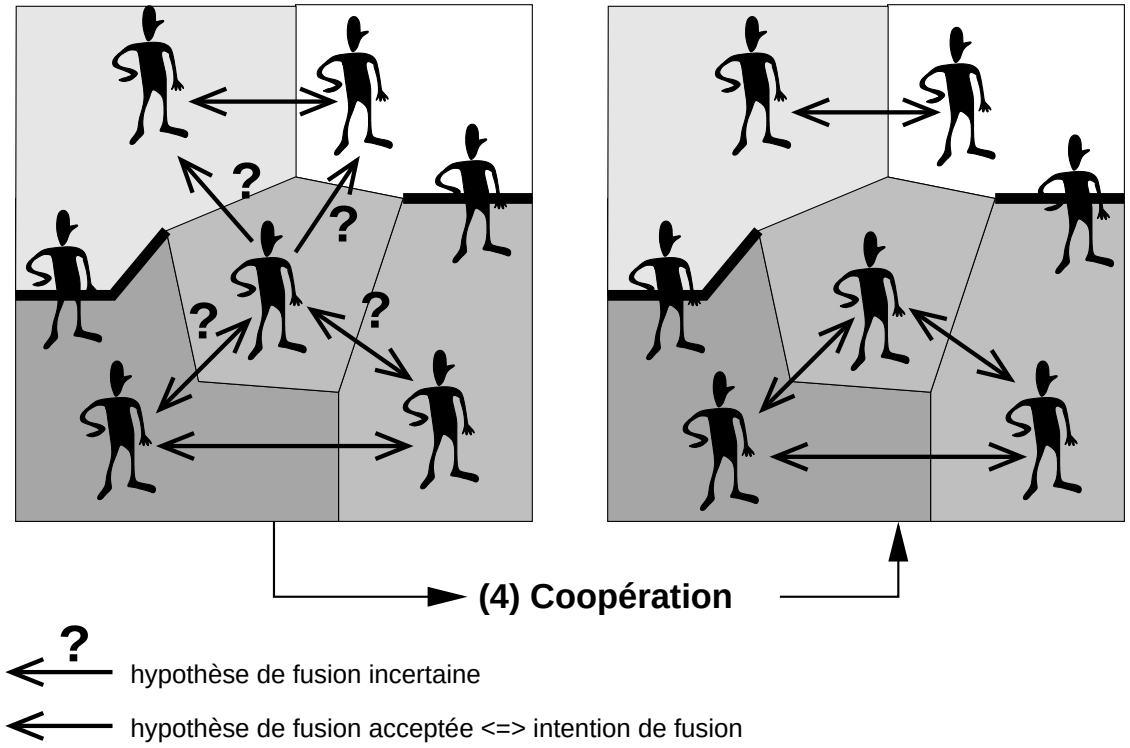


Fig. 9.13 : *Vision schématique du comportement de coopération.*

Afin de lever l'ambiguïté sur les hypothèses de fusions incertaines, l'agent engage avec ses voisins un débat (fig. 9.20, p. 201) visant à prendre en compte leurs points de vue à travers des critiques positives ou négatives.

Remarque : dans l'état de développement actuel, les agents contour se répètent à l'identique, niveau après niveau, ne fusionnant pas les uns avec les autres. Ils interviennent comme complément d'information à l'activité des agents région. Ainsi, deux types de coopération vont être engagées : une coopération région/région et une coopération région/contour.

Chaque agent région va demander pour chacune de ses hypothèses incertaines l'avis de ses voisins région et contour. Cette consultation est faite par l'envoi du message :

```
(Message (performatif ask-one) (sender ?s) (receiver ?r)
         (reply-with critique-clé ?clé)
         (content (critique-à-propos-de (merge (agent ?s)(with ?r))))
)
```

L'avis renvoyé va prendre la forme d'une critique c'est-à-dire une valeur positive ou négative concernant une hypothèse de fusion. L'agent ayant demandé l'avis des ses voisins va ensuite intégrer toutes les critiques reçues à propos d'une hypothèse incertaine et décider si l'hypothèse de fusion en question doit être rejetée ou acceptée. Cette intégration est une procédure de fusion d'information qui justifie l'approche inspirée de Dempster-Shafer que nous avons choisie.

Par sa nature, cette coopération est **confrontative** (§5.4.1 p. 93) tandis que dans sa forme elle est **opportuniste**, car elle n'est engagée que pour venir **combler l'incertitude** du traiteur d'image.

9.8.1 Coopération région/région

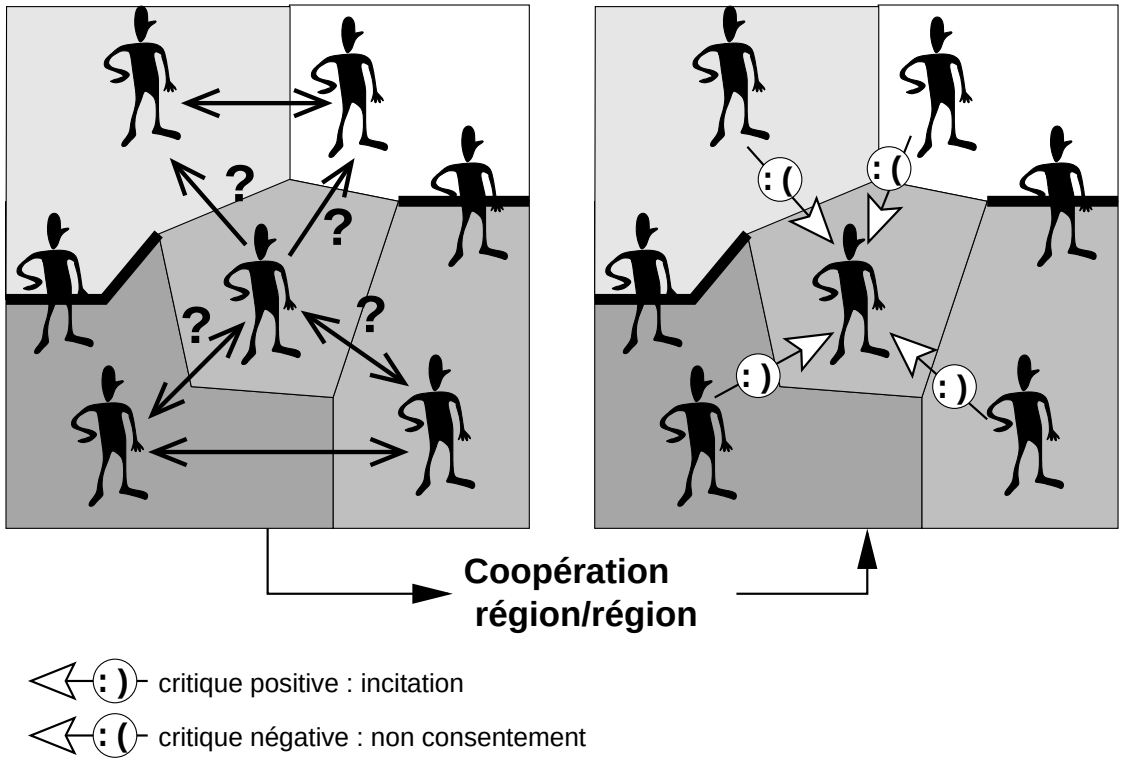


Fig. 9.14 : *Vision schématique de la coopération région/région.*

Un agent région a_s ayant une hypothèse de fusion incertaine h_{sr} à propos de l'agent région a_r, demande à a_r son avis sur cette hypothèse de fusion.

L'agent région interrogé à propos d'une fusion le concernant va, suivant son opinion personnelle, appliquer soit la règle du non consentement (algo. 9.3 p. 195) soit la règle de l'incitation (algo. 9.4 p. 195).

La règle du non consentement. Une conséquence de l'adaptation locale (§9.6.5 p. 189) est d'orienter la société d'agents O^k_{IF} organisée par les intentions de fusions.

Ainsi, un agent a_r ayant exclu a_s de son plan de fusion ($h_{rs} \in \texttt{rejet}(a_s)$), car il a de meilleures fusions à faire, pourrait se trouver absorbé par a_s allant alors à l'encontre des motivations énoncées lors de l'adaptation locale. Pour empêcher ce phénomène, nous utilisons une règle simple dite **du non consentement** (algo. 9.3 p. 195).

Cette règle permet à un agent recevant une demande d'avis le concernant d'interdire une fusion s'il est non consentant, c'est-à-dire que l'autre agent impliqué dans la fusion fait parti des accointances rejetées.

La critique renvoyée dans ce cas est la valeur la plus négative possible : `-MAX_NUMBER`.

```
NON_CONSENTEMENT
(Message (reply-with critique-clé ?clé)(content
(critique-à-propos-de
(merge (agent ?s)(with ?r=Self)))))
(merge (agent ?r)(with ?s)(list rejet))
⇒
sendMsg(Message ...(receiver ?s)(reply-to critique-clé ?clé)
(content (critique  -MAX_NUMBER)))
```

ALGO. 9.3 : *Règle du non consentement appliquée par l'agent courant noté* `Self` *qui vérifie que l'hypothèse de fusion reçue en message le concerne bien. Une clé est utilisée pour identifier l'hypothèse sur laquelle porte la critique.*

La règle de l'incitation. Si un agent a_r souhaite fusionner avec un agent a_s, et qu'il reçoit une demande d'avis sur leur fusion venant de ce dernier, a_r va inciter l'agent incertain a_s à valider cette hypothèse de fusion. Pour cela, a_r va communiquer son désir de fusion avec a_s, incitant a_s, à valider cette fusion, d'autant plus que son désir est grand (algo. 9.4 p. 195).

```
INCITATION
(Message (reply-with critique-clé ?clé)(content
(critique-à-propos-de
(merge (agent ?s)(with ?r=Self)))))
(merge (agent ?r)(with ?s)(list accept)(desire ?d))
⇒
sendMsg(Message ...(receiver ?s)(reply-to critique-clé ?clé)
(content (critique ?d)))
```

ALGO. 9.4 : *Règle où l'agent* `?r` *incite* `?s` *à valider la fusion en question.*

9.8.2 Coopération région/contour

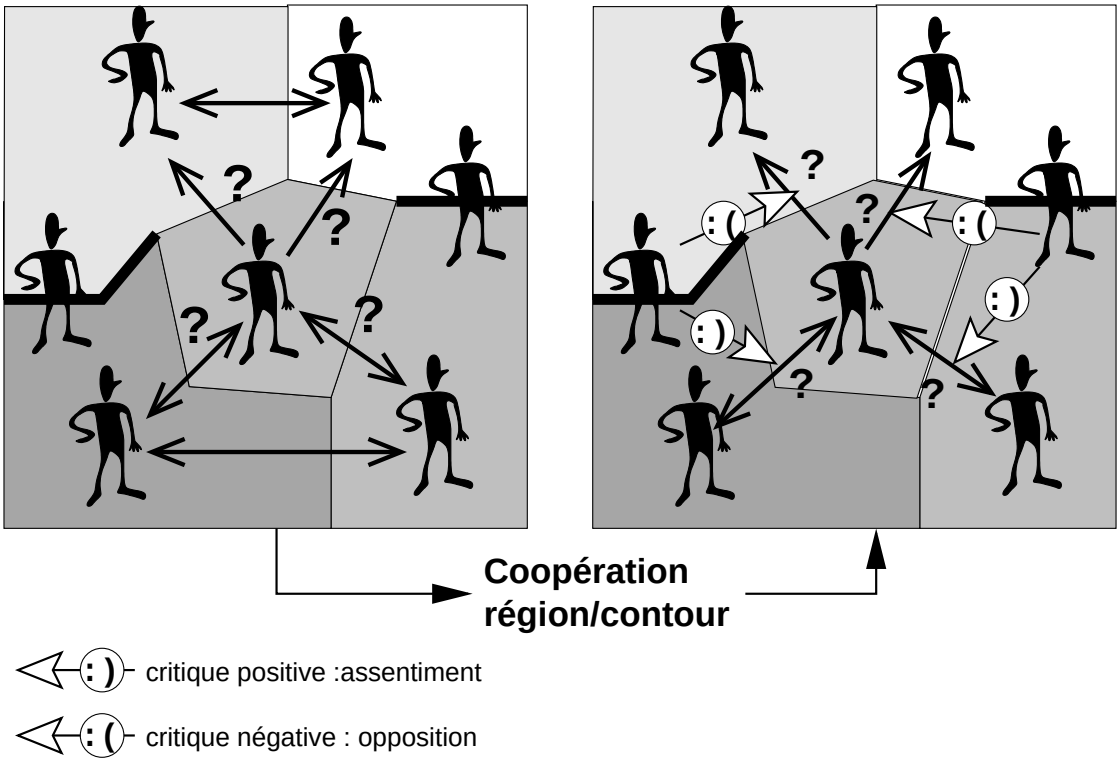


Fig. 9.15 : *Vision schématique de la coopération région/contour.*

Comme évoqué dans le chapitre 6, il est parfois indispensable de pouvoir exploiter l'information de contour dans le cas où les statistiques des régions ne permettent pas de trancher à propos de la fusion de deux régions.

La difficulté réside dans la formulation du cadre d'expression de cette coopération ; à cet effet, nous proposons une approche naturelle, car inspirée des activités sociales, dans laquelle un agent région va demander l'avis aux agents contour dans son voisinage lorsque se présente une situation ambigue (fusion incertaine) .

Ainsi, l'agent contour a_k recevant une demande de critique à propos de l'hypothèse de fusion incertaine h_{ij} (concernant les agents région a_i et a_j) va calculer une première valeur de critique pour chaque segment de contour s qu'il contient. Cette critique est fonction de l'angle $\omega_{sij} \in [0, \pi/2]$ formé par le segment joignant les barycentres des deux régions et le segment de contour s (fig. 9.16, p. 197) & (fig. 9.18, p. 198).

Pour chaque segment, comme pour la coopération région/région, deux situations peuvent se présenter :

1. la critique sera défavorable : règle de l'opposition ;
2. la critique sera favorable : règle de l'assentiment.

La règle de l'opposition est déclenchée si la ligne supportant le segment de contour coupe le segment joignant les barycentres des deux régions (fig. 9.16, p. 197). Cela signifie que le segment de contour tend à séparer les deux régions, la critique sera donc négative.

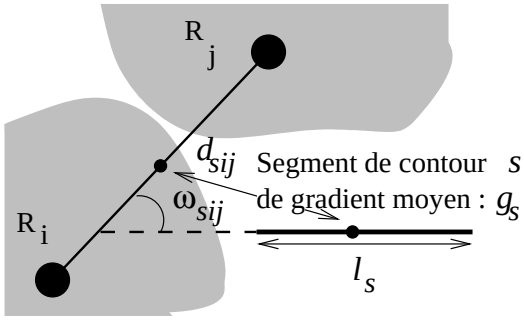


Fig. 9.16 : *Situation engendrant une critique négative (règle de l'opposition).*

Une première valeur de critique négative est calculée selon la formule ci-dessous traduisant la fonction de la figure 9.17.

$$c_k^s(h_{ij}) = \text{OPPOSITION}(\omega_{sij}) \tag{9.3}$$

$$c_k^s(h_{ij}) = -\begin{cases} \frac{4*\omega_{sij}}{\pi} & \textbf{si } 0 \leq \omega_{sij} \leq \pi/4 \\ 1 & \textbf{si } \pi/4 < \omega_{sij} \leq \pi/2 \end{cases} \tag{9.4}$$

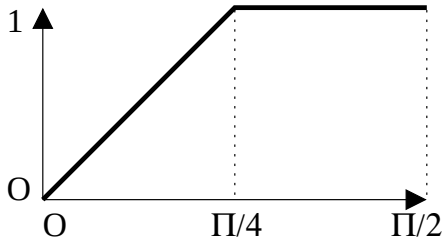


Fig. 9.17 : *Première valeur de critique négative en fonction de l'angle.*

La fonction OPPOSITION permet d'avoir une critique d'autant plus négative que le segment de contour coupe perpendiculairement le segment joignant les deux barycentres.

La règle de l'assentiment est déclenchée si la ligne supportant le segment de contour ne coupe pas le segment joignant les barycentres des deux régions. Dans ce cas (fig. 9.18, p. 198), le segment de contour laisse sur sa droite ou sur sa gauche les barycentres des deux régions, auquel cas il favorise la fusion des deux régions par une critique positive.

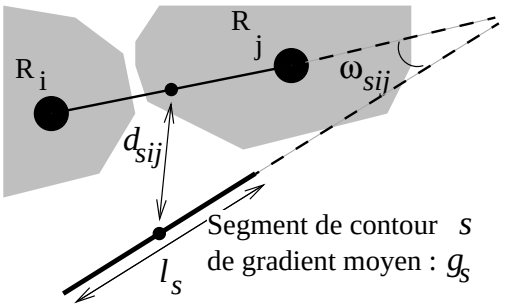


FIG. 9.18 : *Situation engendrant une critique positive (règle de l'assentiment).*

Une première valeur de critique positive est calculée selon la formule ci-dessous traduisant la fonction de la figure 9.19.

$$c_k^s(h_{ij}) = \text{ASSENTIMENT}(\omega_{sij}) \tag{9.5}$$

$$c_k^s(h_{ij}) = \begin{cases} 1 & \textbf{si } 0 \leq \omega_{sij} \leq \pi/4 \\ -\frac{4*\omega_{sij}}{\pi} + 2 & \textbf{si } \pi/4 < \omega_{sij} \leq \pi/2 \end{cases} \tag{9.6}$$

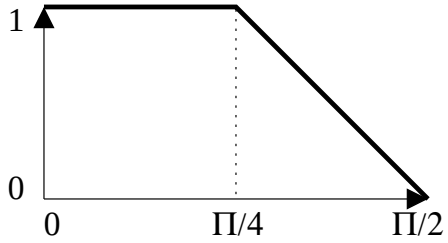


FIG. 9.19 : *Première valeur de critique positive en fonction de l'angle.*

La fonction ASSENTIMENT permet d'avoir une critique d'autant plus positive que le segment de contour est parallèle au segment joignant les deux barycentres.

L'agent contour intègre les critiques issues de ses segments. Afin de formuler une critique unique à propos d'une hypothèse incertaine h_{ij}, l'agent contour intègre les critiques issues de ses segments suivant la formule :

$$c_k(h_{ij}) = \sum_s \frac{c_k^s(h_{ij}) * g_s * l_s}{d_{sij}\sqrt{d_{sij}}} \tag{9.7}$$

Dans laquelle g_s est le gradient moyen le long du segment s, l_s sa longueur et d_{sij} la distance séparant le milieu du segment de contour s du milieu du segment joignant les barycentres des deux régions. Ainsi, la critique venant d'un segment de contour est d'autant plus forte en valeur absolue que le segment de contour est long et présente un fort gradient. La critique sera d'autant plus faible que le segment de contour est éloigné des régions.

Le degré du dénominateur 3/2 atténue un peu plus que linéairement l'influence des segments contours, il a été obtenu empiriquement après une série de tests avec les degrés 1, 3/2 et 2.

9.8.3 Fusion des critiques

Lorsque toutes les requêtes de critiques ont reçu une réponse, l'agent émetteur de la requête va fusionner toutes les critiques à propos de chaque hypothèse incertaine h_{ij}, voir `FusionnerCritiques` (fig. 9.20, p. 201).

À cet effet, il va regrouper les critiques selon deux sommes : d'un côté, toutes les critiques positives et, de l'autre les négatives :

$$c^+(h_{ij}) = \sum_{k:\{c_k(h_{ij})\geq 0\}} c_k(h_{ij}) \qquad \text{et} \qquad c^-(h_{ij}) = \sum_{k:\{c_k(h_{ij})<0\}} c_k(h_{ij})$$

$c^+(h_{ij})$ intègre les critiques de type incitation ou assentiment provenant des voisins tandis que $c^-(h_{ij})$ intègre les critiques de type non consentement ou opposition.

9.8.4 Reclassement des hypothèses incertaines

Sur la base des deux valeurs $c^+(h_{ij})$ et $c^-(h_{ij})$, l'agent ayant émis des demandes de critiques va reclasser l'hypothèse incertaine h_{ij} dans une des deux listes `accept` ou `rejet`. Voir `ReclasserHypothèses` (fig. 9.20, p. 201).

Ainsi, une hypothèse sera reclassée dans le plan de fusion (liste `accept`) si les valeurs positive et négative des critiques (associée à l'hypothèse) respectent des contraintes établies par une politique plus ou moins **prudente** fixée par l'utilisateur.

Ici, la prudence consiste à recevoir beaucoup d'assurances (critiques positives) comparées aux oppositions (critiques négatives) avant de transférer l'hypothèse de la liste `incertitude` à la liste `accept`. Trop peu de prudence peut mener à une fusion erronée, tandis que trop de prudence, dans le meilleur des cas, retarde d'un ou plusieurs niveaux une bonne fusion, et dans le pire des cas provoque une sur-segmentation de l'image.

L'utilisateur peut donc fixer, en cochant des cases via l'interface graphique, le niveau ou politique de prudence de ses agents, en fonction des résultats obtenus. Les différentes politiques sont les suivantes :

- **Politique non prudente :** les critiques doivent être globalement positives

  si $\left(c^+(h_{ij}) + c_k^-(h_{ij}) > 0\right)$ `accept` $\leftarrow h_{ij}$ **sinon** `rejet` $\leftarrow h_{ij}$

- **Politique prudente :** les critiques positives doivent être significativement plus importantes que les critiques négatives

  si $\left(c^+(h_{ij}) > k * |c_k^-(h_{ij})|\right)$ `accept` $\leftarrow h_{ij}$ **sinon** `rejet` $\leftarrow h_{ij}$

  Où k est un coefficient de prudence : pour une valeur de k élevée peu de fusions incertaines aboutiront dans le plan de fusion.

- **Politique très prudente :** aucun agent ne s'oppose à la fusion

  si $\left(c_k^-(h_{ij}) = 0\right)$ `accept` $\leftarrow h_{ij}$ **sinon** `rejet` $\leftarrow h_{ij}$

- **Politique extrêmement prudente :** personne ne s'y oppose et on a au moins l'assentiment d'un agent

  si $\left(c_k^-(h_{ij}) = 0$ **et** $\ c^+(h_{ij}) > 0\right)$ `accept` $\leftarrow h_{ij}$ **sinon** `rejet` $\leftarrow h_{ij}$

Justification du formalisme choisi. La problématique de prise en compte des critiques consiste à fusionner un ensemble de variables : $\omega_{sij}, g_s, l_s, d_{sij}$ afin de prendre une décision au plus tard pour un problème d'incertitude.

La logique floue (§4.6.2.3 p. 81) est un outil adapté au problème de fusion d'information ; cependant, la complexité calculatoire, évaluée lors de tests, pénalise grandement les temps de calculs. Nous avons donc opté pour une heuristique inspirée des ensembles flous. Cette parenté est bien visible sur les variables OPPOSITION (équation 9.4 & fig. 9.17) et ASSENTIMENT (équation 9.6 & fig. 9.19). Quant au mécanisme de combinaison de variables de l'équation 9.7, il utilise le produit au lieu du classique minimum.

Les différentes politiques plus ou moins prudentes de prise en compte des critiques d'incitation/assentiment et d'opposition/non-consentement représentent une heuristique inspirée de Dempster-Shafer (§4.6.2.2 p. 78). Le choix de celle-ci à la place du modèle original a été motivé pour des raisons de complexité algorithmique et des problèmes survenant lorsque les sources ne sont pas indépendantes, comme c'est le cas dans notre situation.

9.8.5 Retour sur le consentement mutuel

À la fin des différentes étapes de coopération, l'union des plans de fusion est équivalente à un graphe de similarité orienté. Ainsi, comme nous le suggérons (§8.2.4 p. 155), il faut retirer l'orientation du graphe.

En termes agent, cela se traduit par la notion de consentement mutuel entre agents. Ainsi, chaque agent a_i vérifie par un envoi de message que chaque intention de fusion IF_{ij} (hypothèse présente dans la liste `accept`(a_i)) est bien partagée par l'agent a_j. Si ce n'est pas le cas, il retire cette intention de la liste `accept`(a_i).

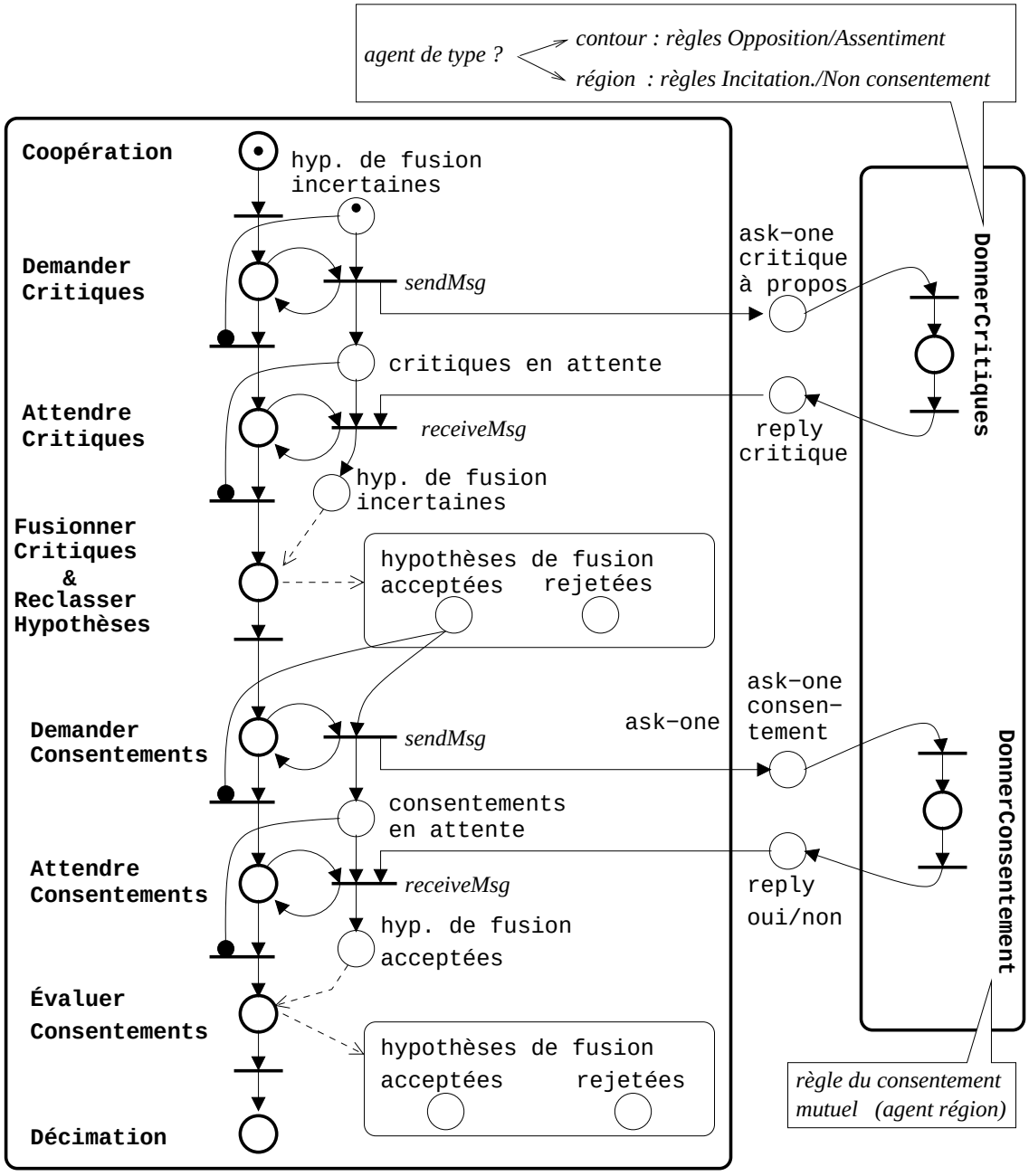


FIG. 9.20 : *Réseau de Pétri du comportement de coopération.*

9.9 Comportement de décimation

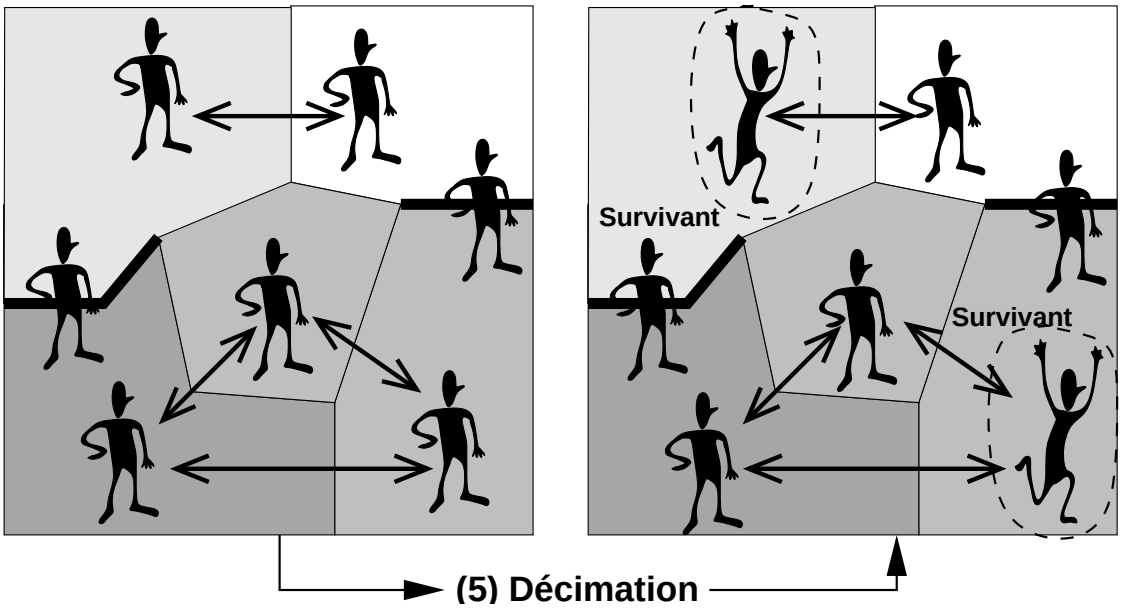


Fig. 9.21 : *Vision schématique du comportement de décimation.*

Les deux derniers comportements ont permis tout d'abord de construire une première ébauche d'un plan de fusion, puis, à travers un comportement coopératif, de lever les ambiguïtés pour aboutir à un plan `accept`(a_s) contenant toutes les intentions de fusion de l'agent a_s.

Comme nous l'évoquions (§9.7 p. 191), l'ensemble les intentions A_{IF}^k de fusion structure la population P^k des agents du niveau k en une organisation $O_{IF}^k(P^k, A_{IF}^k)$ équivalente au graphe de similarité des pyramides irrégulières.

Afin de passer au prochain niveau de la pyramide d'agents, la société des agents du niveau courant doit respecter les lois édictées par les pyramides irrégulières. Pour cela, il est nécessaire de décimer un certain nombre d'agents dans l'organisation $O_{IF}^k(P^k, A_{IF}^k)$. Afin d'assurer une décimation suffisante mais également la représentativité de tous les agents au niveau suivant, un protocole distribué de négociations [Duc00] entre agents respectant les règles énoncées (§3.5 p. 38) est engagé.

Ce protocole va permettre l'élection d'agents survivants qui auront le droit de se reproduire (§9.11 p. 209) en créant un nouvel agent au niveau suivant.

9.9.1 Calcul de l'utilité d'un plan

Dans le prolongement de l'idée présentée dans [Mon91] qui propose l'utilisation d'un opérateur d'intérêt afin d'adapter la décimation au contexte de l'image, nous proposons de favoriser la survie des agents ayant un bon plan de fusion.

Chaque agent va donc calculer **l'utilité**[4] associée à son plan de fusion. L'utilité u_s du plan de fusion de l'agent a_s peut-être évaluée par la somme des désirs des hypothèses qu'il contient

$$u_s = \sum_{i:h_{si} \in \texttt{accept}(a_s)} d_{si}$$

Comme le désir associé à une hypothèse de fusion reflète la similarité entre les deux agents impliqués dans la fusion, l'utilité d'un plan de fusion traduit bien la qualité globale des fusions proposées par l'agent.

L'utilisation de ce critère permet donc d'adapter à l'image la décimation, en favorisant les meilleurs regroupements d'agents.

9.9.2 Transposition agent de la procédure de décimation

L'algorithme 8.3 p. 157, présente la procédure de décimation dans un cadre distribué de machine à mémoire partagée, où des tâches (thread) concurrentes, une par sommet, appliquent une décimation que l'on peut qualifier d'adaptée (elle n'est pas stochastique mais basée sur l'utilité des plans de fusion).

Notre objectif est de transposer cet algorithme dans un cadre agent, afin d'obtenir un protocole de décimation distribuée modélisé sous la forme d'un RdP (fig. 9.22, p. 205).

Pour cela, nous proposons, tout d'abord, une réécriture "agent" (algo. 9.5 p. 204) de l'algorithme de décimation présenté (algo. 8.3 p. 157). Cette réécriture permet de condenser l'algorithme et d'introduire les premiers concepts agent comme la communication par envoi de messages.

Rappel : l'algorithme est une procédure itérative dans laquelle chaque agent a_i manipule deux variables q_i^k et p_i^k, où k représente l'itération. Un agent dont $q_i^k = 1$ reste en compétition pour l'itération suivante, si au contraire $q_i^k = 0$ l'agent est non-survivant. Si par contre $p_i^k = 1$, l'agent est survivant. On note $\texttt{compétition}(a_i)$ la liste de ses voisins en compétition pour survivre, cette liste est initialement la liste de ses intentions de fusion $\texttt{accept}(a_i)$. Tout voisin a_j non-survivant ($q_j^{k-1} \neq 1$) quitte la compétition et peut être retiré de cette liste.

Le problème de la synchronisation. Chaque fois qu'un agent a_i, souhaite obtenir une variable q_j^k ou p_j^k d'un agent a_j il doit se synchroniser avec ce dernier afin de lire la variable à jour et non celle de l'itération précédente.

Le modèle agent fonctionne par envois de messages (mémoire distribuée) par opposition au modèle des tableaux noirs. Aussi, la synchronisation d'un agent a_i avec une liste d'agents voisins en compétition pour survivre est effectuée par envoi de messages en deux étapes :

[4] Le terme est choisi, il fait référence à la théorie de l'utilité [Rus95, chap. 16].

1. **Envoi des messages :** l'agent a_i envoie la variable $X \in \{u_i, q_i^k, p_i^k\}$ à tous ses voisins en compétition

   $\forall a_j \in$ `compétition`(a_i) **sendMsg**`(Message (receiver` a_j`)(content X))`

2. **Réception des messages :** l'agent a_i attend de recevoir un message contenant la variable $X \in \{u_i, q_i^k, p_i^k\}$ provenant de tous ses voisins en compétition

   $\forall a_j \in$ `compétition`(a_i) **receiveMsg**`(Message (sender` a_j`)(content X))`

Décimation distribuée2 (a_i)
- $k \leftarrow 1$
- $u_i \leftarrow$ calcul de l'utilité ..... // `ÉvaluerUtilité`
- `compétition`$(a_i) \leftarrow$ `accept`(a_i)
- $\forall a_j \in$ `compétition`(a_i) **sendMsg**`(Message (receiver` a_j`)(content` u_i`))`
- $\forall a_j \in$ `compétition`(a_i) **receiveMsg**`(Message (sender` a_j`)(content` u_j`))`
- $p_i^k \leftarrow 1$ **si** $u_i > \max(u_j : a_j \in$ `accept`$(a_i))$ ..... // `Évaluer_p`

  / Pour les itérations suivantes, pour survivre le sommet doit être maximal parmi les sommets encore en compétition (q=1) */*
- **tant que**(TRUE)
  - $\forall a_j \in$ `compétition`(a_i) **sendMsg**`(Message (receiver` a_j`)(content` p_i^k`))`
  - $\forall a_j \in$ `compétition`(a_i) **receiveMsg**`(Message (sender` a_j`)(content` p_j^k`))`
  - $q_i^k \leftarrow 1$ **si** $(\forall a_j \in$ `compétition`(a_i) $: p_j^k = 0)$ ..... // `Évaluer_q`
  - $k++$
  - **si** $(q_i^{k-1} \neq 1)$ **fin boucle**
  - $\forall a_j \in$ `compétition`(a_i) **sendMsg**`(Message (receiver` a_j`)(content` q_i^{k-1}`))`
  - $\forall a_j \in$ `compétition`(a_i) **receiveMsg**`(Message (sender` a_j`)(content` q_j^{k-1}`))`
  - `compétition`$(a_i) \leftarrow$ `compétition`$(a_i) - \{a_j : q_j^{k-1} \neq 1\}$
  - $p_i^k \leftarrow 1$ **si** $u_i > \max(u_j : (a_j \in$ `compétition`$(a_i))$ ..... // `Évaluer_p`
  - **si** $(p_i^k = 1)$ **fin boucle**

  fin tant que
- **si** $(p_i^k = 1)$ a_i est survivant **fin si**

Algo. 9.5 : *Réécriture condensée de l'algorithme 8.3 p. 157, de décimation distribuée. Les synchronisations sont désormais effectuées par envois de messages. À droite : correspondance avec les différentes étapes du RdP (fig. 9.22, p. 205). Remarque : une légère perturbation aléatoire est ajoutée à u_i pour dégager des maxima locaux sur les zones homogènes.*

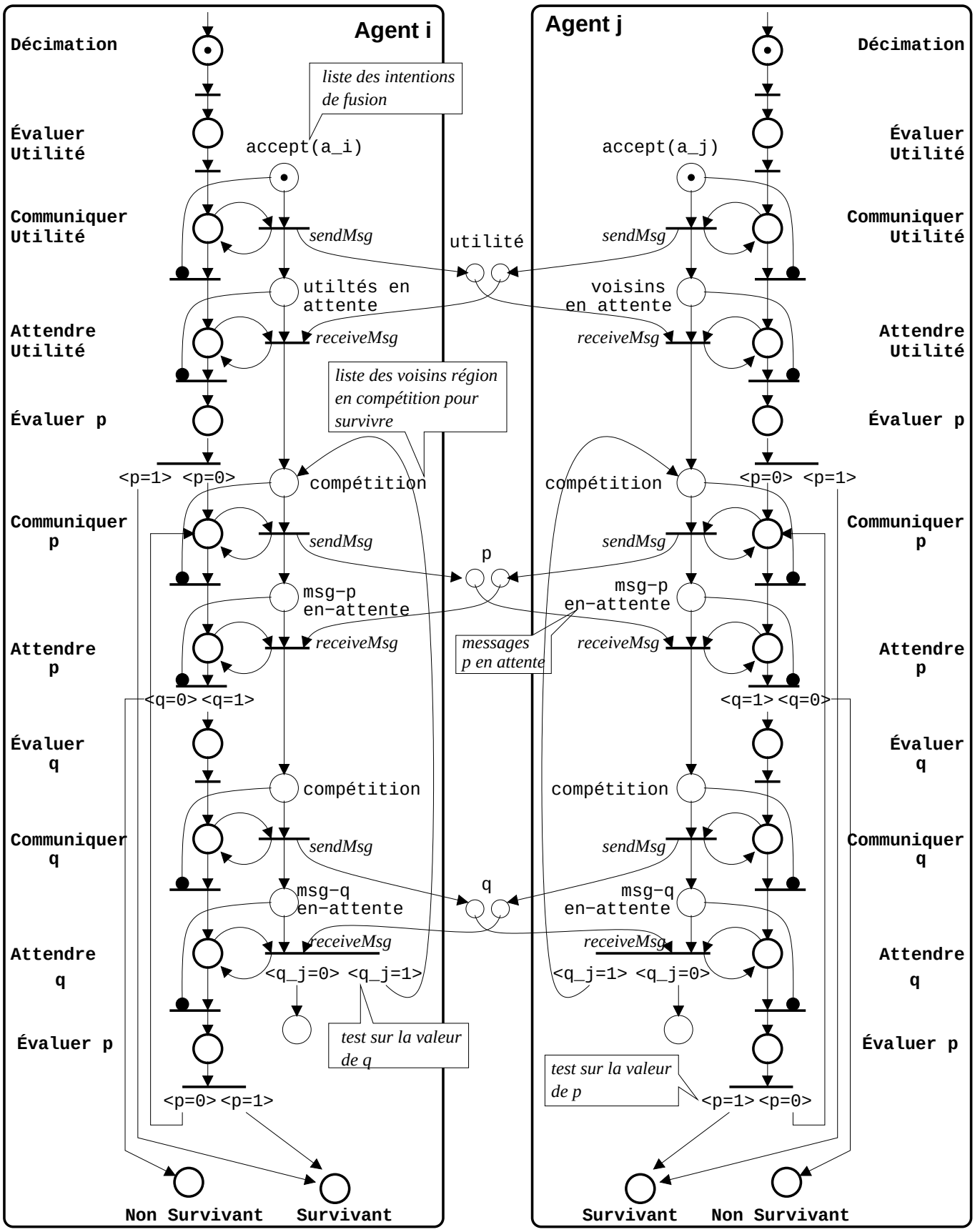


FIG. 9.22 : *RdP du protocole de décimation distribuée. Rappel :* `accept`(a_i) *est une liste contenant les voisins région avec lesquels l'agent* a_i *a l'intention de fusionner.*

9.10 Comportement de rattachement des non-survivants

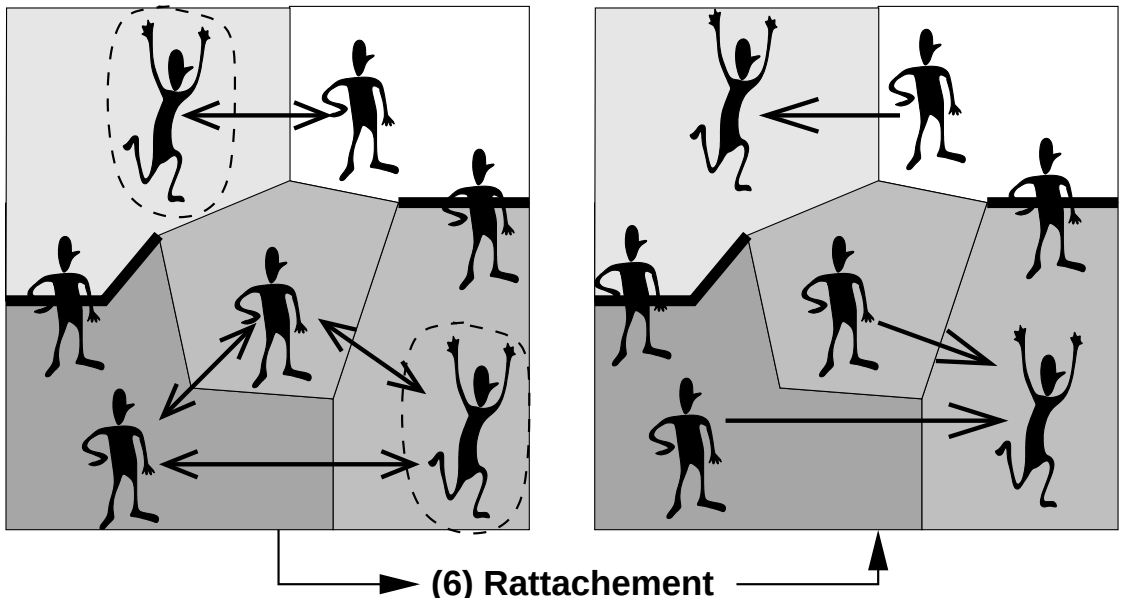


Fig. 9.23 : *Vision schématique du comportement de rattachement.*

Il s'agit désormais de rattacher chaque agent non-survivant à un agent survivant de son voisinage. Si les agents survivants savent que leurs voisins sont tous non-survivants, les non-survivants ne savent pas qui de leurs voisins est survivant ou non.

Pour éclaircir cette situation, tous les agents vont propager des messages dans leur voisinage. Le voisinage dont il est question est considéré dans l'organisation $O^k_{IF}(P^k, A^k_{IF})$, (fig. 9.7, p. 191) structurée par les intentions de fusion contenues dans la liste `accept`(a_i) de chaque agent.

Le comportement des survivants. Les agents survivants vont tâcher de réaliser leurs intentions de fusion. Pour cela, chaque agent survivant va envoyer à tous ses voisins concernés par une de ses intentions de fusion, $\Leftrightarrow \forall a_j \in$`accept`$(a_i)$ (fig. 9.24, p. 208), une proposition de rattachement qui prend la forme d'un message :

```
(Message (performatif ask-one)(sender ?surv)(receiver ?non-surv)
         (reply-with réponse-de ?non-surv à-proposition-de-fusion)
         (content (proposition-de-fusion with ?surv utility ?u))
)
```

Le message contient l'utilité du plan de fusion de l'agent survivant, émetteur de la proposition de rattachement. Cette utilité traduit l'intérêt du plan de fusion du survivant, elle est donc une argumentation à la proposition de rattachement.

Le comportement des non-survivants. Les non-survivants doivent, avant de prendre une décision, être sûrs d'avoir reçu toutes les propositions de rattachement de leurs voisins survivants. Ne sachant pas combien de propositions doivent

leur parvenir, ils vont envoyer à tous leurs voisins un message indiquant qu'il sont `non-survivants`(fig. 9.24, p. 208).

Ainsi, lorsqu'un agent non-survivant a reçu autant de messages qu'il a de voisins, il sait que toutes les propositions de rattachement lui sont parvenues. De tous les messages reçus, seuls les messages de type "proposition de fusion" proviennent de voisins survivants. En conséquence, il peut partitionner son voisinage en `voisins-survivants` et `voisins-non-survivants` (fig. 9.24, p. 208).

Il va donc choisir (le choix est discuté ci-dessous), parmi ses voisins survivants, un voisin auquel se rattacher. Il envoie à ce dernier un message de confirmation `oui`, et a tous les autres voisins survivants un message déclinant la proposition `non` :

```
(Message (performatif tell)(sender  ?non-surv)(receiver ?surv)
         (reply-to réponse-de ?non-surv à-proposition-de-fusion)
         (content (oui attributs-région ?att-non-surv) OR (non))
)
```

Dans le cas où la réponse est positive, il doit aussi transmettre les attributs de la région qu'il représente afin d'être pris en compte dans le prochain niveau de la pyramide.

Remarque sur la convergence de l'algorithme : chaque agent non-survivant possède dans son voisinage au moins un voisin survivant. Ainsi, chaque non-survivant est sûr de recevoir au moins une proposition de rattachement.

Le choix du survivant auquel un non-survivant se rattache se confronte à deux options :

1. **L'intérêt particulier** d'un agent non-survivant est d'être le mieux représenté au prochain niveau. Alors, il choisira parmi les survivants lui ayant fait une proposition celui pour lequel il a le plus de désir (le plus similaire). Un agent peut choisir de se rattacher à un survivant proposant une fusion faiblement utile au détriment d'un autre survivant, dont la proposition exprime une forte utilité.
2. **L'intérêt général** consiste à se rattacher au survivant dont la proposition est la plus utile au détriment de sa représentativité au niveau suivant.

La politique de l'intérêt général favorise les agents représentant de grandes régions, ayant un grand nombre d'intentions de fusion et donc de voisins.

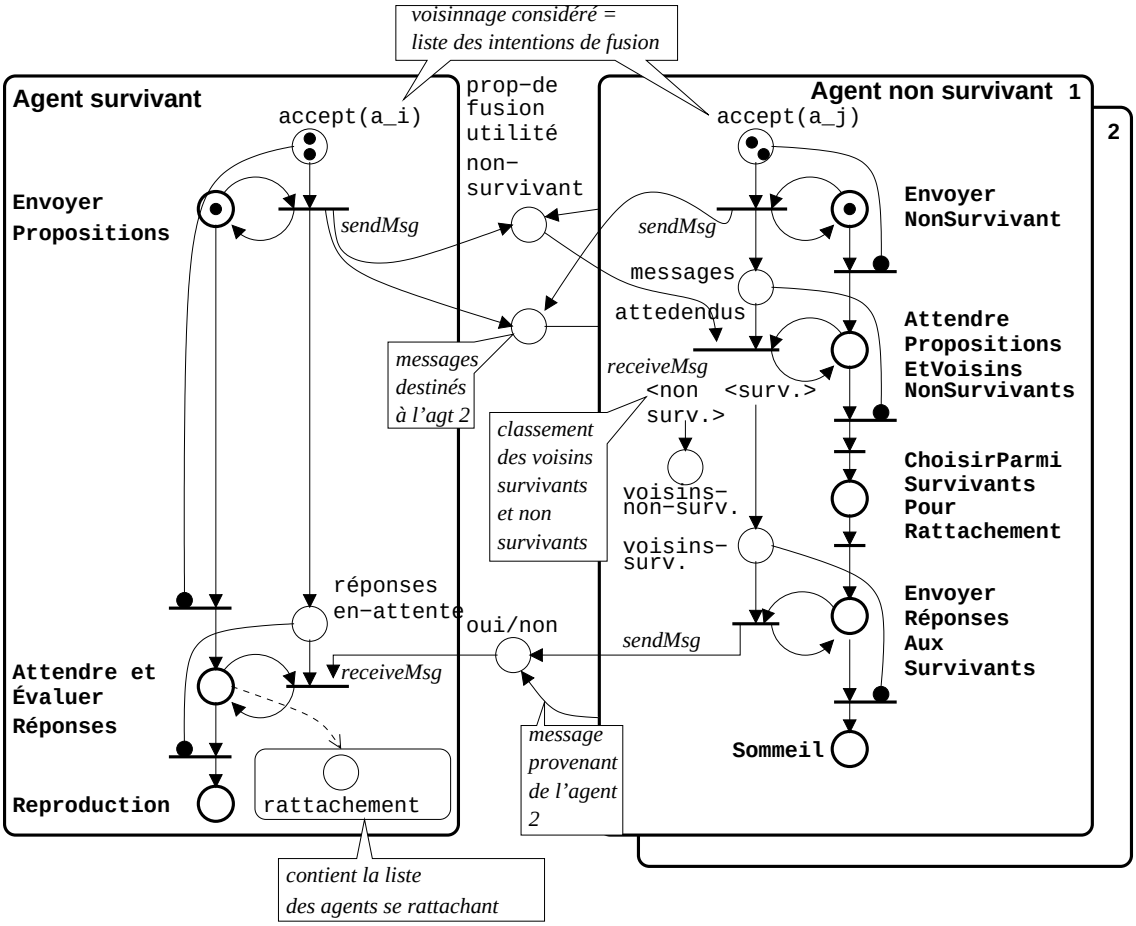


FIG. 9.24 : *Réseau de Pétri du protocole de rattachement. Le RdP modélise un survivant (à gauche) ayant deux voisins non-survivants ; lesquels sont modélisés à droite, ils sont voisins mutuellement et voisins du survivant. Le RdP non visible du deuxième non-survivant est identique à celui du premier (visible).*

9.11 Comportement de reproduction

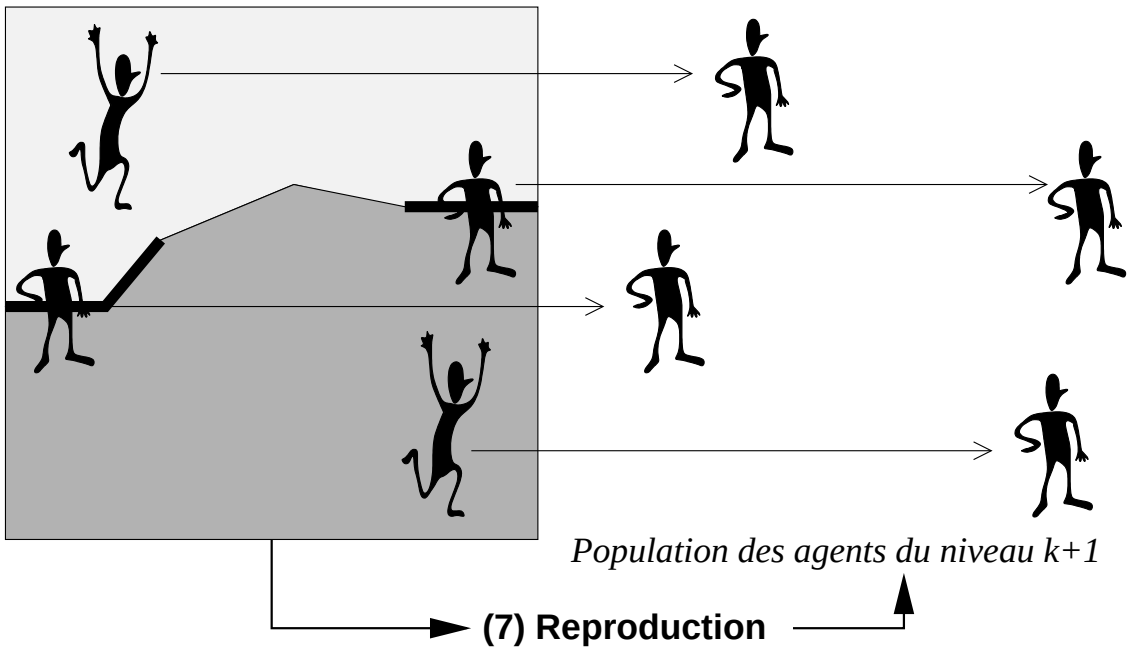


Fig. 9.25 : *Vision schématique du comportement de reproduction.*

Un agent survivant a_i va pouvoir se reproduire, c'est-à-dire créer un nouvel agent, qui sera présent dans le prochain niveau de la pyramide. Afin de respecter la terminologie des structures pyramidales, on appellera `père`(a_i) ce nouvel agent[5] qui devient aussi le père de tous les agents ayant décidé de se rattacher à a_i. La liste de ces agents `rattachement`(a_i) est disponible dans le tableau noir local, elle a été déterminée lors du comportement précédent de rattachement.

Le survivant déclare la création du nouvel agent à l'agent `MonitorAgt` à l'aide d'un message `nouvel-agent` (fig. 8.8, p. 173) & (fig. 9.2, p. 178).

Ce nouvel agent est initialisé avec une liste contenant l'ensemble des attributs de ses enfants c'est-à-dire $\{a_i \cup$ `rattachement`$(a_i)\}$.

La fin de vie d'un agent. Cette étape complétée, l'agent passe dans l'état de `Sommeil` qui provoque (pour tous les agents survivants comme non-survivants) l'envoi d'un message `fin-exécution` à l'agent `MonitorAgt` (fig. 8.8, p. 173) & (fig. 9.2, p. 178).

Naissance d'un nouvel agent. Le nouvel agent va tout d'abord calculer ses attributs $(\mu_s, \sigma_s, h_s, S_s, G_s, cr_s)$ (§9.3 p. 178) en fonction de ceux de tous ces enfants ; puis il va s'endormir en attendant le message `réveil` qui lui parviendra de l'agent `MonitorAgt`. Ce reveil déclenchera le comportement de marquage de territoire (§9.4 p. 179), amorçant les traitements des agents sur le nouveau niveau de la pyramide.

[5]La saga familiale de nos agents est emprunte de schizophrénie.

Chapitre

10

Expérimentations

Nous abordons dans ce chapitre la troisième étape de la description de l'architecture.

1. Description globale et structurelle de l'organisation regroupant les agents (chap. 8).
2. Description locale et fonctionnelle des agents composant le système (chap. 9).
3. Analyse globale et fonctionnelle où l'on vérifie que l'ensemble des interactions locales réalise bien ce pour quoi le système est conçu.

Dans ce chapitre, nous analyserons et comparerons à d'autres approches les résultats des segmentations obtenues sur plusieurs images.

Puis, nous tâcherons de mieux comprendre l'intimité des agents en effectuant une série de mesures pour analyser de manière globale la structure de l'organisation (le nombre d'agents, les relations de voisinage) et les interactions entre agents.

10.1 Introduction

Au cours des chapitres 8 & 9, nous avons d'une part présenté l'organisation du système multi-agents et d'autre part le fonctionnement individuel de chaque agent. Afin de respecter la méthodologie d'analyse des systèmes présenté (§5.2.2 p. 86), il nous reste à analyser **globalement** le comportement du système. Nous proposons une analyse en deux étapes :

1. Analyse brute macroscopique et comparative (§10.2 p. 213) - Lors de cette analyse, nous allons comparer les résultats de segmentation obtenus par la pyramide d'agents avec cinq autres méthodes de segmentation :

1. pyramide irrégulière (méthode de type structurale ou agrégative) ;
2. division/fusion (structurale ou agrégative) ;
3. classification multidimensionnelle Bayésienne (méthode de type classification) ;
4. classification monodimensionnelle non supervisée par la méthode de Fisher (classification) ;
5. détection de contour par l'opérateur de Shen suivie d'une fermeture de contour (approche contour) ;

Ces cinq méthodes appartiennent à trois grandes familles d'approches : (i) structurale ou agrégative ; (ii) classification ; (iii) approche contour.

Les six méthodes vont être appliquées sur cinq images différentes :

1. deux images de scanner du sein, permet d'évaluer les approches sur des images médicales bruitées, dont les transitions ne sont pas franches et qui ont des régions distinctes assez similaires, et parfois en contact ;
2. l'image "muscle" du GDR PRC-ISIS, donne l'occasion de comparer de l'approche agent sur une image médicale bien connue ;
3. "savoise" du GDR PRC-ISIS permet une comparaison sur une image connue, contenant des formes géométriques artificielles ;
4. une image comportant deux dégradés côte à côte, qui ne peut être correctement segmentée qu'avec une adaptation locale des critères de discrimination entre régions.

L'analyse brute des résultats va être effectuée en deux étapes :

1. nous allons d'abord comparer les résultats obtenus par les six méthodes, chacune étant paramétrée de manière optimale ;
2. nous allons ensuite analyser la *robustesse* des méthodes ayant franchi avec succès l'étape précédente.

2. Analyse détaillée : dans l'intimité des agents (§10.3 p. 233) - Dans cette deuxième partie des expérimentations, nous allons effectuer un ensemble de mesures sur la population des agents afin d'analyser le comportement global de l'organisation. Nous essayerons, à la manière des sondages sur une population humaine, d'extraire de ces mesures des tendances globales du SMA.

10.2 Évaluation macroscopique et comparative

Nous allons tout d'abord présenter les différentes images d'expérimentation (§10.2.1 p. 213), puis nous présenterons les cinq autres méthodes (§10.2.2 p. 216) avec lesquelles nous comparerons les résultats obtenus avec la pyramide d'agents.

La comparaison des résultats obtenus par la pyramide d'agents et les cinq autres méthodes se fera en deux étapes :

1. Capacité d'une méthode à segmenter un type d'image. Après avoir configuré les six méthodes, manuellement, de manière à obtenir le meilleur résultat possible, on évalue et on compare la capacité de chacune à segmenter les différentes images. Cette étape (§10.2.3 p. 218) nous permettra de mettre en relief les avantages, inconvénients et différences entre approches agrégatives (dont fait partie la pyramide d'agents) et approches par classification.

2. Robustesse d'une méthode selon deux types de variation (fig. 10.1).

1. Variation sur le paramétrage de l'opérateur sur une même image (§10.2.4 p. 230). Cette variation autour du paramétrage optimal correspond à une dégradation de la qualité de l'information *a priori* fournie.
2. Modification de l'image à traiter à paramétrage constant (§10.2.5 p. 232). On souhaite savoir si une méthode qui segmente correctement une image parvient à traiter correctement une image similaire sans intervention sur le paramétrage (seuils, etc.). Une approche robuste sur ce point permet de limiter l'intervention humaine lors de traitements de séquences d'images similaires (par exemple : un scanner de sein sur des patientes différentes).

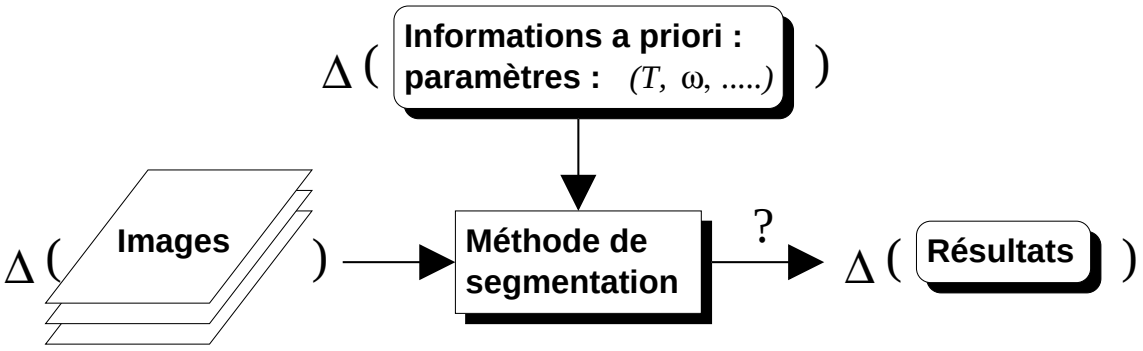


Fig. 10.1 : *Les variations possibles autour de l'optimum sur les entrées d'un opérateur de segmentation ont un impact plus ou moins important sur les résultats. Une faible variation des résultats dénote une méthode robuste.*

10.2.1 Images d'expérimentation

Nous proposons une évaluation sur cinq images : trois de ces images sont réelles et présentent des régions texturées et bruitées ; les deux autres sont des images de synthèse homogènes.

Scanner du sein 1 (fig. 10.2) permet de comparer les méthodes dans un cadre réel d'imagerie médicale. Cette image présente le scanner d'un sein délimité en bas par la peau, en haut par les os des côtes et le muscle pectoral. Le sein est composé de tissus graisseux, dans lequel on trouve le tissu glandulaire vascularisé qui est plus clair au scanner (cf. légende de l'image 6.2 page 108) à cause du produit de contraste.

Une segmentation est de bonne qualité si elle sépare correctement le tissu glandulaire du muscle et du tissu graisseux (voir chapitre 6 (fig. 6.2, p. 108) pour plus de détails).

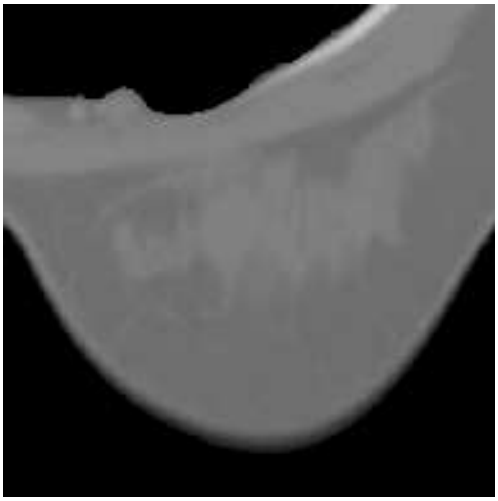


Fig. 10.2 : *Scanner du sein 1.*

Scanner du sein 2 (fig. 10.3) est l'image d'une autre patiente, acquise dans des conditions similaires au scanner du sein 1. Elle diffère cependant car le tissu glandulaire (TG) n'est pas en contact avec le muscle, ce qui rend cette image plus facile à traiter. Il y a une zone faiblement vascularisée autour de TG qui doit être signalée comme telle au clinicien, qui choisira de l'intégrer ou non à la zone d'intérêt (dans laquelle peuvent se développer des tumeurs).

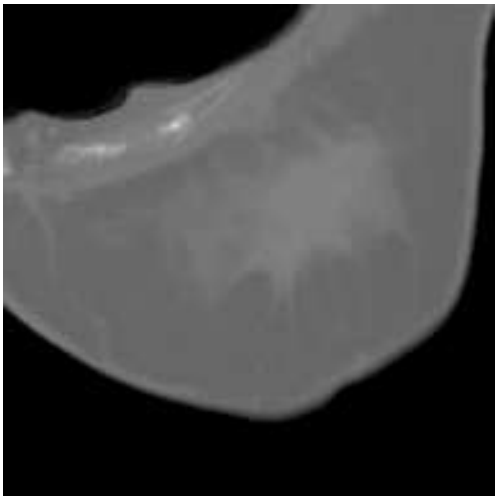


Fig. 10.3 : *Scanner du sein 2.*

Fibres musculaires (extrait de) (fig. 10.4). Cette image correspond à une fenêtre de 192×192 de l'image "muscle" de la base d'image du GDR 134. Cette image, obtenue au microscope présente deux types de fibres musculaires : des fibres foncées et homogènes ; des fibres claires et texturées. Les fibres foncées se démarquent nettement du fond (clair) et des fibres claires. Le contraste entre les fibres claires et le fond est moins marqué. Les frontières entre fibres du même type ne peuvent être établies avec certitude sans l'apport d'informations *a priori* supplémentaires sur la forme ronde des fibres.

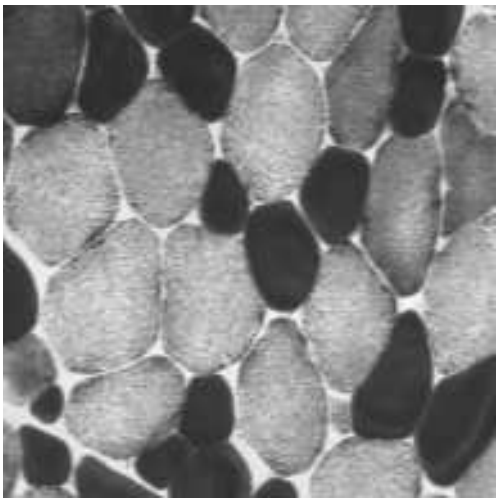


Fig. 10.4 : *Fibres musculaires (extrait de "muscle").*

Savoise (adaptée) (fig. 10.5) . Cette image 192×192 est extrait de l'image "savoise" de la base d'image du GDR 134. Un cadre a été rajouté autour du dégradé permettant de mieux l'isoler du fond. Cette image permet d'évaluer la qualité d'une méthode vis-à-vis de formes géométriques différentes : transitions horizontales, verticales, diagonales et courbes.

Fig. 10.5 : *Savoise (adaptée).*

Dégradés (fig. 10.6) est une image de synthèse présentant deux dégradés variant de 0 à 255. La segmentation idéale est constituée de deux régions, une pour chaque dégradé. Ce type d'image permet d'illustrer l'adaptation locale (seuils et facteurs de pondération) des pyramides ainsi que la notion de dérive des statistiques des approches agrégatives.

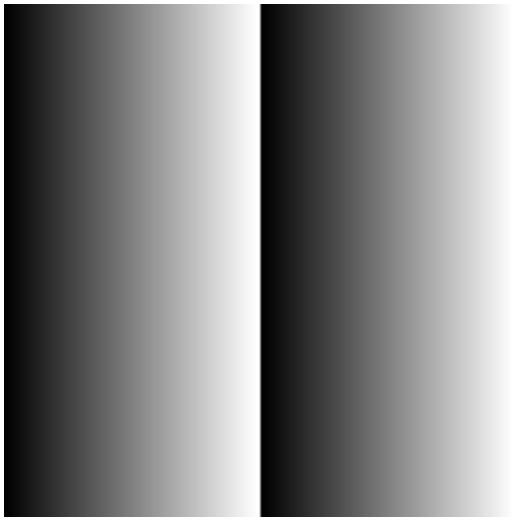


Fig. 10.6 : *Dégradés.*

10.2.2 Méthodes évaluées

Six méthodes ont été appliquées aux images définies précédemment. Nous présentons trois méthodes agrégatives (§2.4 p. 26), deux méthodes de type classification et une méthode contour.

Remarque : l'outil Pandore [Pan01] développé au Greyc de Caen a été choisi pour la mise en œuvre des cinq autres approches figurant dans le comparatif.

Les méthodes agrégatives :

1. La pyramide d'agents qui est l'approche développée dans cette thèse.
Paramétrage : les deux paramètres essentiels sont les deux seuils définissant l'intervalle d'incertitude :

$$\Big[\textit{Désir nécessaire, Désir suffisant}\Big] \Leftrightarrow \Big[D_n, D_s\Big]$$

Ces deux valeurs comprises entre 0 et 1 se calculent directement à partir des valeurs de niveau de gris selon la formule :

$$D_* = 1 - \frac{\text{niveau de gris}}{255}$$

1. Si l'utilisateur veut éviter la fusion de régions éloignées de plus de 20 niveaux de gris, il fixera Dn à $0.92157 = 1 - \frac{20}{255}$.

2. Si l'utilisateur souhaite que les régions éloignées de moins de 5 niveaux de gris puissent fusionner, il fixera Ds à $0.9804 = 1 - \frac{5}{255}$.

La mesure d'éloignement est calculée par la fonction EVAL_DÉSIR() (équation 9.1 p. 186) qui est une somme pondérée de plusieurs paramètres normalisés sur l'intervalle $[0, 1]$. Tous ces paramètres, à l'exception de la longueur de frontière et la corrélation d'histogrammes, sont comparables car ramenés au domaine de variation des niveaux de gris $[0, 255]$, normalisé sur $[0, 1]$. Une réflexion doit être menée sur la sémantique d'une fusion entre la longueur de frontière et la corrélation avec les autres paramètres (ces deux paramètres ayant des unités de mesure différentes). Ces deux paramètres sont désactivés dans le cadre des expérimentations. De plus, lorsque les pondérations des paramètres sont fixées manuellement (pas d'utilisation de l'ACP) l'expérience nous a montré que pour ces images seule la différence de moyenne est utile.

Afin de faciliter la lecture et la comparaison avec les autres méthodes, les deux seuils sont donnés sous leur forme non normalisée c'est-à-dire :

$$\big[D_n \in [0, 255], D_s \in [0, 255]\big]$$

2. La pyramide irrégulière adaptative présentée au chapitre 3. La méthode ne prend qu'un seul paramètre : T. Si la valeur absolue de la différence des niveaux de gris moyens de deux sommets est inférieure à T, une arête reliant les deux sommets est créée dans le graphe de similarité.

3. Une approche de type division/fusion (§2.4.1 p. 27) & (§2.4.2 p. 27). Elle utilise un seul paramètre σ^2, qui est le seuil portant sur la variance des régions. Si la variance d'une région est supérieure au seuil, cette région est découpée suivant un arbre quaternaire (phase descendante). Si la variance de l'union de deux régions voisines dans l'arbre est inférieure au seuil, elles sont fusionnées.

Les méthodes de type classification :

4. Classification monodimensionnelle non-supervisée (§2.2.1 p. 19) de type Fisher utilisant un paramètre n, qui est le nombre de classes souhaitées.

5.Classification multidimensionnelle Bayésienne (§2.2.2 p. 19). Les attributs sur lesquels porte la classification sont : la moyenne et la variance pour chaque site sur un voisinage 3×3. L'utilisateur doit fournir pour chaque classe désirée un ensemble d'échantillons d'apprentissage.

6. L'approche contour proposée est constituée d'un détecteur de type Shen et d'une méthode de fermeture de contour (§A.3 p. 255). Elle utilise deux paramètres, le premier β pour la détection de contour de type Shen est un compromis entre lissage (non sensibilité au bruit) et qualité de la détection du contour (bonne réponse, non multiple et bien localisée du contour). Le deuxième paramètre d est la distance de recherche d'un point candidat à la fermeture du contour.

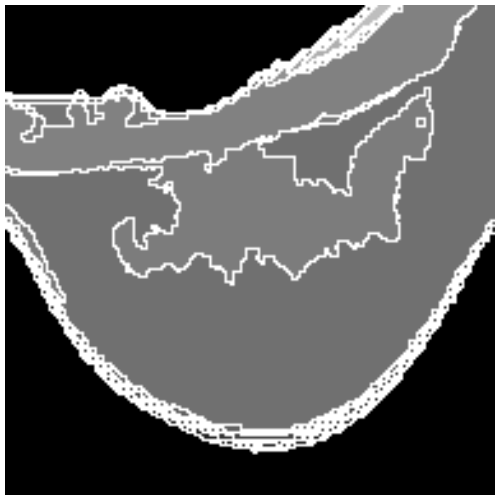
10.2.3 Capacité d'une approche à segmenter une image

Chacune des méthodes a été paramétrée avec soin pour chaque image afin d'obtenir le meilleur résultat (selon une évaluation évoquée ci-dessous) possible. Ainsi, nous allons, lors de cette section, analyser la qualité d'une approche pour un type d'image.

Les critères d'évaluation utilisés sont de deux natures :

1. Critère objectif : on utilise une différence inter-image entre une segmentation idéale faite manuellement et la segmentation obtenue par une des six méthodes.
2. Critères subjectifs : le résultat d'une segmentation est généralement ensuite soumis à une interprétation humaine (médecin), ou automatique (système de vision haut-niveau). Or, comme c'est souvent le cas (par ex. pour "scanners du sein 1 et 2"), les critères objectifs ne coïncident pas avec une interprétation humaine. Les résultats de segmentation maximisant les critères objectifs conduisent à des résultats sur-segmentés inexploitables par un médecin. Aussi, une évaluation qualitative relative au domaine d'application est nécessaire.

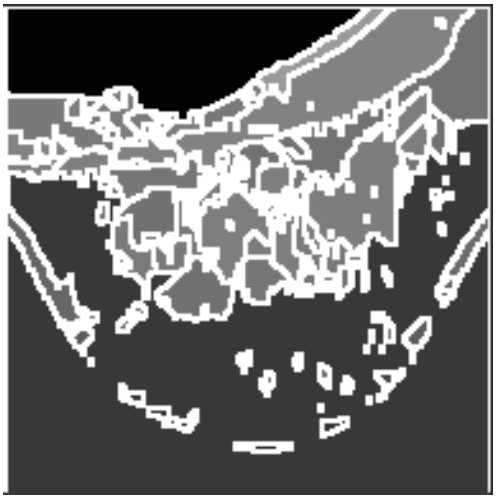
Scanner du sein 1 : résultats



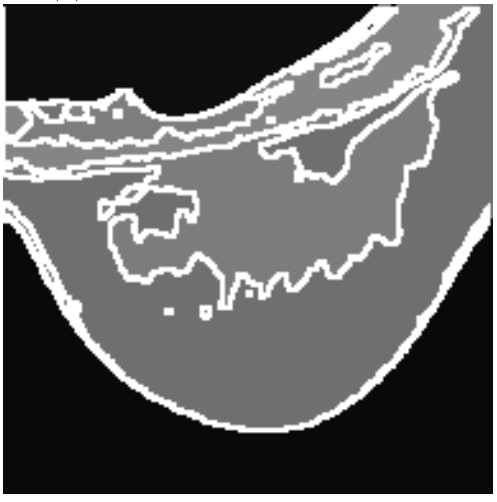
(a) Pyramide d'agents, $[D_n = 10, D_s = 2]$

(b) Pyramide irrégulière adaptative, $T = 4$

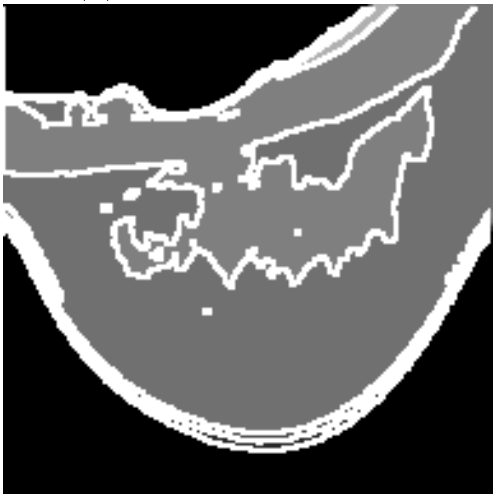
(c) Division et fusion, $\sigma^2 = 9$



(d) Shen, $\beta = 0.9, d = 50$



(e) Bayes



(f) Fisher, $n = 6$

Fig. 10.7 : *Segmentation par différentes méthodes de l'image "scanner du sein 1". Remarque pour les méthodes autres que la pyramide d'agents les frontières ont deux pixels d'épaisseur.*

Scanner du sein 1 : analyse et commentaires

Les résultats et les paramètres utilisés pour cette image sont présentés (fig. 10.7, p. 219).

Les approches par classification Bayésienne ou Fisher ((e), (f), fig. 10.7) tendent à sous-segmenter (fig. 10.7), en fusionnant le muscle et TG pour Fisher, et une partie du muscle avec TG pour la classification Bayésienne. C'est précisément la difficulté que pose cette image, TG et le muscle ayant presque les mêmes caractéristiques photométriques (tab. 6.1).

La présence d'un point de contact à transition faible (voir profil de ligne à travers la transition 6.3 page 109) entre régions, dont les caractéristiques photométriques sont similaires, contraint les approches par classification à une sous-segmentation ou à une sur-segmentation.

Les approches agrégatives (pyramide d'agents, pyramide irrégulière adaptative et division fusion) ((a), (b), (c), fig. 10.7, p. 219) obtiennent de bons résultats sur ces images car elles réussissent à isoler TG du muscle et des tissus graisseux sans sur-segmentation excessive. La division/fusion est la moins bonne des trois : elle sur-segmente sensiblement les tissus graisseux.

Cependant, la qualité de ces résultats doit être relativisée par la grande sensibilité des approches "pyramide irrégulière adaptative" et "division/fusion" à la précision de l'information *a priori* fournie (seuil) (§10.2.4 p. 230).

Discussion sur l'intérêt des approches agrégatives pour ce type d'image. Les approches agrégatives, dont la pyramide d'agents relève, montrent leur intérêt intrinsèque grâce à leur caractère **incrémental**. En effet, les approches agrégatives construisent incrémentalement une primitive en fusionnant d'abord avec les voisins adjacents les plus similaires. Comme le montrent les niveaux 1 à 5 de la pyramide d'agents (fig. 10.8, p. 221), les petites régions de TG adjacentes à des régions du muscle ont plutôt tendance à fusionner avec d'autres régions plus proches du centre de TG qu'avec des régions du muscle. Ce scénario induit, fusion après fusion, une convergence des statistiques vers les vraies valeurs des régions TG et muscle (tab. 6.1). Permettant ainsi de les séparer si les seuils sont suffisamment bas.

Cette **évolution des statistiques des régions**, caractéristique des approches incrémentales, est dans ce cas opportune et fait apparaître des transitions artificielles favorisant une bonne distinction du muscle et de TG.

La prise en compte précoce de la proximité spatiale, puis des ressemblances photométriques dans une dynamique incrémentale, est un point essentiel de distinction avec les méthodes de classification qui exploitent la proximité spatiale dans un second temps par regroupement des labels connexes.

Cet aspect des approches agrégatives est un des points qui a motivé l'inscription de notre approche dans le cadre de ces méthodes.

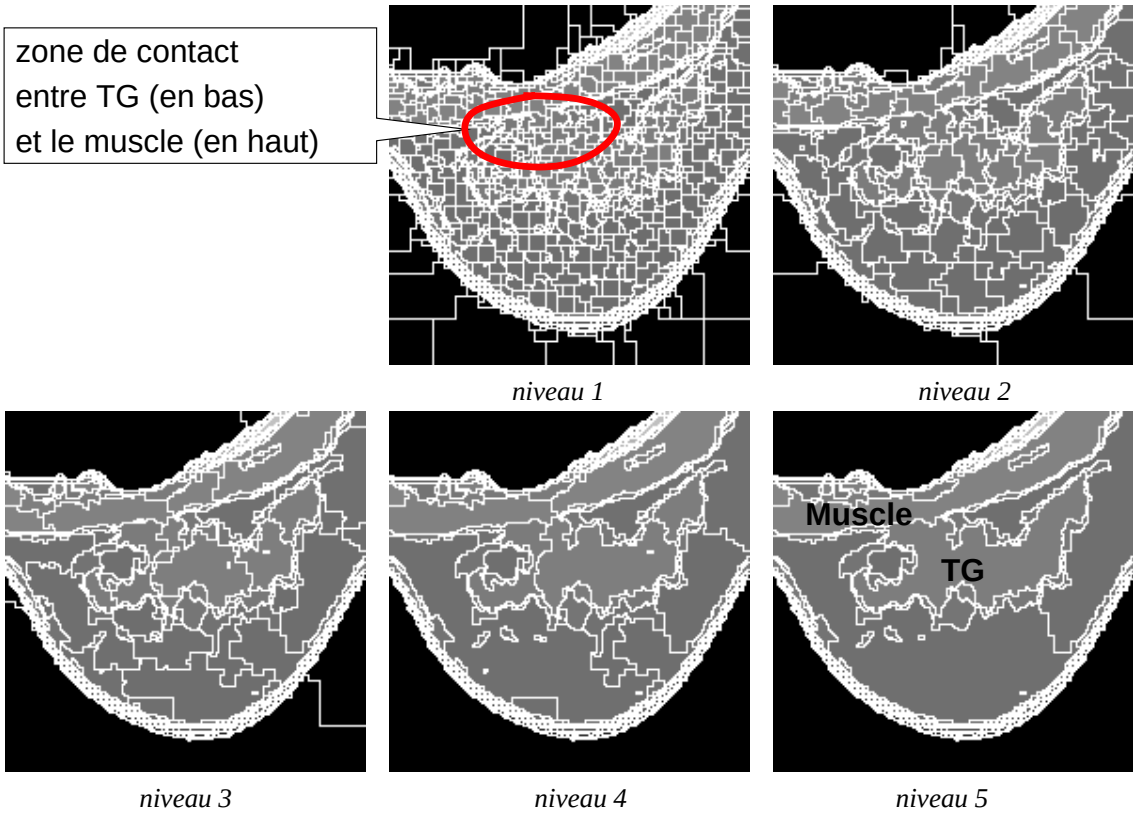


Fig. 10.8 : *Les meilleures fusions en premier permettent aux approches par agrégation (ici la pyramide d'agents) d'obtenir une segmentation correcte de régions faiblement différentes et en contact.*

Limites des approches agrégatives. Le scénario évoqué ci-dessus est optimiste, un seuil un peu trop haut provoque une fusion entre régions du muscle et de TG. Cette fusion entraîne une évolution irrémédiable[1] des statistiques (dans ce cas inopportune) vers la moyenne des deux régions, et donc une sous-segmentation. À l'opposé, un seuil trop bas implique une sur-segmentation. La marge de manoeuvre entre les deux étant extrêmement faible (2 niveaux de gris), la pyramide irrégulière adaptative même pourvue d'une adaptation locale des seuils, ainsi que l'approche division/fusion ne peuvent être qualifiées de robustes pour ce type de problème. Elles sont en effet trop sensibles à la précision de l'information *a priori* fournie.

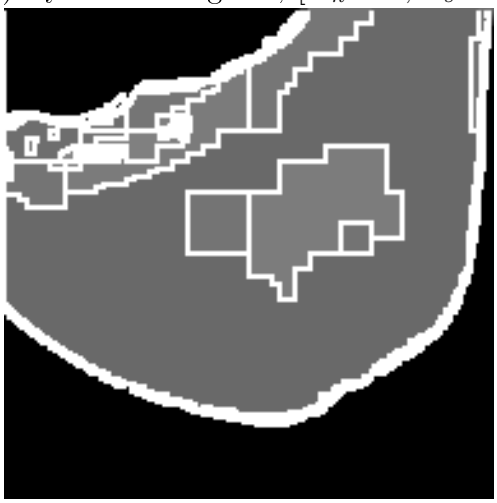
L'approche contour (Shen) ne peut fonctionner, étant donné l'aspect bruité des régions TG et muscle.

[1] Cet aspect irrémédiable implique la notion de prudence souvent évoquée lors du chapitre 9 ; à moins que l'on utilise ultérieurement une division descendante.

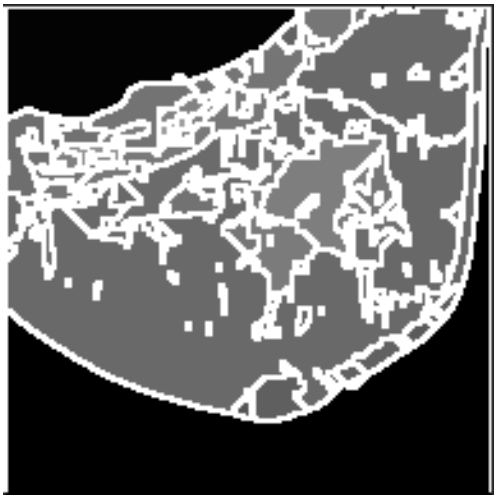
Scanner du sein 2 : résultats

(a) Pyramide d'agents, $[D_n = 8, D_s = 4]$

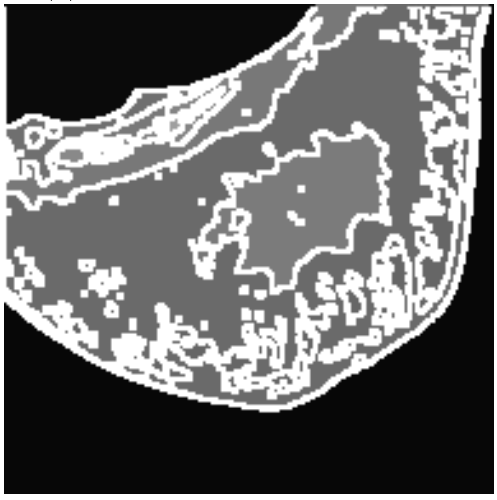
(b) Pyramide irrégulière adaptative, $T = 6$



(c) Division et fusion, $\sigma^2 = 23$



(d) Shen, $\beta = 0.9, d = 50$



(e) Bayes

(f) Fisher, $n = 6$

Fig. 10.9 : *Segmentation par différentes méthodes de l'image "scanner du sein 2".*

Scanner du sein 2 : analyse et commentaires

Les résultats et les paramètres utilisés pour cette image sont présentés à la figure (fig. 10.9, p. 222). Cette image est moins problématique que l'image du sein 1, TG et le muscle n'ayant pas de points de contact. Cependant, la zone de tissus graisseux est moins homogène, et la transition avec TG présente une région intermédiaire que nous souhaitons distinguer des tissus graisseux et du noyau de TG.

Les approches par classification. La classification Bayésienne tend à sur-segmenter le tissu graisseux et à sous-segmenter TG. Fisher donne de bons résultats sur le tissu graisseux et TG, mais elle sur-segmente le muscle.

Les approches agrégatives. La pyramide irrégulière adaptative et la pyramide d'agents donnent des résultats satisfaisants et très similaires. L'approche division/fusion donne un résultat sur-segmenté au niveau du muscle, avec en plus un "effet de pavage" important qui nuit à la bonne localisation des frontières des régions. Cet effet est dû à un seuil de variance élevé qui est pourtant le minimum, évitant la fusion des régions muscle et TG.

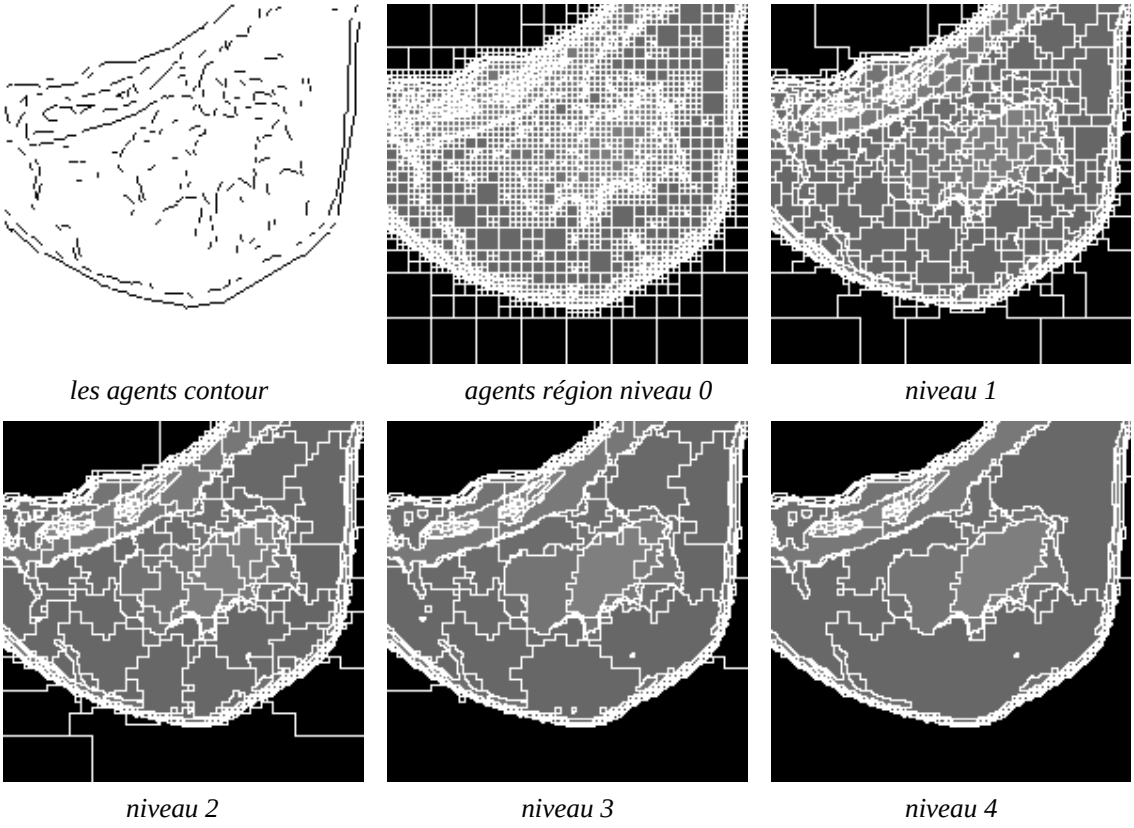
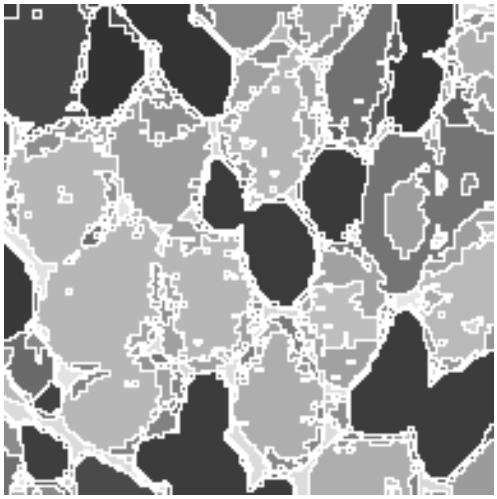


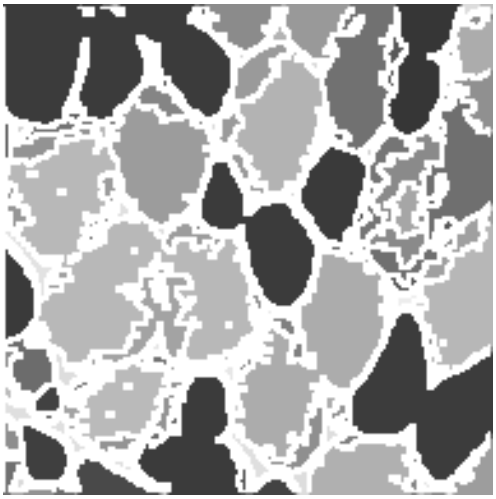
Fig. 10.10 : *Les cinq premiers niveaux de la pyramide d'agents, ainsi que les segments de contour, représentés par les agents contour.*

L'approche contour (Shen) ne peut fonctionner étant donné l'aspect bruité des régions TG, muscle et tissus graisseux.

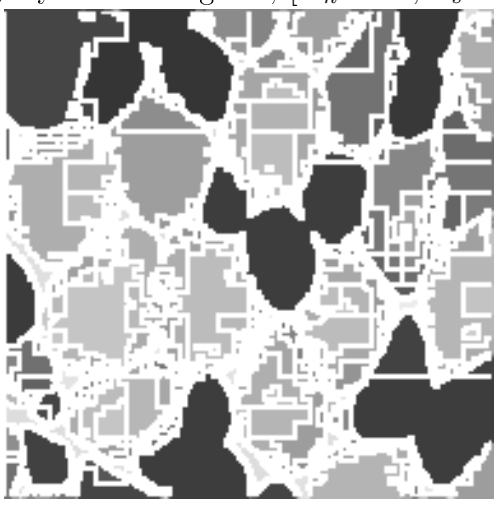
Fibres musculaires (extrait de) : résultats



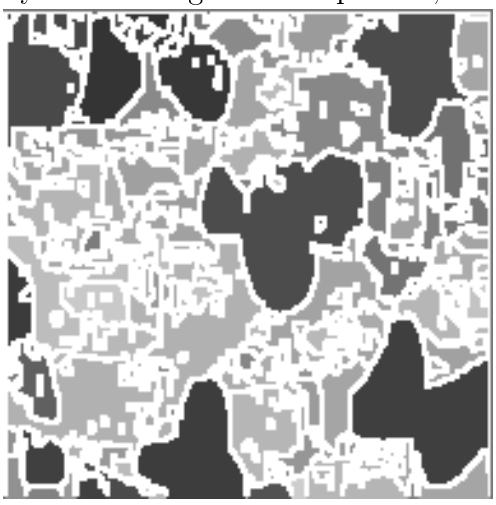
(a) Pyramide d'agents, $[D_n = 25, D_s = 5]$



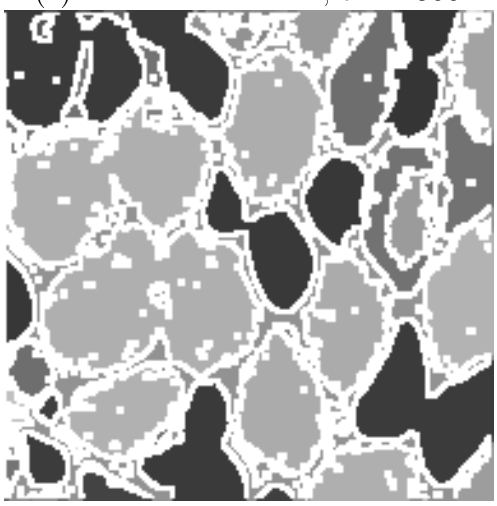
(b) Pyramide irrégulière adaptative, $T = 20$



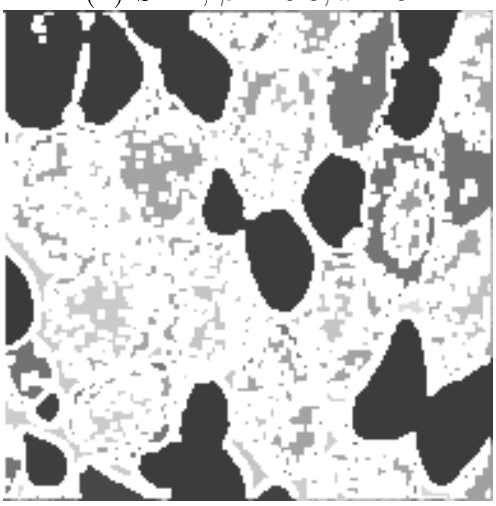
(c) Division et fusion, $\sigma^2 = 300$



(d) Shen, $\beta = 0.3, d = 5$



(e) Bayes



(f) Fisher, $n = 58$

Fig. 10.11 : *Segmentation par différentes méthodes de l'image extraite de "fibre musculaire".*

Fibres musculaires (extrait de) : analyse et commentaires

Les résultats et les paramètres utilisés pour cette image sont présentés à la figure (fig. 10.11, p. 224). L'évaluation permet de comparer notre approche sur une image connue. Une analyse des résultats des approches classiques, appliquées à cette image, est disponible dans [Coc95]. La pyramide d'agents sans réglage particulier y montre des résultats comparables à la pyramide irrégulière adaptative.

La coopération agents région/ agents contour permet d'éviter une fusion des fibres noires en haut à gauche. Cependant, elle est d'une faible utilité pour séparer les fibres grises texturées. En effet, la procédure d'initialisation des agents contour n'a pas été paramétrée pour ce type d'image. Un paramétrage adapté (seuils plus élevés) au bruit présent dans les fibres grises devrait permettre l'élimination d'une partie des agents contour (fig. 10.12) présents dans les fibres grises, et donc une meilleure exploitation de l'information contour.

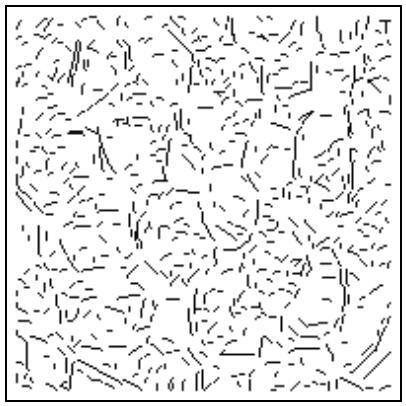
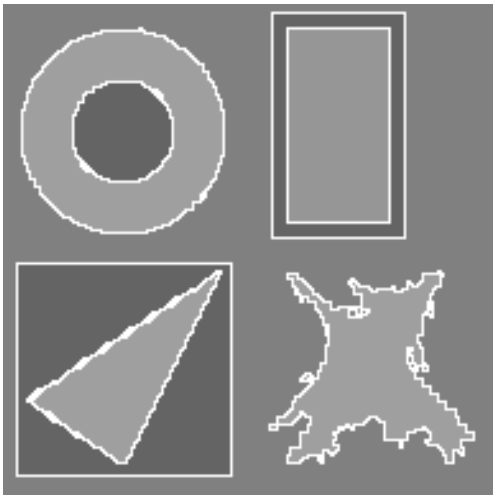


Fig. 10.12 : *Les agents contour dans l'image "muscle".*

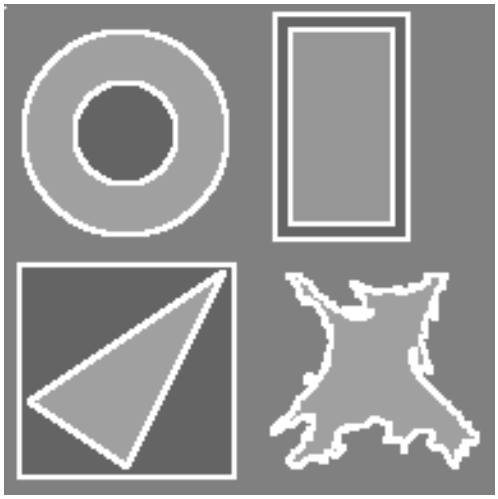
L'agent comme support de l'information issue du modèle. L'enrichissement conceptuel que fournit la notion d'agent par rapport à un simple sommet de graphe, permet d'envisager la mise en œuvre de procédures plus intelligentes adaptées à l'image traitée. Par exemple, dans le cas de l'image "fibre musculaire", l'agent peut tenir à jour des attributs géométriques (comme la compacité) en plus des attributs photométriques. Un agent dont les attributs photométriques correspondent à une fibre (foncée ou clair) refuse toute fusion, provoquant une forte diminution de cette compacité.

Cet exemple traduit l'hypothèse *a priori* de compacité et circularité des fibres. L'agent fournit donc un cadre favorisant la représentation et la mise en œuvre d'informations issues du modèle.

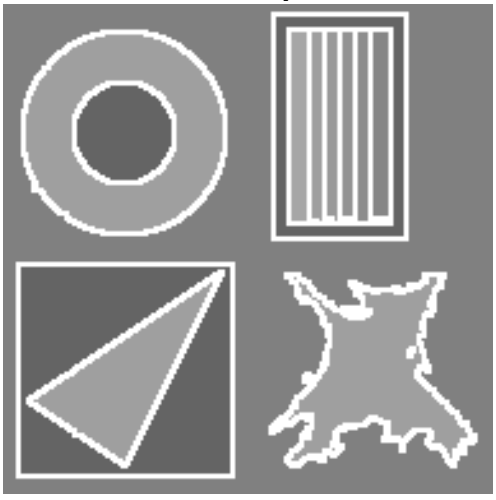
Savoise (adaptée) : résultats



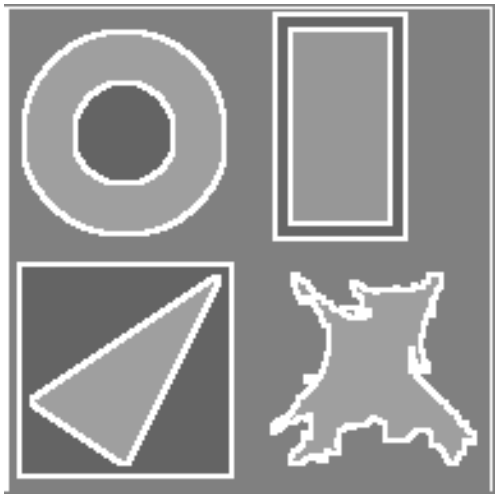
(a) Pyramide d'agents, $[D_n = 25, D_s = 10]$



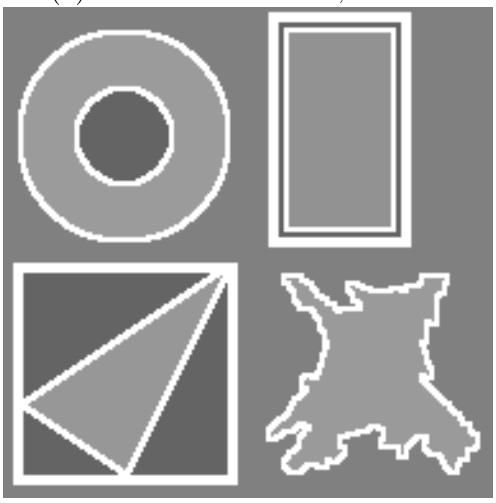
(b) Pyramide irrégulière adaptative, $T = 25$



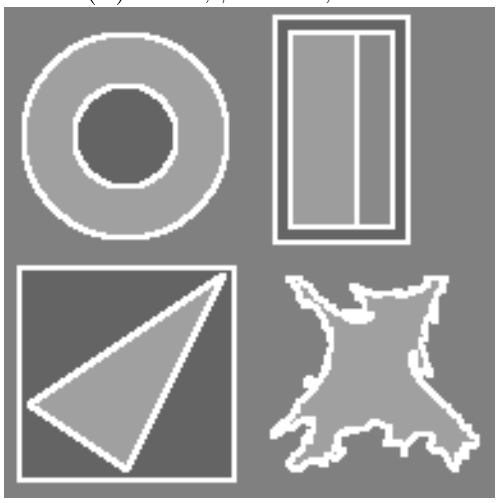
(c) Division et fusion, $\sigma^2 = 6$



(d) Shen, $\beta = 0.8, d = 1$



(e) Bayes



(f) Fisher, $n = 3$

FIG. 10.13 : *Segmentation par différentes méthodes de l'image "Savoise adaptée".*

Savoise (adaptée) : analyse et commentaires

Les résultats et les paramètres utilisés pour cette image sont présentés (fig. 10.13, p. 226). Les résultats de la pyramide d'agents sont comparables à ceux obtenus par la pyramide irrégulière adaptative. Les structures géométriques sont correctement segmentées sans apparition notable d'effets de pavage. La figure 10.14 montre la construction des premiers niveaux de la pyramide sur cette image.

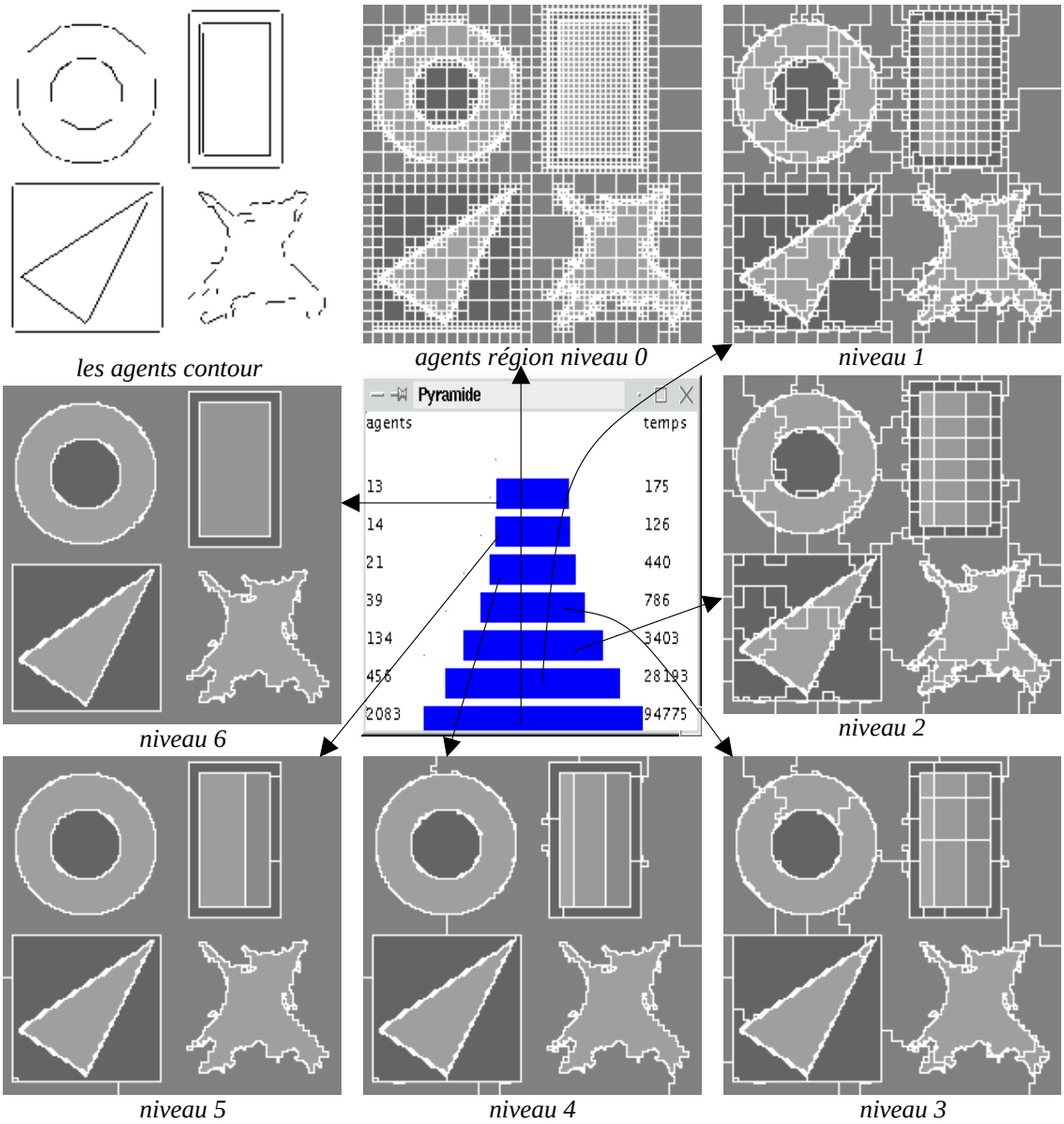
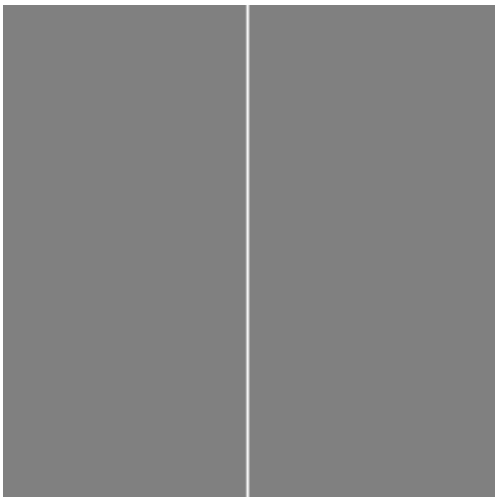


Fig. 10.14 : *Les niveaux de la pyramide d'agents, ainsi que les segments de contour, représentés par les agents contour. Au centre une capture d'écran de l'interface graphique.*

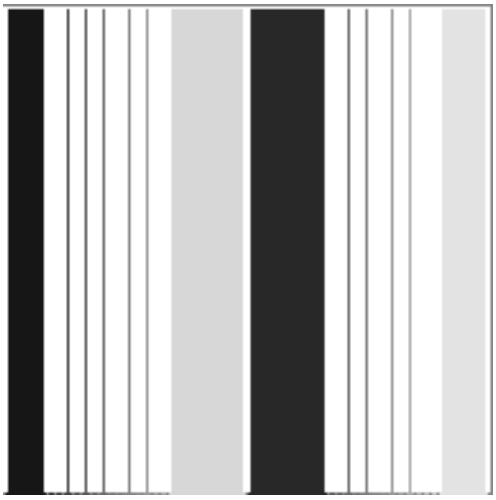
Dégradés : résultats



(a) Pyramide d'agents, $[D_n = 20, D_s = 2]$

(b) Pyramide irrégulière adaptative, $T = 80$

(c) Division et fusion, $\sigma^2 = 2000$



(d) Shen, $\beta = 0.8, d = 1$

(e) Bayes

(f) Fisher, $n = 2$

Fig. 10.15 : *Segmentation par différentes méthodes de l'image "dégradés".*

Dégradés : analyse et commentaires

Les résultats et les paramètres utilisés pour cette image sont présentés (fig. 10.15, p. 228). Toutes les approches classiques (hormis l'approche contour) donnent des résultats comparables. Les segmentations montrent au moins quatre bandes verticales, nombre minimum nécessaire pour éviter la fusion entre les deux dégradés.

Adaptation locale des pondérations par ACP. Pour cette image seulement, nous avons intégré le spécialiste d'adaptation locale des pondérations par ACP (§9.6.3 p. 187) portant sur les affinités (attributs) lors de l'évaluation du désir (similarité).

En effet, le code de ce spécialiste n'est pas encore stabilisé. L'objectif est simplement d'illustrer cette possibilité sur une image de synthèse bien maîtrisée.

Comme évoqué (§9.6.3 p. 187), l'ACP modifie les pondérations apportées aux affinités (attributs). Le contraste à la frontière (après l'ACP) se voit affecté un facteur de pondération plus important que le facteur de pondération affecté à la différence (en valeur absolue) des moyennes.

Seule cette adaptation locale permet d'obtenir les résultats escomptés, à savoir l'obtention de deux régions finales correspondant chacune à un dégradé.

La figure 10.16 page 230 illustre l'adaptation locale du désir nécessaire. Cette adaptation favorise les fusions avec les voisins les plus ressemblants, provoquant la création de bandes verticales jusqu'au niveau 5. Du niveau 0 à 5, les adaptations locales des facteurs de pondération sont très faibles puisque le contraste à la frontière et les différences de moyennes (en valeur absolue) ont tous deux un faible pouvoir discriminant.

On constate à ce niveau l'apparition de fausses transitions entre bandes verticales, qui étaient absentes de l'image source.

À partir du niveau 5, les régions fusionnent horizontalement indiquant que le seuil de désir nécessaire local se relâche pour atteindre la valeur globale.

C'est au niveau 8 que l'adaptation locale des pondérations joue pleinement : les bandes 1 et 2 ont la même moyenne, mais leur contraste à la frontière est grand. Ce dernier attribut (plus discriminant) se voit allouer un facteur de pondération bien supérieur au facteur de pondération alloué à la différence des moyennes évitant ainsi la fusion des deux bandes.

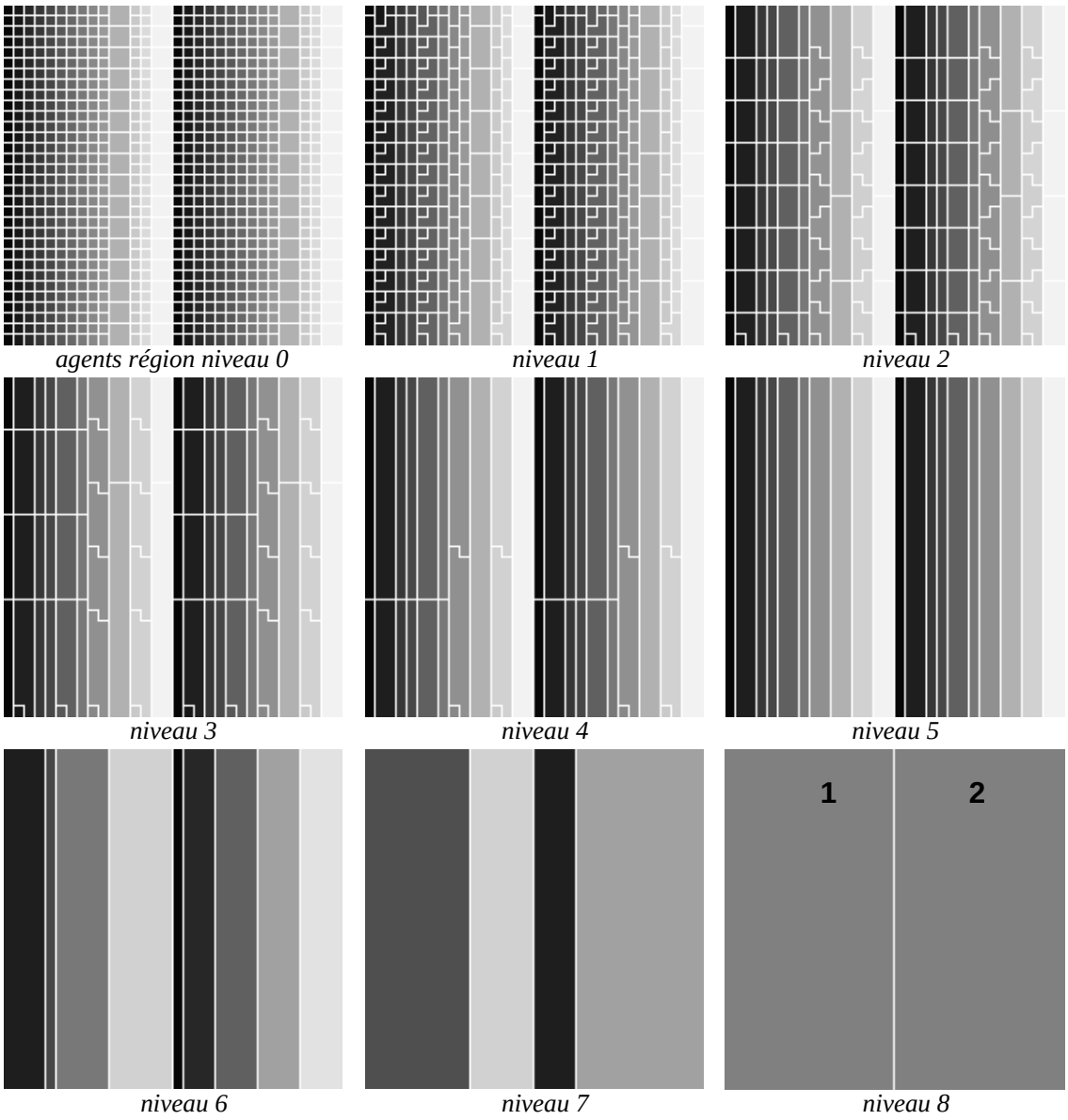


Fig. 10.16 : *Les sept premiers niveaux de la pyramide d'agents.*

10.2.4 Robustesse à la dégradation de l'information *a priori*

Comme nous l'évoquions lors de la section précédente à propos des images de type "scanners du sein 1 & 2", les approches agrégatives classiques sont très sensibles à la précision des seuils fournis par le traiteur d'image. Ainsi, comme le montre la figure (fig. 10.17, p. 231), ce dernier est contraint de fixer après analyse et pour chaque image un seuil dans un intervalle extrêmement restreint (2 à 3 niveaux de gris).

	sur-segmentation	sous-segmentation
Scanner du sein 1	 (a) Pyram. irré. adap., $T = 2$	 (b) Pyram. irré. adap., $T = 5$
	 (c) Division et fusion, $\sigma^2 = 7$	 (d) Division et fusion, $\sigma^2 = 10$
Scanner du sein 2	 (e) Pyram. irré. adap., $T = 3$	 (f) Pyram. irré. adap., $T = 7$
	 (g) Division et fusion, $\sigma^2 = 9$	 (h) Division et fusion, $\sigma^2 = 25$

Fig. 10.17 : *Sensibilité des seuils des approches agrégatives.*

Le tableau 10.1 page 232, représente pour chaque technique agrégative classique et les deux images de scanner, l'intervalle des seuils donnant les résultats escomptés sous la forme :

$$\Big[\text{Seuil maximum avant sous-segmentation} \ ; \ \text{Seuil minimum après sur-segmentation}\Big]$$

Les valeurs données pour la division/fusion sont celles de l'écart-type afin de pouvoir être comparées aux autres approches. Les valeurs données pour la pyramide d'agents correspondent à l'intervalle d'incertitude maximum donnant les résultats escomptés. Les segmentations obtenues avec les intervalles d'incertitude, compris dans l'intervalle [10 ; 2], donnent des résultats équivalents ou meilleurs (du fait de la diminution de l'intervalle d'incertitude).

	Scanner de sein 1	Scanner de sein 2
Pyramide d'agents	$[10 \ ; \ 2] \Rightarrow \Delta = 9$	$[10 \ ; \ 2] \Rightarrow \Delta = 9$
Pyramide irrégulière adaptative	$[4 \ ; \ 3] \Rightarrow \Delta = 2$	$[6 \ ; \ 4] \Rightarrow \Delta = 3$
Division et fusion	$[3 \ ; \ 2.82] \Rightarrow \Delta < 1$	$[4.9 \ ; \ 3.16] \Rightarrow \Delta < 3$

Tab. 10.1 : *Intervalles des seuils donnant une segmentation attendue, la pyramide d'agents est plus tolérante que les deux autres approches agrégatives.*

Remarque : on constate à la lecture du tableau que la pyramide d'agents offre une marge de manoeuvre bien plus large au traiteur d'image, rendant l'approche bien moins sensible (donc plus robuste) à la précision (ou à l'incertitude) portant sur l'information *a priori* fournie.

10.2.5 Capacité à traiter plusieurs images

Chaque méthode de segmentation admet deux entrées : un paramétrage et une image. Nous avons évalué la robustesse des approches agrégatives vis-à-vis d'une variation du paramétrage. Nous allons maintenant évoquer leur robustesse à paramétrage constant selon un changement des images à traiter.

Ce point permet d'évaluer la capacité de généralisation d'une approche qui, paramétrée pour une image, pourra traiter d'autres images similaires de manière automatique.

La technique de fusion d'informations par coopération agent région/agent contour (qui peut être assimilée à une approche de type consensus) semble apporter cette robustesse car un paramétrage identique donne de bons résultats sur les deux images de scanner du sein. Pour la pyramide irrégulière adaptative, seule la valeur de seuil 4 donne des résultats satisfaisants sur les deux images. Ceci confirme la sensibilité de l'approche. La division/fusion ne permet pas de traiter les deux images avec le même paramétrage.

Ces résultats doivent être confirmés sur des bases d'images plus larges, où des problèmes pourraient survenir. Néanmoins, nous souhaitions montrer que l'approche

agent peut rendre une approche agrégative plus robuste grâce à un modèle enrichi permettant une meilleure prise en compte de l'information bas-niveau (région et contour) présente dans l'image.

Cet enrichissement donne aussi un cadre d'intégration des informations issues du modèle. Le modèle apporte des contraintes supplémentaires pouvant espérer de rendre l'approche encore plus robuste. Donnons un exemple :

1. Chaque agent dispose de règles qui prennent en condition la photométrie de l'agent et essayent de déduire sa nature sémantique. Cette opération est difficile pour déterminer si un agent est de type TG ou muscle, mais la photométrie des os (côtes) (fig. 6.2, p. 108) se distingue aisément.
2. Les agents, ayant déterminé leur nature sémantique, la propagent à leurs voisins. Ceci peut être fait à l'aide d'un nouveau comportement intercalé entre exploration et planification.
3. Prenons un agent dont les attributs photométriques le situent entre TG et muscle. Un tel agent recevant "je suis une côte", de la part d'un voisin situé au-dessus de lui, peut décider qu'il est du tissu musculaire. Ainsi interprété, l'agent muscle va pouvoir spécialiser ses traitements, etc.

Nous reviendrons plus en détails sur les perspectives de prise en compte du modèle lors du chapitre 11.

10.3 Évaluation détaillée : dans l'intimité des agents

Dans cette deuxième partie des expérimentations, nous allons analyser la société d'agents qui compose la pyramide. À la manière des sondages sur une population humaine, nous effectuons un ensemble de mesures sur cette société afin d'effectuer une analyse en deux étapes :

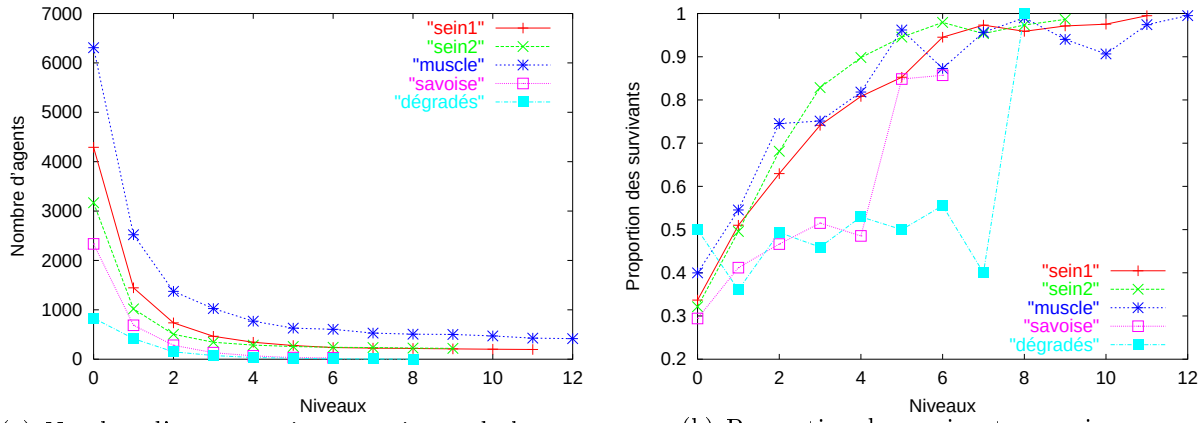
1. L'analyse organisationnelle (§10.3.1 p. 234) se préoccupe de la *population* d'agents et de l'évolution de cette population à travers les niveaux de la pyramide. Puis, elle focalise son attention sur les *éléments structurant* l'organisation, à savoir les relations de voisinage et les intentions de fusions.

2. L'analyse des interactions (§10.3.2 p. 238) vise à quantifier globalement *l'activité* du système en observant par exemple le nombre de messages échangés. Ensuite, elle se portera vers l'impact des interactions coopératives sur les intentions (de fusion) des agents. Enfin, nous analyserons l'évolution des interactions en fonction de l'intervalle d'incertitude.

Parmi les cinq images les trois de type médical (scanner de sein 1,2 et muscles) qui présentent des textures (légères) et du bruit, conduisent globalement aux mêmes évolutions de la pyramide d'agents. Les deux images de synthèse (savoise, dégradés), qui présentent des surfaces homogènes, induisent des évolutions similaires sur la société d'agents.

10.3.1 Analyse organisationnelle

10.3.1.1 Évolution de la population à travers les niveaux



(a) Nombre d'agents région par niveau de la pyramide.

(b) Proportion de survivants par niveau.

Fig. 10.18 : *Évolution de la population à travers les niveaux.*

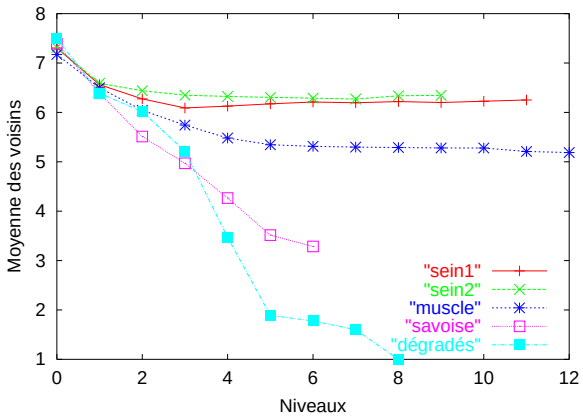
La décroissance du nombre d'agents 10.18(a) dans les premiers niveaux de la pyramide est similaire sur toutes les images, elle suit l'exponentielle :

$$\text{nombre d'agents(niveau)} = 2^{-\text{niveau}} * \text{nombre d'agents au niveau 0}$$

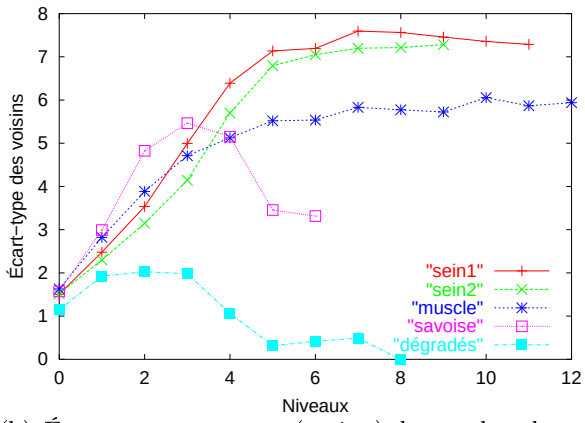
Cette valeur est confirmée par le graphe 10.18(b) qui montre que la proportion de survivants pour les premiers niveaux est proche de 0.5.

Assez rapidement, les images médicales se distinguent des images homogènes, ces dernières maintiennent presque jusqu'à la fin une décroissance exponentielle. Au contraire, pour les images médicales, la proportion de survivants augmente rapidement s'approchant de 1 dès le sixième niveau. Nous pouvons déjà suggérer que le facteur limitant la décimation n'est pas le même pour les images médicales que pour les images homogènes. L'explication (§10.3.1.3 p. 236) est donnée par la chute des intentions de fusion entre agents dans le cas des images médicales.

10.3.1.2 Les relations de voisinage



(a) Moyenne par agent (région) du nombre de voisins région.



(b) Écart-type par agent (région) du nombre de voisins région.

FIG. 10.19 : *Les relations de voisinage dans l'image.*

Nous allons analyser le premier élément structurant l'organisation d'agents : l'adjacence spatiale dans l'image entre agents région.

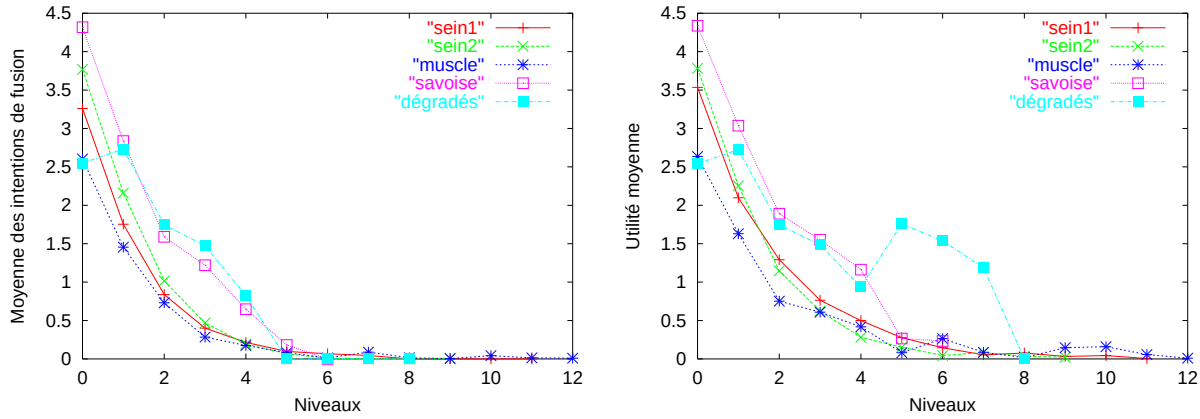
Dans le cas des images médicales, le graphe 10.19(a) montre que, les agents région conservent un nombre moyen constant de voisins. Ces agents semblent bloqués (voir section suivante) dans leur évolution.

Dans le cas des images homogènes, l'accroissement de la surface des agents, qui génère un plus grand nombre de voisins par agent, n'équilibre pas une diminution globale plus importante du nombre d'agents. Ce phénomène est clairement visible sur l'évolution du nombre de voisins dans les niveaux 3, 4 et 5 de l'image "dégradés" (graphe 10.19(a)) & (fig. 10.16, p. 230), où au niveau 5, il n'y a plus que deux voisins (un à gauche et un à droite) par agent.

Le graphe 10.19(b) nous montre un déséquilibre de la taille des voisinages dans les images médicales. Les "petits"[2] agents ne parvenant pas à fusionner (par ex. ceux situés sur le bord inférieur de sein (fig. 10.8, p. 221)& (fig. 10.10, p. 223)) conservent un nombre constant de voisins. En revanche, les "grands" agents (TG, tissus graisseux) sont en contact avec d'autres "grands" agents mais aussi beaucoup de ces "petits" agents évoqués ci-dessus. Ce déséquilibre de voisinage est symptomatique des pyramides irrégulières ; le degré théorique d'un sommet est non borné, il est fortement dépendant du contenu de l'image. L'expérimentation confirme ce fait pour la pyramide d'agents dont l'organisation est structurée par la pyramide irrégulière.

[2]Agents représentant des régions de petite taille.

10.3.1.3 Les intentions de fusion



(a) Moyenne par agent (région) du nombre d'intentions de fusion.

(b) Moyenne par agent (région) de l'utilité des plans de fusion.

Fig. 10.20 : *Les intentions de fusion entre agents.*

Nous allons analyser le deuxième élément structurant l'organisation d'agents : les intentions de fusion entre agents. Si les courbes (graphe 10.20(a)) sont toutes décroissantes, il faut distinguer le cas des images médicales de celui des images homogènes.

Dans le cas des images médicales on peut faire trois observations : (i) la décroissance du nombre moyen d'intentions de fusion par agent montre que ces agents ont de moins en moins de voisins avec lesquels fusionner. (ii) Néanmoins, le nombre moyen de leurs voisins reste constant (graphe 10.19(a) p. 235). (iii) L'observation du graphe 10.20(b) nous indique que l'utilité des plans de fusion des agents décroît très rapidement.

Ces trois observations permettent de conclure qu'après s'être agrégés entre voisins localement similaires, ces agents se retrouvent entourés de voisins différents avec lesquels ils ne peuvent pas construire de plans de fusion utiles. Autrement dit les fusions cessent faute de voisins similaires.

Ceci explique la faible décimation observée (graphe 10.18 p. 234), en effet, la plupart des agents ne fusionneront plus, ils sont "figés" se répétant niveau après niveau, en attendant que d'autres régions plus grandes, ailleurs dans l'image, aient fini de s'agréger entre elles.

En revanche dans le cas des images homogènes, la décroissance des intentions de fusion (graphe 10.20(a)) est due essentiellement à la diminution du nombre de voisins (graphe 10.19(a) p. 235). En effet, on constate en observant le graphe 10.20(b) que les agents (des images homogènes) conservent plus longtemps (jusqu'au niveau 4 pour savoise et niveau 7 pour dégradés) la capacité de concevoir des plans de fusion utiles. Ceci explique le fait que de nombreuses fusions peuvent continuer à s'effectuer jusqu'au niveau 5 pour savoise et 8 pour dégradés (graphe 10.18(b) p. 234).

10.3.1.4 Temps et distribution des calculs

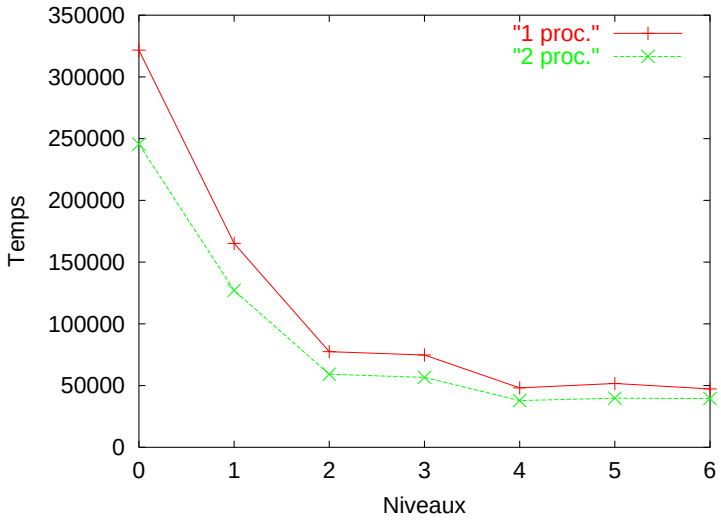


FIG. 10.21 : *Temps de calcul en millisecondes, pour l'image savoise.*

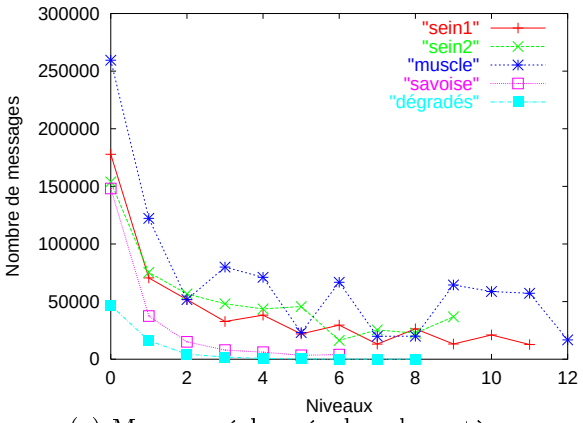
La figure 10.21 donne une évaluation des temps de calcul en millisecondes pour 1 puis 2 processeurs (de type Pentium II 450 mghz). L'architecture n'a pas été optimisée, de plus les évaluations ont été menées sans JIT (chap. 7), qui permet d'envisager un facteur d'accélération variant de deux à cinq.

L'utilisation d'un deuxième processeur accélère d'un facteur 1.3 l'exécution. Ce facteur s'il reste significatif, dénote un goulet d'étranglement présent dans le mécanisme d'allocation de mémoire dynamique de la machine virtuelle java (JVM).

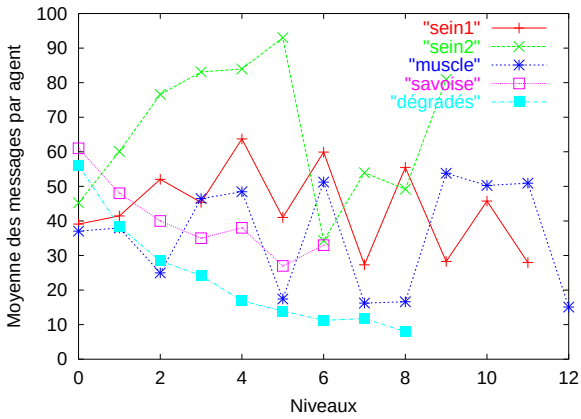
Une programmation rigoureuse et optimisée (évitant au maximum les créations d'objets) permettrait d'augmenter ce facteur. Ce dernier serait alors limité par les trois points de synchronisation globale (§8.6 p. 170).

10.3.2 Analyse des interactions

10.3.2.1 Les échanges de messages



(a) Messages échangés dans le système.



(b) Moyenne par agent de messages échangés dans le système.

Fig. 10.22 : *Les messages échangés entre agents.*

Le graphe 10.22(a) montre la décroissance globale du nombre de messages échangés dans le système.

Cas des images homogènes. La décroissance de messages est plus marquée pour les agents des images homogènes. De plus, cette décroissance est corrélée avec la diminution du nombre d'agents traitant les images homogènes (graphe 10.18(b) p. 234).

L'observation du graphe 10.22(b) nous indique que, pour les agents des images homogènes, la moyenne de messages par agent décroît d'une manière fortement corrélée avec la diminution du nombre moyen de voisins par agent (graphe 10.19(a) p. 235).

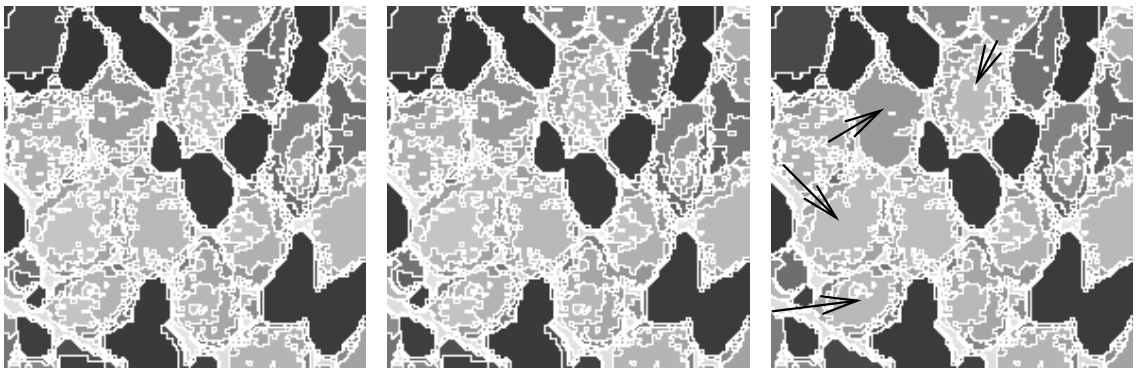
Ces deux corrélations nous autorisent à conclure que la quantité d'interactions dans le système[3] est fonction du nombre d'agents qui y sont présents mais aussi du nombre de liens (relations d'accointance) dont disposent ces agents.

Cas des images médicales. Les remarques précédentes se vérifient (en moyenne) également. En effet, le nombre global (graphe 10.22(a)) de messages diminue avec le nombre d'agents.

On peut faire deux observations, (i) la moyenne de messages par agent (graphe 10.22(b)) oscille autour d'une moyenne ; (ii) le nombre de voisins des agents (graphe 10.18(b) p. 234) reste constant. Ces deux observations nous autorisent à conclure que la quantité d'interactions liée à un agent est corrélée au nombre de ses voisins.

[3] mesurée ici par le nombre de messages échangés

Les oscillations s'expliquent par un surcroît d'activité des agents à un niveau donné. Or, ce surcroît d'activité est corrélé avec des fusions de grandes régions. Prenons le cas de "muscle" au niveau 5 : il y a peu de messages échangés, les évolutions vers l'image du niveau 6 sont faibles (fig. 10.23, p. 239) ; tandis que le passage au niveau 7 montre des fusions entre régions de tailles non négligeables (fléchés sur la figure 10.23).



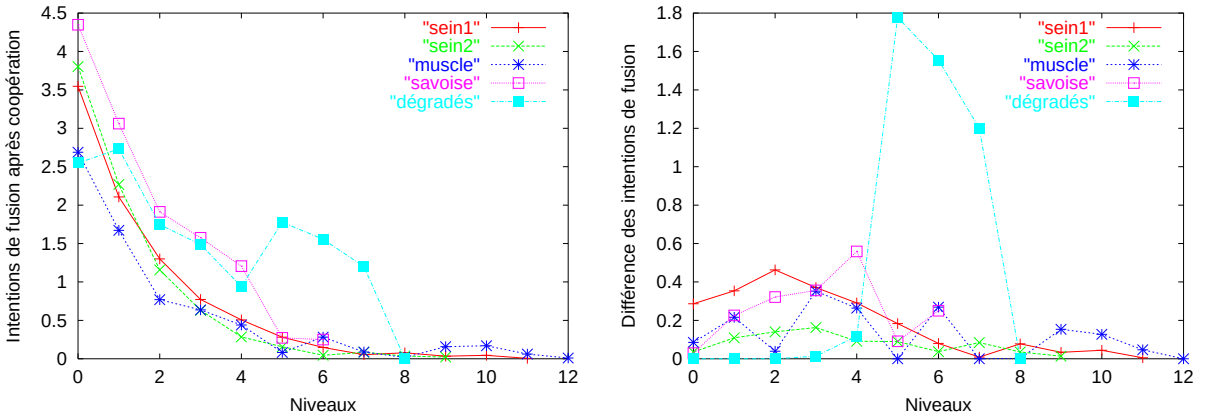
niveau 5 *niveau 6* *niveau 7*

Fig. 10.23 : *Trois niveaux successifs de la pyramide d'agents traitant l'image "muscle". Les résultats de fusions significatives sont indiqués par une flèche.*

De plus, on remarque que les fusions faites entre les niveaux 6 et 7 ont impliquées des agents région qui ont dans leur voisinage un grand nombre d'agents contour (fig. 10.12, p. 225).

Ces derniers ont donc émis un grand nombre de critiques à propos de ces fusions, expliquant l'augmentation de la quantité de messages échangés. Cette hypothèse est confirmée par l'augmentation des critiques pour l'image "muscle" au niveau 6 (graphe 10.26(a,b) p. 241).

10.3.2.2 Les interactions entre agents modifient leurs intentions



(a) Moyenne par agent des intentions de fusion après coopération.

(b) Différence des intentions de fusion après-avant coopération.

Fig. 10.24 : *L'impact de la coopération sur les intentions de fusion.*

Le graphe 10.24(b) est la différence entre le nombre moyen d'intentions de fusion avant coopération (graphe 10.20(a) p. 236) et après coopération 10.24(a). Il permet de mesurer l'impact du comportement coopératif sur les intentions de fusion des agents. Rappelons que le comportement coopératif permet de réunir plus d'informations (région et contour) pour éliminer une hypothèse de fusion incertaine ou la transformer en intention de fusion.

L'impact lors des premiers niveaux de la pyramide est relativement faible : moins de 0.4 intentions rajoutées pour plus de 2.5 intentions. Mais à partir du sixième niveau et pour toutes les images, la plupart des fusions se font à l'aide d'un complément d'informations obtenu par coopération (région/région et région/contour).

Les agents, lors des premiers niveaux, disposent d'un nombre plus important de bonnes fusions possibles. Ceci s'observe sur le graphe de l'utilité des fusions (graphe 10.20(b) p. 236). Un tel environnement favorable incite les agents à resserrer localement leurs seuils de désir nécessaire et ainsi de réduire l'intervalle d'incertitude, ce qui a pour conséquence de limiter l'usage des coopérations.

Les meilleures fusions effectuées, le seuil de désir nécessaire se relâche, ce qui agrandit l'intervalle d'incertitude et provoque le besoin, chez les agents, d'un complément d'informations pour choisir avec qui fusionner.

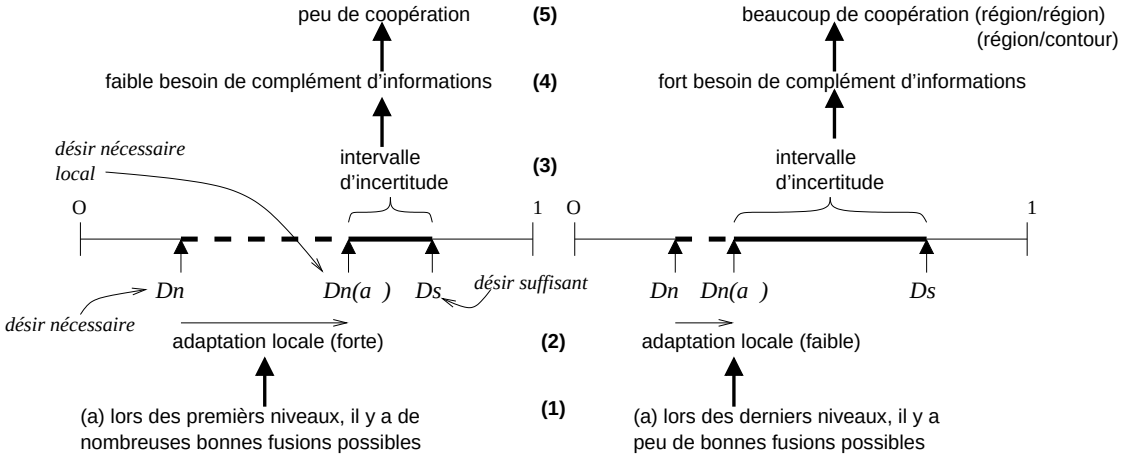
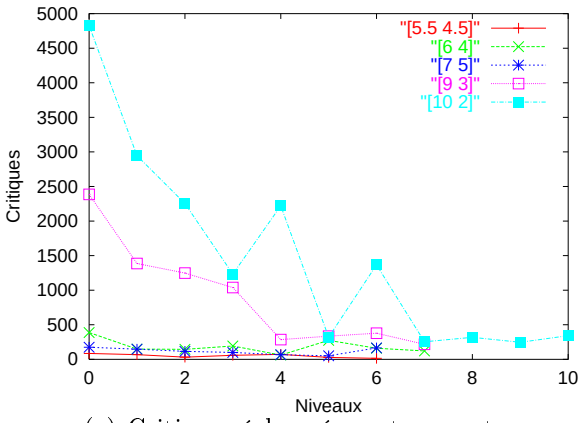


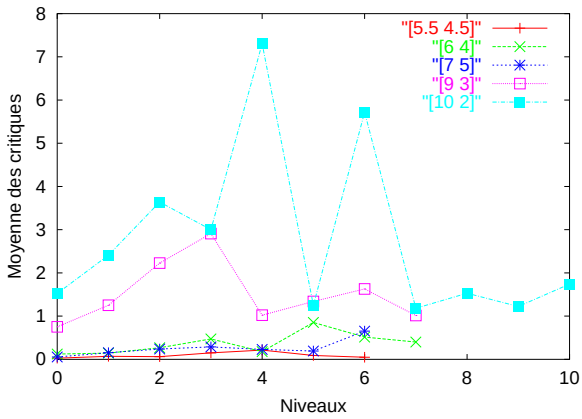
FIG. 10.25 : *L'incertitude entraîne des coopérations entre agents.*

Les expériences montrent donc que l'adaptation locale (resserrant ou agrandissant la fenêtre d'incertitude) liée à un comportement coopératif, permet de récolter des informations supplémentaires uniquement quand les choix de fusions deviennent plus délicats.

10.3.2.3 La coopération face à l'incertitude



(a) Critiques échangées entre agents.



(b) Moyenne par agent du nombre de critiques échangées.

Fig. 10.26 : *Influence de l'a priori sur la coopération.*

Le graphe 10.26(a) montre pour une même image "scanner du sein 2" le nombre de critiques échangées entre agents, selon l'intervalle d'incertitude fixé par le traiteur d'image.

Nous rappelons que cet intervalle correspond aux seuils [désir nécessaire, désir suffisant] (les valeurs ne sont pas normalisées). Par exemple, [10, 2] signifie que pour moins de 2 niveaux de gris de différence, les agents auront l'intention de fusionner ensemble ; pour plus de 10 niveaux de gris de différence il ne voudront pas fusionner ; finalement, pour une valeur intermédiaire, l'agent aura recours à un comportement coopératif, à moins que le désir nécessaire (ici 10) ait été ramené à une valeur plus restrictive par adaptation locale (par ex. 5).

Moins l'information *a priori* est précise (seuils proches l'un de l'autre), plus les agents se trouvent confrontés à des fusions incertaines (voir fig. 10.25, p. 240 sans tenir compte de l'adaptation locale). Pour pallier cette incertitude, ils récoltent des informations supplémentaires à l'aide du comportement coopératif. Ils obtiennent en réponse de leurs voisins (région/contour) des critiques à propos des hypothèses de fusions incertaines.

Pour résumer : l'accroissement de l'incertitude provoque un accroissement des coopérations.

Remarque : Les intervalles d'incertitude proposés (du plus précis [5.5, 4.5] au moins précis [10, 2]) sur les graphes 10.26 donnent tous une segmentation correcte de l'image.

Chapitre

11

Conclusions et perspectives

11.1 Synthèse et bilan

La lenteur des progrès en vision par ordinateur dénote de la difficulté du problème.

Si la recherche sur de nouveaux opérateurs (de segmentation, de reconnaissance de forme...) est essentielle, des progrès significatifs peuvent aussi être espérés grâce à de nouvelles méthodologies de conception et d'analyse des systèmes de traitement de l'information.

Une étude des méthodes de segmentation, menée au chapitre 2, nous a permis d'identifier trois aspects devant être traités avec soin :

1. l'adaptation locale des traitements ;
2. l'intégration et l'expression plus ou moins précise de l'information *a priori* ;
3. la coopération entre approches.

Cependant, plutôt que tâcher de proposer une "panacée" au traitement de ces aspects, nous avons positionné nos travaux au niveau de la mise en œuvre logicielle.

Notre contribution a donc consisté à proposer une architecture logicielle de contrôle basée sur les SMA situés dans l'image. Cette architecture, dont le domaine d'application principal est la segmentation d'images, fournit un cadre conceptuel permettant un traitement efficace des aspects évoqués ci-dessus.

Soucieux d'inscrire notre démarche dans un cadre méthodologique plus global, nous avons mené au chapitre 4 une étude des systèmes de vision.

À la suite de cette étude, nous proposons au chapitre 5 une typologie des architectures logicielles des systèmes de vision. Pour notre approche, nous avons opté pour la famille des agents situés dans l'image qui, selon nous, est la plus à même de favoriser l'adaptation locale des traitements.

Dans le chapitre 7, nous mettons entre parenthèses la thématique image pour présenter la plate-forme agent générique que nous avons développée lors de la thèse.

Cette plate-forme répond à des contraintes (granularité variable des agents...) qui n'étaient remplies par aucun des outils SMA disponibles au début des travaux de thèse.

Si les SMA présentent des avantages dans la conception de systèmes d'information complexes, ils requièrent une nouvelle méthodologie d'analyse et de description des systèmes. En conséquence, nous proposons la méthodologie suivante :

1. Description *globale* et *structurelle* de *l'organisation* regroupant les agents. Dans cette étape, développée au chapitre 8, nous proposons d'adopter les pyramides irrégulières comme élément organisationnel de la population d'agents. Ce choix inscrit notre approche dans la famille des méthodes agrégatives de segmentation d'image.
2. Description *locale* et *fonctionnelle* des agents composant le système. Au cours de cette étape, présentée au chapitre 9, nous proposons une mise en œuvre particulière des aspects (i) de coopération région/région et région/contour ; (ii) de l'adaptation locale ; (iii) de l'intégration de l'incertitude dans la connaissance *a priori* .
3. Analyse *globale* et *fonctionnelle* où l'on vérifie que l'ensemble des interactions locales réalisent bien ce pour quoi le système est conçu. Les expérimentations comparatives menées lors de cette dernière étape (chap. 10) montrent que la pyramide d'agents obtient des résultats au moins aussi bons et dans tous les cas plus robustes que cinq autres méthodes de segmentation. À la suite de cela, nous proposons une analyse globale tant structurelle qu'interactionnelle de l'organisation d'agents.

11.2 Perspectives

Améliorer l'utilisation des contours : il semble opportun de donner aux agents contour un rôle plus important que celui qu'ils occupent actuellement. Ceci favoriserait une meilleure localisation des frontières en évitant l'aspect "déchiqueté" visible sur la segmentation de l'image "muscle" (fig. 10.11, p. 224). Nous suggérons pour cela l'utilisation de contours actifs à la place de simples chaînes de segments de contour. Les agents région permettraient l'initialisation automatique d'agents "contours actifs", et ces derniers contraindraient les fusions entre agents région.

La décimation duale (§3.9.4 p. 49) pourrait avantageusement remplacer la décimation classique des pyramides irrégulières que nous employons. En effet, l'observation de la figure 8.2 page 160, montre la construction séquentielle des niveaux.

La décimation duale permettrait une extraction rapide des maximas locaux, c'est-à-dire des régions importantes et faciles à extraire. Cette information peut ensuite être utilisée dans une approche descendante pour assister l'extraction des régions plus délicates.

Vers l'interprétation : les agents fournissent une abstraction adaptée à la mise en œuvre d'informations *a priori* plus riches, reflétant des aspects géométriques voire

sémantiques.

On peut imaginer qu'un modèle représenté par une hiérarchie (fig. 11.1) d'agents (mémoire à long terme) interagisse avec la pyramide d'agents (mémoire de travail à court terme). Le rôle de cette dernière n'est plus seulement de regrouper les primitives entre elles, mais aussi d'identifier les concepts dans l'image.

Un agent de la pyramide ayant identifié un concept se reproduit, en créant un nouvel agent doté de connaissances *a priori* spécialisées dans l'identification de nouveaux concepts.

L'introduction progressive de connaissances de plus en plus contraignantes, portant sur des niveaux de description de plus en plus élevés, suggère la capacité de la pyramide d'agents à traiter les niveaux intermédiaire et haut de la vision.

De telles perspectives semblent difficiles à envisager dans le cadre de la pyramide de graphes.

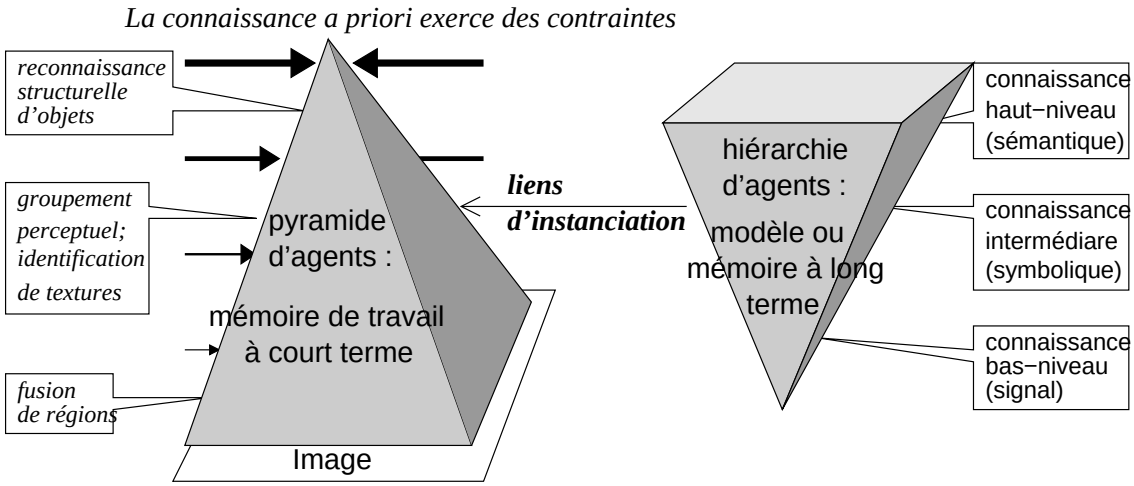


Fig. 11.1 : *Perspectives d'intégration des différentes étapes de la vision.*

Troisième partie

Annexes

Annexe

A

Compléments sur les méthodes de segmentation

Nous évoquons dans ce chapitre un complément d'information sur les différentes techniques de segmentation traitées au chapitre 2.

Nous donnons plus de détails sur les autres méthodes de segmentations que nous avons utilisées lors des expérimentations comparatives du chapitre 10.

A.1 Classification

A.1.1 Les classifieurs supervisés

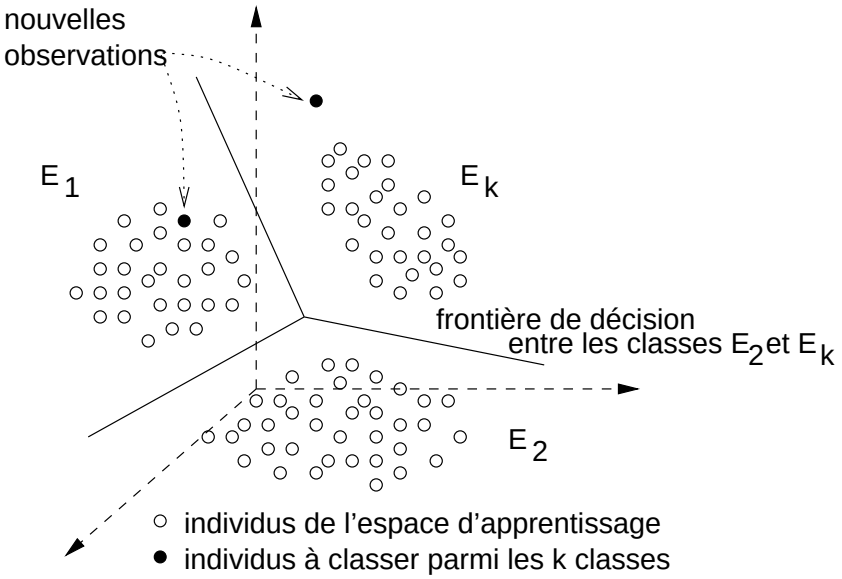


Fig. A.1 : *Classification supervisée : cas de fonctions discriminantes (frontières de décisions) linéaires.*

De tels classifieurs supposent que l'on dispose d'informations *a priori* disponibles sous la forme d'un ensemble de texels d'apprentissage dont on connaît l'appartenance parmi k classes. L'apprentissage consiste donc à construire une fonction de discrimination capable de séparer les échantillons de l'ensemble d'apprentissage en respectant (au mieux) leur classification *a priori* . On aborde ensuite l'étape décisionnelle de ces techniques qui consiste à affecter les nouveaux individus dans les classes en leur appliquant la fonction de discrimination.

L'analyse factorielle discriminante [Tom88] est une méthode de discrimination géométrique qui procède d'abord par une recherche des axes les plus discriminants. Ces axes sont ceux qui maximisent la variance interclasse (effet de séparation) et qui minimise la variance intraclasse (effet d'agrégation). Dans le cas de deux classes, cet axe est le vecteur joignant leurs centres de gravité.

La règle d'affectation d'une nouvelle observation est simple : on la projette sur cet axe et on l'affecte à la classe dont le centre de gravité est le plus proche. La métrique pour évaluer la proximité est souvent l'inverse de la matrice de variance/covariance globale, elle prend en compte la dispersion globale des individus (permettant de "réduire" les distances suivant les axes où la dispersion est la plus grande).

Les approches géométriques ne permettent pas de prendre en compte l'information *a priori* sur les classes, à savoir : les différences de dispersion et de probabilité *a priori* propres à chaque classe.

Les classifieurs Bayésien abordent la classification sous un aspect probabiliste qui fournit un cadre de modélisation des informations *a priori* .

On observe un vecteur (individu, pixel ou texel) $x \in \mathbb{R}^p$ (espace des attributs) le problème est de savoir de quelle classe (texture) -parmi $E_1, .., E_k$- cette observation est issue. Ces informations introduites dans le modèle sont les suivantes :

- les probabilités *a priori* de chaque classe notées $p(E_1), p(E_2), ..., p(E_k)$ correspondant à leurs fréquences d'apparition ;
- la loi d'observation $p(x|E_i)$ (loi conditionnelle) qui traduit la loi de probabilité de la variable aléatoire (individu, pixel ou texel) sachant que sa classe d'appartenance est E_i.

Cependant, la probabilité qu'il nous faut connaître est $p(E_i|x)$ c'est-à-dire la probabilité d'appartenir à la classe E_i sachant que x est réalisé. Pour cela on utilise la loi de Bayes :

$$\begin{aligned} p(E_i|x)p(x) &= p(x|E_i)p(E_i) \\ p(E_i|x) &= \frac{p(x|E_i)p(E_i)}{p(x)} \\ p(E_i|x) &= \frac{p(x|E_i)p(E_i)}{\sum_{i=1}^{k} p(x|E_i)p(E_i)} \end{aligned}$$

La règle Bayésienne consiste à attribuer x à la classe dont il maximise la probabilité *a posteriori* : $p(E_i|x)$. On suppose que $x|E_i$ suit une loi normale $N(\mu_i, \Sigma_i)$ dont les

paramètres sont connus :

$$p(x|E_i) = \frac{1}{(2\pi)^{p/2}(\det \Sigma_i)} \exp -\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)$$

Le dénominateur étant le même pour toutes les classes, la règle Bayésienne devient :

$$x \in E_i \text{ tel que } i = \arg\max_{E_i}\{(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i) - 2\ln(p(E_i)) + \ln(\det \Sigma_i)\}$$

La frontière de décision entre E_j et E_i est une surface au niveau de laquelle la valeur du calcul ci-dessus s'équilibre pour les deux classes. En général cette surface est une quadrique, on parle alors de classifieurs non linéaires. Dans le cas particulier où les matrices de variance/covariance sont toutes égales, la surface est un hyperplan (de $\mathbb{R}^p$), le classifieur devient linéaire. Si par ailleurs toutes les classes sont équiprobables, la règle Bayésienne correspond à la règle géométrique énoncée plus haut. Au final si les matrices Σ_i sont aussi diagonales, la frontière de décision est l'hyperplan orthogonal à la droite qui joint les centres des deux classes.

L'approche Bayésienne permet de modéliser une information *a priori* très riche et de cette manière d'obtenir une classification optimale. Comme nous venons de le voir, l'approche peut-être simplifiée avec précaution. Par exemple, si l'échantillon d'apprentissage permettant d'estimer les matrices Σ_i est trop petit, il peut être préférable de se ramener à des surfaces de discrimination linéaires. Il se peut aussi que l'on ne connaisse pas la loi de probabilité conditionnelle suivie par $x|E_i$. Dans ce cas, on utilisera une approche non paramétrique comme la méthode des K-plus proches voisins.

K-plus proches voisins [Cov67] est une méthode de classification supervisée non paramétrique. En effet, si l'on ne connaît pas la loi de probabilité de la variable aléatoire conditionnelle $x|E_i$, il faut l'estimer. La première approche (celle des fenêtres de Parzen[Par62]) consiste à estimer $p(x|E_i)$ en subdivisant $\mathbb{R}^p$ en cellules dans lesquelles on compte le nombre d'individus. En divisant ce nombre par le nombre total d'individus, on obtient une estimation de la probabilité conditionnelle dans cette cellule. Or une bonne estimation suppose un nombre suffisant d'individus dans chaque cellule, ce qui n'est pas forcément le cas si leur répartition n'est pas homogène dans $\mathbb{R}^p$. Dans la méthode des K-plus proches voisins, on estime la densité de probabilité (ddp) non pas sur une cellule de taille fixe mais sur les k plus proches voisins. De fait on attribuera x à la classe la plus représentée parmi ses k plus proches voisins. Cette méthode est relativement simple à mettre en œuvre et donne des résultats inférieurs mais proches de la classification Bayésienne.

A.1.2 Les classifieurs non supervisés

Ces méthodes cherchent une partition de l'espace des attributs, laquelle appliquée aux échantillons (texels), va maximiser un critère. Ces approches utilisent (par définition) très peu d'informations *a priori* : on leur fournit le critère à maximiser et parfois le nombre de classes que doit comporter la partition. De telles approches sont donc à utiliser avec précaution, on les emploie lorsque l'on ne dispose pas de modèle ou échantillons des classes à extraire.

Citons par exemple la méthode des *Nuées dynamiques* [Did71] qui cherche une partition optimale en k classes. Chaque classe E_i est représentée par un noyau n_i (aussi appelé centre mobile) dont la position évolue suivant une procédure itérative impliquant les deux étapes ci-dessous :

1. Connaissant les k noyaux, une fonction "d'affectation" attribue chaque individu à un noyau (le plus proche au sens d'une métrique donnée). On obtient ainsi une partition de nos individus.
2. De cette partition, on déduit pour chaque classe son nouveau représentant ou noyau qui sera son centre d'inertie.

Cette procédure itérative permet de minimiser un critère d'adéquation entre l'ensemble des noyaux et une partition des individus. L'algorithme converge lorsque les noyaux sont "au centre" des classes qu'ils doivent représenter. Malheureusement, cette procédure est hautement dépendante du placement initial des noyaux.

A.1.3 Remarques et conclusions

Les approches statistiques de classification fournissent un cadre formel dans lequel peut s'exprimer de façon optimale la connaissance *a priori* que l'on a des textures à traiter. Les modèles peuvent être dégradés et simplifiés dans le cas où cette information est moins riche, cette dégradation rend le processus de segmentation moins robuste (exemple : sensibilité à l'état initial).

Ces techniques procèdent par une approche locale tant pour le calcul des attributs que pour le choix d'affectation dans une des classes. La question est de savoir quelle doit être la taille de la fenêtre de calcul : si elle est trop petite, les paramètres estimés ne traduiront pas la texture que l'on cherche à segmenter et l'image sera alors sursegmentée ; si elle est trop grande, des pixels n'appartenant pas à la texture seront pris en compte pour le calcul des attributs. On est donc confronté à un problème de choix de la résolution optimale pour analyser une texture, cette résolution étant différente selon la texture considérée. Les approches multirésolutions [Uns89] ou pyramidales [Lam94] proposent des solutions pour traiter les textures selon une résolution ou une échelle plus adaptée.

Plaçons nous dans le cas de deux sous-régions distinctes "pour l'oeil humain" comme par exemple deux cellules légèrement en contact et dont les pixels sont tous de la même classe. L'approche classification puis regroupement des composantes (pixels) connexes fusionnera ces deux sous-régions en une seule. Il peut être argumenté que cette situation nécessite trop d'informations *a priori* et ne relève pas de la vision bas niveau. Cependant, une coopération avec une approche contour permet bien souvent de résoudre le problème.

Notre propos est de faire remarquer l'intérêt des approches coopératives région-contour ou même région-région, afin de "faire au mieux" le plus tôt possible et limiter les corrections et reprises de traitements imposées par les couches haut-niveau de la vision.

Il est toutefois intéressant de pouvoir intégrer ces techniques dans un cadre souple où les couches haut niveau pourront réagir aux résultats fournis par la segmentation et guider au mieux celle-ci par les connaissances *a priori* qu'elles possèdent de la scène.

Chaque pixel est classé indépendamment de ses voisins, il n'y a pas de fonction globale à optimiser, ainsi les décisions sont prises "trop tôt" sans tenir compte de toute l'information extraite de l'image, contrairement aux approches basées sur les champs de Markov. De plus, bien que l'aspect anisotropique d'une texture peut être pris en compte via l'utilisation des matrices de cooccurrence, les champs de Markov fournissent une formalisation intéressante pour la modélisation de texture dont la variation d'intensité des pixels est fonction de la direction.

A.1.4 Classification : méthodes markoviennes

Depuis les publications de Geman & Geman [Gem84], les champs de Markov en vision bas niveau sont couramment employés [Dub89]. Les approches markoviennes plongent l'analyse d'image dans un cadre stochastique où la meilleure segmentation (attribution d'une étiquette à chaque pixel) minimise une énergie globale. Cette minimisation s'effectue sur un graphe où la loi globale est décomposée au travers d'interactions locales entre noeuds du graphe. Ceci est rendu possible par l'équivalence contenue dans le théorème de Hammersley-Clifford entre les champs de Markov et les distributions de Gibbs.

On utilise un large éventail d'algorithmes d'optimisation déterministe (Maxima Conditionnels Itérés ICM) ou stochastique (recuit simulé) au comportement connu. Cette optimisation ne s'effectue pas simultanément sur toute l'image, mais en étudiant de façon itérative chaque site afin de trouver l'étiquette qui minimise une énergie locale calculée sur le voisinage du site.

Soit S un ensemble de sites, soit $\lambda = \{\lambda_i | i \in S\}$ l'ensemble des étiquettes associées à chaque site et soit $x = \{x_i | i \in S\}$ l'image. Si les vecteurs aléatoires λ et x sont des champs de Markov, alors la loi de probabilité conditionnelle $P(\lambda_i | \{\lambda_j | j \in S\} - \lambda_i)$ est égale à la loi de probabilité conditionnelle de λ_i, connaissant seulement son voisinage $P(\lambda_i | V(\lambda_i))$. Ceci étant aussi valable pour x_i. Cette propriété nous permet d'analyser la valeur de λ_i avec de simples interactions locales dans le voisinage du site i.

Toute la connaissance *a priori* sur chaque texture k à reconnaître est codée dans une loi de paramètres θ_k qui donne la probabilité de λ_i sachant x_i, son voisinage $V(x_i)$, et le voisinage $V(\lambda_i)$ de l'étiquette λ_i.

$$P^{\theta_k}(\lambda_i | x_i, V(x_i), V(\lambda_i)) = \frac{1}{Z(\theta_k)} \exp(-U_1^{\theta_k}(\lambda_i | V(\lambda_i)) - U_2^{\theta_k}(\lambda_i | x_i, V(x_i)))$$

Cette loi est composée de deux fonctions d'énergies locales :

- La fonction $U_1^{\theta_k}(\lambda_i | V(\lambda_i))$ représente le modèle *a priori* liant l'étiquette du site i aux étiquettes de son voisinage. Ce modèle de régularisation traduit souvent le fait que les étiquettes voisines ont une forte probabilité d'être égales.

– La fonction $U_2^{\theta_k}(\lambda_i|x_i, V(x_i))$ représente le modèle d'observation liant l'étiquette du site i aux valeurs des pixels dans le voisinage du site.
Dans le cas d'une texture isotrope, on choisira pour $U_2^{\theta_k}$ un modèle de lissage où l'énergie est minimale (la probabilité maximale) lorsque les pixels du voisinage de i ont des valeurs proches de x_i et ce quelle que soit l'orientation des cliques.
Dans le cas d'une texture anisotrope, on pondérera l'énergie apportée par des cliques ayant une certaine orientation afin de favoriser certaines structures orientées dans la luminance, (par ex. de l'herbe). Citons des modèles comme le lissage anisotropique, le modèle de Ising ou de Potts.

On attribuera à λ_i l'étiquette (la texture) dont la loi maximise la probabilité conditionnelle, soit :

$$\lambda_i = \arg\max_{\theta_k} P^{\theta_k}(\lambda_i|x_i, V(x_i), V(\lambda_i))$$

Ce calcul (qui nécessite par exemple des inversions de matrices) doit être itéré sur chaque site, et plusieurs fois sur toute l'image, en fonction du choix de l'algorithme d'optimisation. Les approches Markoviennes sont donc assez exigeantes en ressources de calcul.

A.2 Filtrage et transformées

Il s'agit de trouver une transformation de l'image vers une base qui fasse émerger des propriétés de "concentration de l'information". Ainsi, la classification des pixels, ou blocs de pixels, en textures à reconnaître se fait dans un espace transformé de dimension réduite. Le principe de base consiste à exploiter la forte corrélation (redondance d'informations) qui existe entre pixels voisins. Ainsi, une représentation fréquentielle de notre image concentrera l'essentiel de son information (énergie) dans les basses fréquences. On procède par une transformation locale (TF, DCT ou Hadamard) d'une sous-image (des blocs de taille fixe) ou sur une fenêtre centrée sur chaque pixel. Par exemple, dans le cas d'analyse d'images LANDSAT [Gra73]), l'image est découpée en blocs 32×32 et une transformée de Fourier est calculée sur chacun de ces blocs.

La discrimination parmi les textures à reconnaître peut être effectuée à l'aide d'un filtre adapté (pour chaque texture à reconnaître) s'appliquant sur la puissance spectrale tronquée aux basses fréquences ; on choisit la texture dont le filtre donne la plus forte réponse. Cette analyse peut aussi être faite à l'aide des méthodes de classification multidimensionnelles (§2.2.2 p. 19) qui font partie de la boîte à outils des traiteurs de signaux. Remarquons que les approches basées sur l'analyse de la puissance spectrale se sont montrées moins efficaces que celles basées sur des moments spatiaux d'ordre deux (matrice de cooccurrence) ou même d'ordre un [Wes76].

Dans le cas des micro-textures, le voisinage sur lequel est effectué la transformation doit être petit. Ce voisinage est agrandi en fonction de la taille de la texture à traiter. En effet, la description de texture est hautement dépendante de la résolution d'analyse. Ainsi les ondelettes [Mal89] ou la transformée de Gabor [Bov90]

proposent une base de décomposition de l'image adaptée à une analyse multi-échelle des textures.

Un second argument favorise les transformées de type Gabor ou ondelettes par rapport aux transformées de type Fourier. Si ces dernières isolent bien les fréquences du signal, elles perdent la localisation spatiale de l'occurrence d'une fréquence donnée dans l'image (un événement présent dans l'image sera répercuté sur tout le spectre). Les approches par ondelettes ou Gabor font une analyse de l'image traitée à une résolution de plus en plus faible en étirant à la manière d'un accordéon la fonction de base de décomposition. Cette méthode permet de conserver la structure spatiale de l'image.

Un exemple d'utilisation est présenté par [Bou91] : une première analyse à faible résolution permet de localiser approximativement les frontières entre textures, puis on affine la localisation de ces frontières par une analyse à plus forte résolution contrainte par les résultats de la première analyse.

A.3 Approches frontière : fermeture de contours

Les chaînes de sites de contour extraites sont souvent incomplètes. En effet, pour éviter les erreurs (construction d'une chaîne de site due au bruit), il a fallu employer des seuils et des contraintes générant la création d'un grand nombre de petites chaînes de points de contours. On souhaite désormais relier ces chaînes entre elles afin d'obtenir de plus grandes chaînes ou même des contours fermés définissant une région, permettant finalement de segmenter notre image par une approche contour.

Nous présentons deux techniques basées sur la théorie des graphes : la programmation dynamique et un algorithme glouton A*. Ces deux approches supposent, comme nous allons le voir, la notion de sous-structure optimale. Chaque site de l'image des gradients est associé à un noeud du graphe, soit x_a, x_b deux noeuds, on cherche le chemin optimal (suivant les critères précédemment définis) entre ces deux noeuds dans le graphe. C'est un problème classique de théorie des graphes dont la solution est basée sur la programmation dynamique.

La programmation dynamique est une méthode d'optimisation opérant par phases (ou séquences) dont l'efficacité repose sur le principe d'optimalité de Bellman : "Toute politique optimale est composée de sous-politiques optimales". Un chemin optimal entre deux noeuds du graphe est composé de sous-chemins optimaux. La recherche est faite sur un graphe reflétant une fenêtre $N \times M$ englobant les deux extrémités.

Soit $P_{(a,b)}$ un chemin menant de x_a à x_b, si $P_{(a,b)}$ est optimal et que $x_i \in P_{(a,b)}$ alors $P_{(a,i)}$ est optimal. Ainsi l'optimisation globale peut être ramenée à une optimisation locale. Soit x_i un noeud ayant trois prédécesseurs (en 8 connexité noté $pred(x_i)$) qui sont les noeuds précédents x_i dans tous les chemins menant à x_i depuis x_a. Soit $C(x_k)$ le coût (minimal) du chemin optimal menant à x_k depuis x_a, soit $g(k,i)$ le coût

de l'ajout de x_i dans ce chemin. Le principe d'optimalité nous dit que la politique optimale (de coût minimal) menant à x_i peut être obtenu en choisissant la politique localement optimale depuis les chemins partiels menant à x_i :

$$C(x_i) = \min_{x_k \in pred(x_i)} \{C(x_k) + g(k,i)\}$$

Programmation dynamique
/* `liste` *et* `liste-suivante` *: structure de données contenant une liste de point*
x_i : noeud courant
$succ(x_i)$: fonction retournant tous les noeuds successeurs de x_i c'est-à-dire des noeuds voisins de x_i appartenant à un chemin menant vers x_b avec une contrainte de non retour en arrière
*$pred(x_i)$: fonction retournant tous les noeuds prédécesseurs de x_i */
tant que(VRAI)
- **pour tout**($x_i \in$ `liste`)
  - $C(x_i) = \min_{x_k \in pred(x_i)}(C(x_k) + g(k,i))$
  - on crée un liant avec le prédécesseur qui minimise
  - `liste-suivante` $\leftarrow$ `liste-suivante` $+succ(x_i)$
  - **si** $*(x_i = x_b)$ **retourne** $P_{(a,b)}$ en suivant les liens construits

  fin pour tout
- `liste` $\leftarrow$ `liste-suivante`

fin tant que

Algo. A.1 : *Fermeture de contour par programmation dynamique.*

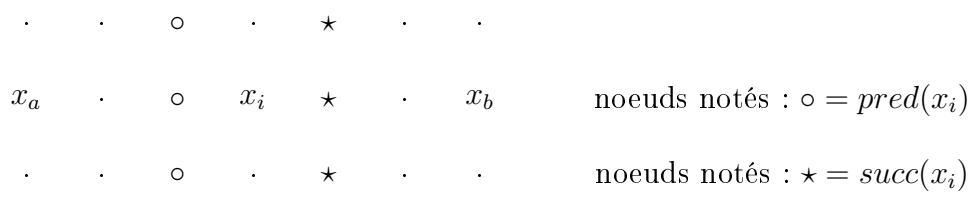


Fig. A.2 : *Les noeuds prédécesseurs et successeurs du noeud courant x_i, cette relation traduit le fait que le graphe respecte la structure de l'image.*

L'algorithme A* [Har68] est issu de l'intelligence artificielle (utilisation en image [Mar72]). En effet, la programmation dynamique est une approche parfois trop lourde, une approche gloutonne faisant toujours le choix du meilleur sur le moment peut être suffisante. A* effectue donc une recherche arborescente du meilleur chemin entre x_a à x_b suivant une stratégie gloutonne et dite du "meilleur d'abord" car l'exploration s'effectue à partir du noeud intermédiaire x_i dont le coût $f(x_i)$ est minimal.

$$f(x_i) = g(x_i) + \hat{h}(x_i)$$

Où $g(x_i)$ est le coût cumulé du chemin x_a à x_i. La fonction de coût du noeud x_i est une somme de deux fonctions : (i) $g(x_i)$ qui est le coût cumulé du chemin menant de x_a à x_i ; (ii) $\hat{h}(x_i)$ est une heuristique estimant $h(x_i)$ qui est le coût réel du chemin de x_i à x_b.

A*
$x_i \leftarrow x_a$
tant que(VRAI)
– `liste` $\leftarrow$ `liste` $+succ(x_i)$
– évalue $f(x_k)$ pour tous les nouveaux noeuds insérés dans la liste
– $x_{next} \leftarrow \arg\min_{x_k \in \texttt{liste}} f(x_k)$
– **si** $(x_{next} = x_b)$ **retourne** $P_{(a,b)}$ en suivant les liens construits
– **sinon** crée un lien menant de x_{next} à x_i, $x_i \leftarrow x_{next}$
fin tant que

Algo. A.2 : *Fermeture de contour par l'algorithme A*.*

$g(x_i) = \sum_{P_{(a,i)}} c(x_i)$ où $c(x_i)$ est le coût de l'ajout de x_i dans un chemin. On choisira pour $c(x_i)$ des critères identiques à ceux décrits dans le paragraphe précédent c'est-à-dire favorisant de forts gradients et de faibles variations d'angle. Le choix de l'heuristique $\hat{h}(x_i)$ évite d'explorer le chemin qu'elle estime inutile, elle a donc un fort impact sur la rapidité de l'algorithme. C'est un peu une boule de cristal permettant d'estimer l'avenir à la manière d'un joueur d'échec qui n'explore pas certaines combinaisons car son intuition ou expérience le lui déconseille. $\hat{h}$ est dite admissible si elle ne sous-estime jamais h, dans ce cas A* garantit de trouver la solution optimale. Si $\hat{h}(x_i) = 0$ la recherche est exhaustive, de complexité $O(N3^{M-1})$, on voit donc l'intérêt d'une bonne heuristique qui diminue de façon significative la complexité en moyenne. On pourra fixer par exemple : $\hat{h}(x_i) = dist(x_i, x_b)$.

Comparaisons : les approches à base d'heuristique nécessitent le stockage de tous les chemins partiels construits, mais elles se montrent plus performantes pour trouver un chemin entre deux points[Mar76], bien que la complexité de l'approche programmation dynamique soit en $O(3N(M-1))$. Toutefois cette dernière permet la construction simultanée (sans sur-coût) de chemins entre un ensemble de points de départ et un ensemble de points d'arrivée, caractéristique intéressante si l'on ne connaît pas exactement les extrémités à joindre.

Cornen et al. proposent dans [Cor94] chap. 16 et 17 une présentation et une comparaison entre stratégies gloutonnes et programmation dynamique d'où il ressort que la programmation dynamique est plus lourde en termes de traitements que les approches gloutonnes. Néanmoins, ces dernières, afin d'être optimales, nécessitent d'être plongées dans le cadre d'une structure dont les sous-structures sont optimales de type "Matroïdes".

Annexe

B

Compléments sur les techniques de représentation de la connaissance

Nous évoquons dans cette annexe un complément d'information sur les différentes techniques de représentation de la connaissance traitées au chapitre 4.

Nous abordons tout d'abord le problème de l'inférence des systèmes experts, en présentant l'algorithme de Rete utilisé par le système expert que nous employons : Jess.

Puis nous évoquons les frames et les réseaux sémantiques qui sont deux techniques de représentation de la connaissance couramment employées par les systèmes de vision présentés au chapitre 5.

B.1 Mécanismes d'inférence des systèmes experts

Problème des performances : la compilation des règles. Afin d'améliorer (en chaînage avant) les mauvaises performances inhérentes aux systèmes à base de règles, des algorithmes Rete [For82] (cf. fig. B.1) ou Treat [Mir90] effectuent une compilation de ces règles sous la forme d'un réseau[1]. Les noeuds du réseau (mémoires alpha) représentent un type de fait (par ex. `(père-de ?x ?y)`). Les liens entre ces mémoires traduisent les liens et/ou entre faits dans la partie condition des règles. Rete utilise et maintient des mémoires supplémentaires : les mémoires béta. Elle reflètent les jointures sur les variables entre types de fait. Par exemple : `(père-de ?x ?y) AND (frère-de ?x ?z)` génère une jointure sur la variable `?x`.

Treat ne maintient pas ces jointures, il les calcule dynamiquement, accélérant le traitement dans le cas où certains faits sont retirés de la base (cas non monotone).

En bref dans ces types de réseaux, ce sont les nouveaux faits ajoutés qui cherchent les règles à déclencher plutôt que l'approche naïve qui procéderait de façon inverse.

[1] *rete* signifie réseau en latin

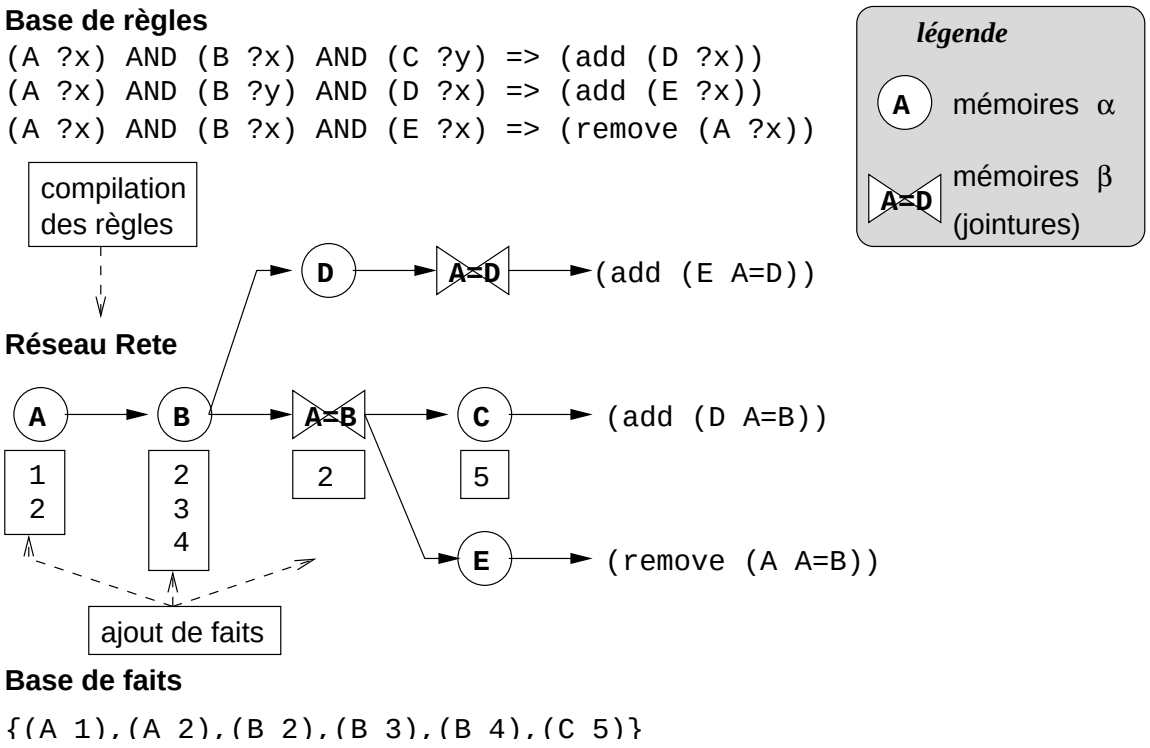


FIG. B.1 : *Le réseau Rete compile les règles afin d'accélérer les performances de l'inférence.*

La non monotonie (retrait de faits) de certains systèmes pose des problèmes de cohérence des faits dont la vérité est conditionnelle à celle d'autres faits. Des systèmes comme TMS (Truth Maintenance System) [Doy79] et ATMS [Kle86a, Kle86b, Kle86c] proposent de résoudre ce type de problèmes en maintenant à tout moment le support[2] d'une proposition. Ce support permet aussi de donner la justification de l'existence d'un fait, renforçant ainsi un des points forts des systèmes à base de connaissances c'est-à-dire leur lisibilité et capacité d'explication de leurs raisonnement. Sur ce dernier point, on reproche souvent aux approches connexionnistes leur aspect "boîte noire".

Les problèmes du contrôle et des performances peuvent aussi être abordés en groupant les règles en paquets, lesquels seront inhibés ou activés par des méta-règles de contrôle. Remarquons qu'une architecture composée de spécialistes (paquets de règles) et contrôleurs (règles de contrôle) conduit assez naturellement aux architectures de type tableaux noirs.

B.2 Les frames

Proposés par Minsky [Min75] en 1975 pour la vision par ordinateur, les Frames permettent une représentation plus naturelle que les règles des prototypes d'objets

[2]Le support d'un fait F est l'ensemble de faits ayant provoqué l'assertion de F.

et des scènes (connaissances haut-niveau). Un Frame se présente comme un objet composé d'attributs (*slots*) représentant la connaissance descriptive. À chaque attribut peuvent être attachées différentes *facettes* reflétant les valeurs possibles ou par défaut de l'attribut. Une facette peut aussi prendre la forme d'une procédure réflexe déclenchée par une action particulière sur l'attribut. Par exemple, la procédure associée à la facette *if-added* se déclenche lorsque l'on positionne la valeur de l'attribut, permettant une propagation réflexe de l'information. Ce mécanisme correspond à un raisonnement guidé par les données (chaînage avant). La procédure associée à la facette *if-required* se déclenche lorsque l'on demande la valeur non disponible de l'attribut, permettant une propagation réflexe de requêtes. Ce mécanisme correspond à un raisonnement guidé par les buts (chaînage arrière).

Cet attachement procédural permet de représenter la connaissance opératoire du système. Dans VISIONS [Han78], Hanson et Riseman utilisent ce mécanisme pour propager l'information et donc générer de nouvelles hypothèses à partir de concepts facilement identifiables. Dans ce système, un Frame[3] représente une classe d'objets, et les slots la connaissance descriptive – photométrique et géométrique – de cette classe. Les slots sont aussi utilisés pour coder les relations spatiales et les probabilités conditionnelles entre classes. Un Frame est donc une brique active de connaissance élémentaire insérée dans un réseau de Frames. Une stratégie hétérarchique d'interprétation s'appuie donc sur les connaissances et les liens du Frame pour générer de façon opportuniste de nouvelles hypothèses au sein du réseau. Cette génération pouvant être :

- guidée par les données : ascendante ;
- par les buts : descendante ;
- décentralisée c'est-à-dire gérée par les Frames eux mêmes comme dans Schema System [Dra89]
- centralisée comme dans VISIONS.

Si les Frames fournissent un formalisme intéressant pour décrire les relations entre entités du système à un instant donné, il semble cependant qu'ils ne permettent pas de spécifier suffisamment la dynamicité (le contrôle de l'exécution) d'un système. En effet, comme le remarque Draper dans [Dra89, p. 219] les comportements d'itération et de test (c'est-à-dire le flux de contrôle) semblent difficile à appréhender. L'IAD fournit des modèles d'exécution comme les langages acteurs [Agh86] ou les SMA (chap. 5) rendant possible de résoudre en partie ce problème.

B.3 Les réseaux sémantiques

Les réseaux sémantiques fournissent une représentation graphique de la logique du premier ordre. Un tel réseau est un graphe étiqueté dont les sommets dénotent les concepts, et les arcs les relations entre ces concepts. Les étiquettes portées par les arcs vont refléter les différents types de relations (cf. §4.3.2) pouvant exister entre les concepts c'est-à-dire des relations de composition, de spécialisation de vue, etc.

[3]Les auteurs utilisent le terme de schéma qui est une structure analogue au Frame

Bibliographie

[Ada94] R. Adams and L. Bischof. Seeded region growing. *PAMI*, 16(6) :641–647, June 1994.

[Agh86] G. Agha. *Actors : A Model of Concurrent Computation in Distributed Systems*. MIT Press, Cambridge, Mass., 1986.

[Alo88] Y. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. *Int. Journal of computer Vision*, 2 :333–356, 1988.

[Alo90] Y. Aloimonos. Purposive and qualitative active vision. *ICPR*, 1 :346–360, 1990.

[Alo94] Y. Aloimonos. What i have learned : Reply. *CVGIP*, 60(1) :74–85, July 1994.

[And87] H.L. Anderson, R. Bajcsy, and M. Mintz. A modular feedback system for image segmentation. *UPenn*, 1987.

[Baj88] R. Bajcsy. Active perception. *Proc. of IEEE, Spécial issue on computer vision*, 76(8) :996–1005, August 1988.

[Bal91] D.H. Ballard. Animate vision. *Artificial Intelligence*, 48(1) :57–86, February 1991.

[Bec99] A. Becker and P. Naïm. *Les réseaux bayésiens ; Modèles graphiques de connaissance*. Eyrolles, 1999.

[Bel92] A. Belaïd and Y. Belaïd. *Reconnaissance des formes. Méthodes et applications*. InterEdition, 1992.

[Bel94] F. Bellet, M. Salotti, and C. Garbay. Low level vision as the opportunist scheduling of incremental edge and region detection processes. *ICPR94*, pages A :517–519, 1994.

[Bel98] F. Bellet. *Une Approche incrémentale à base de processus coopératifs et adaptatifs pour la segmentation des images en niveaux de gris*. Ph.D. thesis, Thèse de l'Université Joseph Fourier de Grenoble, 1998.

[Ber93] H. Bertin, E.and Bischof. Voronoi pyramids controlled by hopfields networks. Technical Report PRIP-TR-24, Institute f. Automation 1836/2, Dept. For Pattern Recognition and Image Processing, TU Wien, Austria, 1993.

[Ber94] E. Bertin. *Diagrammes de Voronoi 2D et 3D : application en analyse d'images*. Ph.D. thesis, Thèse de l'Université Joseph Fourier de Grenoble, 1994.

[Ber95] P. Bertolino. *Contribution des pyramides irrégulières en segmentation d'images multirésolution*. Ph.D. thesis, Thèse de l'Institut National Polytechnique de Grenoble, 1995.

[Ber96] P. Bertolino and A. Montanvert. Coopération région-contours multirésolution en segmentation d'images. In *In 10ème Congès RFIA Rennes, France*, pages 299–307. 1996.

[Ber99] Donato Bergandi. Réduire, c'est appauvrir ! l'idée d'émergence. *hors-série science et avenir*, pages 70–74, décembre 1999.

[Bev89] J.R. Beveridge, J.S. Griffith, R.R. Kohler, A.R. Hanson, and E.M. Riseman. Segmenting images using localized histograms and region merging. *IJCV*, 2(3) :311–352, January 1989.

[Boi93] O. Boissier. *Problème du contrôle dans un système intégré de vision ; utilisation d'un système Multi-Agents*. Ph.D. thesis, Thèse de l'Institut National Polytechnique de Grenoble, 1993.

[Boi94a] O. Boissier and Y. Demazeau. ASIC : An architecture for social and individual control and its application to computer vision. *In Perram, J.W. and Müller, J.P. Eds. Lecture Notes in Artificial Intelligence, 1069 (Proceedings of the MAAMAW Workshop, Odense, Denmark, August 1994)*, pages 135–149, 1994.

[Boi94b] O. Boissier and Y. Demazeau. MAVI : a multi agent system for visual integration. In *In Proc. of the IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, pages 731–738. Las Vegas, October 1994.

[Boi94c] O. Boissier, Y. Demazeau, Masini G., and H. Skaf. Une architecture multiagent pour l'implantation du bas niveau d'un système de compréhension de scènes. In *In JFIADSMA'94*. 1994.

[Boi97] O. Boissier and Y. Demazeau. Une architecture multi-agent pour les systèmes de vision ouverts et décentralisés. *Technique et science informatique*, 16(8) :1039–1062, 1997.

[Bou91] Bouman and Liu. Multiple resolution segmentation of textured images. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 13(2) :99–113, 1991.

[Bou99] A. Boucher. *Une approche décentralisée et adptative de la gestion d'informations en vision*. Ph.D. thesis, Thèse de l'Université Joseph Fourier de Grenoble, 1999.

[Bov90] A.C. Bovik, M. Clark, and W.S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 12(1) :55–73, 1990.

[Bov01] D.P. Bovet and M. Cesati. *Understanding the Linux Kernel*. O'Reilly, 2001.

[Boy00] K.L. Boyer and S. Sarkar, editors. *Perceptual Organization For Artificial Visions Systems*. Kluwer Academic Publishers, 2000.

[Bra88] M.E. Bratman, D.J. Israel, and M.E. Pollack. Plans and ressource-bounded practical reasonning. *Computationnal Intelligence*, 4 :349–355, 1988.

[Bra95] G. Braviano. *Logique floue en segmentation d'images : seuillage par entropie et structures pyramidales irrégulières*. Ph.D. thesis, Thèse de l'université Joseph Fourier - Grenoble, 1995.

[Bro86] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2 :14–23, 1986.

[Bro89] A. Brolio, B.A. Draper, R. Beveridge, and A.R. Hanson. ISR : A database of symbolic processing in computer vision. *IEEE Comp.*, 22(12) :22–30, 1989.

[Bro91] R.A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47, 1991.

[Bur81] P.J. Burt. Fast filter tranforms for image processing. *CVGIP*, 16 :20–51, 1981.

[Bur83] P.J. Burt. The laplacian pyramid as a compact image code. *IEEE Trans. on Communication*, COM-31 :532–540, 1983.

[Can83] J. Canny. Finding edges and lines in images. In *Technical Report, AI-TR-720, MIT*. 1983.

[Can86] J. Canny. A computational approach to edge detection. *PAMI*, 8(6) :679–698, November 1986.

[Cha91] D. Charlesbois, J.F. Deguise, S.M. Goodenough, S. Matwin, and M. Robson. A case-based planner to automate reuse of ES software for analysis of remote sensing data. *Digest-International Geoscience and Remote Sensing Symposium (IGARSS)*, 3(IEEE Cat. No. 91CH2971-0) :1851–1854, 1991.

[Cha94] Y.L. Chang and X.B. Li. Adaptive image region-growing. *IP*, 3(6) :868–872, November 1994.

[Che78] P.C. Chen and T. Pavlidis. Segmentation by texture using a co-occurrence matrix and a split-and-merge algorithm. In *ICPR78*, pages 565–569. 1978.

[Cho97] K.J. Cho and P. Meer. Image segmentation from consensus information. *CVIU*, 68(1) :72–89, October 1997.

[Chu93] C.C. Chu and J.K. Aggarwal. The integration of image segmentation maps using region and edge information. *PAMI*, 15(12) :1241–1252, December 1993.

[Cli01] Clips. URL `www.ghg.net/clips/CLIPS.html`, Accessed Oct. 2001.

[Clo99] R. Clouard, A. Elmoataz, C. Porquet, and M. Revenu. BORG : A knowledge-based system for automatic generation of image processing program. *IEEE trans. on Pattern Analysis and Machine Intelligence*, 21(2), 1999.

[Clé93] V. Clément and M. Thonnat. A knowledge-based approach to integration of image procedures processing. *CVGIP : Image Unterstanding*, 57(2), 1993.

[Coc95] J.-P. Cocquerez, S. Philipp, et al. *Analyse d'image : filtrage et segmentation*. Masson edition, 1995.

[Coh90] P.R. Cohen and H.J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42 :213–261, 1990.

[Cor94] T. Cormen, C. Leiserson, and R. Rivest. *Introduction à l'algorithmique*. Dunod, 1994.

[COR01] CORBA. `www.corba.org` & `www.omg.org`, Accessed Oct. 2001.

[Cot00] F. Cotter, J. Delacroix, C. Kaiser, and Mammeri Z. *Ordonnancement temps réel*. Hermes, 2000.

[Cov67] T.M. Cover and P.E. Hart. Nearest neighbor pattern classification. *IT*, 13(1) :21–27, January 1967.

[Cre97] D. Crevier and R. Lepage. Knowledge-based understanding systems : A survey. *Computer Vision and Image Understanding*, 67(2) :161–185, 1997.

[Cro89] J.L Crowley et al. Vision as a process. Technical report, ESPRIT Basic Research Action 3038, Aalborg, 1989.

[CVG94] CVGIP. A computationnal and evolutionary perspective on the role of représentation in vision. *CVGIP : Image Understanding*, 60(1) :65–118, 1994.

[Des37] René Descartes. Discours de la méthode. 1637.

[Did71] E. Diday. La méthode des nuées dynamiques. *Revue Statist. Appl. Ann. Math. Stat.*, 19(2) :19–34, 1971.

[Dor99] J.F. Dortier et al. *Le cerveau et la pensée, le révolution des sciences cognitives*. Sciences Humaines Éditions, 1999.

[Doy79] J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12(3) :231–272, 1979.

[Dra89] B.A. Draper, R.T. Collins, A. Brolio, A.R. Hanson, and Riseman E.M. The schema system. *International Journal of Computer Vision*, 2 :209–250, 1989.

[Dub87] D. Dubois and H. Prade. *Théorie des Possibilités. Application à la représentation des connaissances en informatique*. Masson, 1987.

[Dub89] R.C. Dubes and A.K. Jain. Random field models in image analysis. *AppStat*, 16(2) :131–164, 1989.

[Duc00] E. Duchesnay, J.J. Montois, and Y. Jacquelet. Résolution distribuée de conflits dans un réseau d'agents. *Neurosciences et sciences pour l'ingénieur*, 2000.

[Eli91] Norbert Elias. *La société des individus*. Fayard, 1991.

[Erm80] L.D. Erman, F. Hayes-Roth, V.R. Lesser, and Reddy D.R. The HERSAY-II speech understanding system : Integrating knowledge to uncertainty. *Computing Survey*, 12(2), 1980.

[Fer95] Jacques Ferber. *Les systèmes multi-agents. Vers une intelligence collective*. InterEditions, 1995.

[Fin93] T. Finin et al. *DRAFT, Specification of the KQML Agent-Communication Language*. The DARPA Knowledge Sharing Initiative, 1993.

[For82] C.L. Forgy. RETE : A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence journal*, 19 :17–37, 1982.

[Gar01] G. Gardarin. *Bases de données*. Eyrolles, 3 edition, 2001.

[Gei94] A. Geist et al. *PVM : Parallel Virtual Machine A Users' Guide and Tutorial for Networked Parallel Computing*. MIT Press, 1994.

[Gei99] J.M. Geib, C. Gransart, and P. Merle. *Corba. Des concepts à la pratique*. Dunod Masson, 1999.

[Gem84] S. Geman and D. Geman. Stochastic relaxation, gibbs distribution by constrained optimization. *IEEE trans. on Pattern Analysis and Machine Intelligence*, 6 :191–204, 1984.

[Gen91] M.R. Geneserth and R.E. Fikes. *Knowledge Interchange Format Version 3.0*. Logic-92-1, Stanford University Logic Group, 1991.

[Ger99] L. Germond. *Trois principes de coopération pour la segmentation en imagerie de résonance magnétique cérébrale*. Ph.D. thesis, Thèse de l'Université Joseph Fourier de Grenoble, 1999.

[Gia89] J. Giarratano and G. Riley. *Expert Systems ; Principles and Programming*. PWS-KENT Publishing Company, 1989.

[Gra73] N. Gramenopoulos. Terrain type recognition using erts-1 mss image. In *Record of the Symposium on Significant Result Obtained from the Earth Ressources Technology Satellite*, NASA SP-327, pages 1229–41. 1973.

[Gut00] O. Gutknecht and J. Feber. Madkit : Une expérience d'architecture de plate-forme multi-agent générique. In *JFIADSMA'OO : 8èmes Journées Francophones d'Intelligence Artificielle et Systèmes Multi-Agents*, pages 223–236. St Jean-la-Vêtre, Loire, France, octobre 2000.

[Han78] A.R. Hanson and Riseman E.M. VISION : A computer system for interpreting scènes. *PAMI*, 6(5) :555–577, September 1978.

[Har68] P. Hart, N. Nilson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst., Sci., Cybern*, SCC-4 :100–107, 1968.

[Har81] R.M. Haralick and L.T. Watson. A facet model for image data. *CGIP*, 15 :113–129, 1981.

[Har85] R.M. Haralick and L.G. Shapiro. Image segmentation techniques. *CVGIP*, 29(1) :100–132, January 1985.

[Hat91] J.P. Haton et al. *Le raisonnement en intelligence artificielle. Modèles, techniques et architectures pour les systèmes à base de connaissance*. InterEditions, 1991.

[Hoc96] J. Hoc. *Supervision et contrôle de processus : la cognition en situation dynamique*. PUG, Grenoble, 1996.

[Hof85] D. Hofstadter. *Gödel Esher Bach : Les brins d'une guirlande éternelle*. InterEditions, 1985.

[Hor74] S.L. Horowitz and T. Pavlidis. Picture segmentation by a directed split and merge procedure. *ICPR74*, pages 424–433, 1974.

[Hor78] S.L. Horowitz and T. Pavlidis. A graph-theoretic approach to picture processing. *CGIP*, 7(2) :282–291, April 1978.

[HR85] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26(3) :251–321, 1985.

[HR88] B. Hayes-Roth and M. Hewett. *BB-1 : An Implementation of the Blackboard Control Architecture*. Blackboard Systems. Addison-Wesley, Reading, Massachusetts, 1988.

[Jes01] Jess. URL `herzberg.ca.sandia.gov/jess`, Accessed Oct. 2001.

[Jia93] H. Jiang, J. Toriwaki, and H. Suzuki. Comparative performance evaluation of segmentation methods based on region growing and division. *SCJ*, 24(13) :28–42, 1993.

[J.M94] Alliot J.M. and T. Schiex. *Intelligence Artificielle et Informatique Théorique*. Cépaduès-Éditions, 1994.

[Jol92] J.M. Jolion and A. Montanvert. The adaptative pyramid : A framework for 2d image analysis. *CVGIP :IU, 55*, pages 339–348, 1992.

[Jol94] J.M. Jolion. Computer vision methodologies. *CVGIP : Image Unterstanding*, 59(1) :53–71, 1994.

[Jol01] J-M. Jolion et al. *Les systèmes de vision*. Hermes, 2001.

[Kho87] C.A. D Khol, A.R. Hanson, and Riseman E.M. A goal-directed intermediate level executive for image interpretation. *Proc. 10th International Joint Conf. on Articial Intelligence*, pages 811–814, August 1987.

[Kle86a] J.d. Kleer. An assumption-based thruth maintenance system. *Artificial Intelligence*, 28 :127–162, 1986.

[Kle86b] J.d. Kleer. Extending the ATMS. *Artificial Intelligence*, 28 :163–196, 1986.

[Kle86c] J.d. Kleer. Problem solving with the the ATMS. *Artificial Intelligence*, 28 :197–224, 1986.

[Kro94] W. G. Kropatsch. Building irregular pyramids by dual graph contraction. Technical Report PRIP-TR-35, Institute f. Automation 1836/2, Dept. For Pattern Recognition and Image Processing, TU Wien, Austria, 1994.

[Kro95] W. G. Kropatsch. Equivalent contraction kernels and the domain of dual irregular pyramids. Technical Report PRIP-TR-42, Institute f. Automation 1836/2, Dept. For Pattern Recognition and Image Processing, TU Wien, Austria, 1995.

[Kun00] M. Kunt, G. Coray, G. Granlund, J.P. Haton, R. Ingold, and M. Kocher. *Reconnaissance des formes et analyse de scènes*. collection éléctricité ; traitement de l'information : vol 3. Presses polytechniques et universitaires romandes, 2000.

[Lam94] S.W.C. Lam and H.H.S. Ip. Structural texture segmentation using irregular pyramid. *PRL*, 15(7) :691–698, July 1994.

[Leb95] L. Lebart, A. Morineau, and M. Piron. *Statistique exploratoire multidimensionnelle*. Dunod, 1995.

[Lew75] Lewes. Problems of life and mind. 1875.

[Li97] C.-T. Li and R.G. Wilson. Textured image segmentation using multiresolution markov random fields and a two-component texture model. In *Proceedings of SCIA-10, Lappenranta*. 1997.

[Liu99] J. Liu and Y.Y. Tang. Adaptive image segmentation with distributed behavior-based agents. *IEEE trans. PAMI*, 21(6) :544–551, 1999.

[Low85] D. Lowe, editor. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, 1985.

[Lum83] R. Lumia, R.M. Haralick, O.A. Zuniga, L.G. Shapiro, T.C. Pong, and F.P. Wang. Texture analysis of aerial photographs. *PR*, 16(1) :39–46, January 1983.

[Lâa87] H. Lâasri, B. Maître, and J.P. Haton. ATOME : Outil d'aide au développement de systèmes multi-experts. In *In 6ème Congès AFCET/RFIA Antibes, France*, pages 749–759. 1987.

[Mal89] S. Mallat. A theory of multiresolution signal decomposition : The wavelet representation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 11 :674–693, 1989.

[Mar72] A. Martelli. Edge detection using heuristic methods. *CGIP*, 1(2) :169–182, August 1972.

[Mar76] A. Martelli. An application of heuristic search methods to edge and contour detection. *CACM*, 19(2) :73–83, February 1976.

[Mar82] D. Marr. *Vision*. W. H. and Company, New York, 1982.

[Mat89] T. Matsuyama. Expert systems for image processing : Knowledge-based composition of image analysis process. *Computer Vision and Image Processing*, 48, 1989.

[Mat90] T. Matsuyama and V. Hwang. *SIGMA : a Knwoledge-Based Aerial Image Understanding System*. Advances in Computer Vision and Machine Intelligence. Plenum New York, 1990.

[Mee89] P. Meer. Stochastic image pyramids. *CVGIP, 45*, pages 269–294, 1989.

[Min75] M. Minsky. A framework for representing knowledge. *in The Psychology of Computer Vision(P.H. Winston, Ed.) McGraw-Hill*, pages 211–281, 1975.

[Min88] Marvin Minsky. *La société de l'esprit*. InterEditions, 1988.

[Mir90] D.P. Miranker. *TREAT : A New and Efficiant Match Algorithm for AI Production System*. Pitman Londres, 1990.

[Mon91] A. Montanvert, P. Meer, and A. Rosenfeld. Hierarchical image analysis using irregular tessellations. *PAMI*, 13(4) :307–316, April 1991.

[Mon94] A. Montanvert, P. Meer, and P. Bertolino. Hierarchical shape analysis in grey-level images. In *MDSG94*, pages 511–524. 1994.

[Mér86] A. Mérigot, P. Clermont, F. Devos, and B. Zavidovique. A pyramidal system for image processing. In V. Cantoni and Levialdi S., editors, *Pyramidal systems for image processing and computer vision*, NATO ASI, pages 109–124. Srpinger Verlag, 1986.

[Nak79] Y. Nakagawa and A. Rosenfeld. Some experiments on variable thresholding. *Pattern recognition*, 11 :191–204, 1979.

[Naz84] A.M. Nazif and M.D. Levine. Low level image segmentation : An expert system. *PAMI*, 6(5) :555–577, September 1984.

[Oak99] S. Oaks and H. Wong. *Java Threads*. O'Reilly, 2 edition, 1999.

[Pan01] Pandore. URL `www.greyc.ismra.fr/~regis/Pandore`, Accessed Oct. 2001.

[Par62] E. Parzen. On the estimation of a probability density function and mode. *Ann. Math. Stat.*, 33 :1075–1076, 1962.

[Pav77] T. Pavlidis. *Structural Pattern Recognition*. Springer, 1977.

[Pav90] T. Pavlidis and Y.T. Liow. Integrating region growing and edge detection. *PAMI*, 12(3) :225–233, March 1990.

[Pav92] T. Pavlidis. Why progress in machine vision is so slow. *PRL*, 13(4) :221–225, 1992.

[Pit93] I. Pitas. *Digital Image Processing Algorithms*. Prentice Hall, 1993.

[Pro90] G.M. Provan. The application of dempster-shafer theory to a logic-based visual recognition system. *In Uncertainty in Artificial Intelligence 5 (M. Henrion et al, Eds)*, pages 389–405, 1990.

[Rao88] A.R. Rao and R. Jain. Konwledge representation and control in computer vision system. *IEEE Expert*, 3(3) :64–79, 1988.

[Rao91] A. S. Rao and M. P. Georgeff. Modelling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proc. of Knowlege Representation and Reasonning (KR&K91)*, pages 473–484. Morgan Kaufman, 1991.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13 :81–131, 1980.

[Ros76] A. Rosenfeld et al. Scene labeling by relaxation operation. *IEEE trans. on System, Man and Cybernetics*, pages 420–433, 1976.

[Ros82] A. Rosenfeld and A. C. Kak. *Digital Picture Processing*. Academic Press, New York, 1982.

[Rus95] S. Russel and Norvig P. *Artificial Intelligence ; A Modern Approach*. Prentice-Hall Internationnal, 1995.

[Sah88] P.K. Sahoo et al. A survey of thresholding techniques. *CVGIP*, 41(2) :233–260, 1988.

[Sal94] J.-M. Salotti. *Gestion des informations dans les premières étapes de la vision par ordinateur*. Ph.D. thesis, Thèse de l'Université Joseph Fourier de Grenoble, 1994.

[San95] F. Sandakly. *Contribution à la mise en oeuvre d'une architecture à base de connaissances pour l'interprétation de scènes 2D et 3D*. Ph.D. thesis, Thèse de l'université de Nice-Sophia Antipolis, 1995.

[Sha76] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[Son95] M. Sonka, M.D. Winniford, and S.M. Collins. Robust simultaneous detection of coronary borders in complex images. *IEEE Trans. on Medical Imaging*, 14(1) :151–161, 1995.

[Son99] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis and Machine Vision*. ITP, 1999.

[Sue87] P. Suetens and A. Oosterlinck. Using expert system for image understanding. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 1(2) :237–250, 1987.

[Tan75] S.L. Tanimoto and T. Pavlidis. A hierarchical data structure for picture processing. *CGIP*, 4(2) :104–113, June 1975.

[Tan94] A. Tanenbaum. *Systèmes d'exploitation*. Dunod/Prentice Hall, 3 edition, 1994.

[Tar94] M.J. Tarr and M.J. Black. A computational and evolutionary perspective on the role of representation in vision. *CVGIP : Image Unterstanding*, 60(1) :65–73, July 1994.

[Ten77] J.M. Tenenbaum and H.G. Barrow. Experiments in interpretation-guided segmentation. *Artificial Intelligence*, 8 :241–274, 1977.

[Ter83] A. Terry. The CRYSALIS project : Hierarchical control of production systems. Technical report, Stanford University, 1983.

[The98] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, 1998.

[Thi97] S. Thiria, Y. Lechevallier, O. Gascuel, and S. Canu. *Statistique et méthodes neuronales*. Dunod, 1997.

[TL92] F. Thomson Leighton. *Introduction aux algorithmes et architectures parallèles*. Morgan Kaufman, 1992.

[Tom88] R. Tomassone, M. Danzart, J.J. Daudin, and J.P. Masson. *Discrimination et classement*. Masson, 1988.

[Tso94] J.K. Tsotsos. REPLY there is no one way to look at vision. *CVGIP : Image Unterstanding*, 60(1) :95–97, 1994.

[Uns89] M. Unser and M. Eden. Multiresolution feature extraction and selection for texture segmentation. *PAMI*, 11(7) :717–728, July 1989.

[Var98] Francisco Varela. Le cerveau n'est pas un ordinateur. c'est de l'activité permanente du corps qu'émerge le sens de son monde. *La recheche*, pages 109–113, avril 1998.

[Wes76] J.S. Weszka, C.R. Dyer, and A. Rosenfeld. A conmparative study of texture measures for terrain classification. *IEEE Transaction on Systems, Man, and Cybernetics*, 6 :269–285, 1976.

[Wil94] D. Willersinn. Parallel graph contraction for dual irregular pyramids. Technical Report PRIP-TR-28, Institute f. Automation 1836/2, Dept. For Pattern Recognition and Image Processing, TU Wien, Austria, 1994.

[Wro87] B. Wrobel and O. Monga. Segmentation d'images naturelles : coopération entre un détecteur-contours et un détecteur-régions. *In 11ème Colloque "Traitement du signal et des images", GRETSI, Nice France*, pages 539–542, 1987.

[Xia92] Y. Xiaohan and J. Ylä-Jääski. Image segmentation combining region growing and edge detection. *In Proc. 11th International Conference on Pattern Recognition*, 3 :481–484, 1992.

[Yan98] K. Yanai and K. Deguchi. An architecture of object recognition system for various images based on multi-agent. In *International Conference on Pattern Recognition*, pages 278–281. 1998.

[Zad65] L. Zadeh. Fuzzy sets. *Information and Control*, pages 338–353, 1965.

[Zad84] L.A. Zadeh. Review of books : A mathematical theory of evidence. *The AI Magazine*, pages 81–83, 1984.

Résumé

Les agents situés dans l'image fournissent un cadre privilégié pour la mise en oeuvre de stratégies coopératives et localement adaptées en segmentation d'image. Ils facilitent l'intégration des connaissances *a priori*, expressions d'un modèle, permettant ainsi de dégager de nouvelles contraintes indispensables à toutes les étapes de la vision par ordinateur (de la segmentation à l'interprétation).

Si les SMA fournissent un nouveau type d'architecture logicielle adaptée aux besoins identifiés ci-dessus, ils requièrent de nouvelles méthodologies de description et d'analyse des systèmes d'information. Ainsi, nous proposons un cadre conceptuel pour l'architecture logicielle d'un système de vision bas-niveau basée sur des agents situés dans l'image. Une telle architecture est articulée en trois niveaux d'analyse et de description :

1. Description *globale* et *structurelle* de **l'organisation** regroupant les agents. Cette étape de description s'attache à établir les liens entre agents. Nous proposons comme élément organisationnel la **pyramide irrégulière** qui impose sa structure à la population d'agents, afin de garantir un comportement globalement contrôlable et convergent de ces derniers.

2. Description *locale*, *fonctionnelle* et *comportementale* des agents composant le système. Nous proposons une mise en œuvre particulière de l'architecture logicielle de vision bas-niveau. Dans cette dernière, deux familles d'agents, qui traduisent des primitives région et contour, interagissent localement au sein de la pyramide.

Notre objectif est de montrer comment cette méthodologie permet une implémentation riche, flexible et distribuée des aspects précédemment identifiés ; à savoir : l'adaptation locale, l'intégration et l'expression d'incertitudes dans l'information *a priori* et des traitements coopératifs région/région et région/contour.

3. Finalement, une analyse *globale*, *comparative* et *fonctionnelle* vérifie que l'ensemble des interactions locales produit une bonne segmentation des images. Nous comparons notre approche avec d'autres méthodes de segmentation sur des images médicales et des images de synthèse.