République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire pour l'obtention
d'un diplôme d'ingénieur d'état en informatique.
Option : Intelligence Artificielle

Sujet :

Les Apports des Systèmes Multi-Agents en Segmentation d'image

Présenté par: BOUZNADA Lotfi
TOURI Tarik

Promoteur : Mlle BENBLIDIA N
Mlle REGUIEG F.Z

Soutenue le: 22/10/2007 **, devant le jury composé de :**

Président : Melle BOUSTIA

Examinatrice : Mme AOUSSAT

Examinateur : Mr AIDJA

- 2006-2007-

Remerciements

Nous remercions tout d'abord **ALLAH** qui nous a accordé la force et la santé pour mener a bien ce travail.

Nous tenons à remercier :

Notre promotrice Mlle **BENBLIDIA.N** et Mlle **REGUIEG F.Z** pour leurs conseils fructueux et enrichissant, leur suivi permanent, leur aide et la pertinence de leur remarques et suggestions pour mener à terme ce travail.

Nos parents qui ont tellement sacrifié pour nous, pour leurs soutiens et leur compréhension.

Nos remerciements vont envers les membres de jury qui nous honorer pour apprécier notre travail en acceptant de juger notre travail.

Nos enseignants de la FAC et plus précisément ceux du département d'Informatique.

Enfin, nous remercions tous ceux qui, de près ou de loin, chacun a sa manière, ont contribué a l'élaboration de ce modeste travail

RESUME

Ce travail présente une approche coopérative de segmentation d'images basée sur une architecture multi-agents. Cette architecture a pour objectif la modélisation et la simulation de systèmes basés sur des entités autonomes en interaction. Initialement, l'image est décomposer en imagettes à l'intérieur de chacune d'elle mis en œuvre des agents d'interprétation locaux, qui coordonnent et échangent des informations afin de réalisé une segmentation optimale. Le critère d'homogénéité utilisé permet de réaliser une coopération région-région et région-contour.

L'approche proposée a été implémentée avec le langage de programmation java. Les résultats obtenus sont satisfaisant.

Mots clés : Segmentation d'image, approche coopérative,
Système multi agents, analyse d'histogramme

TABLES DES MATIERES

Remerciements	
Dédicace	
Liste des figures	
Résumé	
1. Introduction générale	1
2. Problématique	1
3. Objectif du travail	2
4. présentation du mémoire	3
CHAPITRE 1 : INTRODUCTION AUX TECHNIQUES DE SEGMENTATION	
1. Introduction	4
2. Différentes techniques de segmentation	4
2.1. Segmentation par approche « frontière »	4
2.2. Segmentation par approche « région »	5
2.3. Segmentation par approche «globale de l'image»	7
2.4. Segmentation par approche «coopérative»	11
3. Conclusion	14
CHAPITRE 2 : Etude Des SYSTEMES MULTI-AGENTS	
1. Introduction	15
2. Historique des SMA	15
3. Généralités sur les systèmes multi-agents	16
3.1. Définition d'un agent	16
3.2. Déférents types d'agents	17
3.2.1. Les Agents réactifs	17
3.2.2. Les agents proactifs	18
3.2.3. Agents cognitifs	18
3.2.4. Les agents cognitifs, type BDI	19
3.3. Définition de système multi agents	19
3.4. Différents types de SMA	20
3.5. Structure d'un SMA	20
3.5.1. L'environnement	20
3.5.2. Les interactions	21
3.5.3. L'organisation	25
4. Conclusion	25
CHAPITRE 3 : APPROCHE PROPOSEE	
1. Introduction	26
2. Pré requis	26
3. Les agents d'SMA	26
3.1 Les agents de gestion du système	26

3.1.1 L'agent moniteur	27
3.1.2 Les agents de contrôle local	27
3.2 L'agent interface	
3.3 Les agents de segmentation	27
3.3.1. Les agents régions	27
33.2. les agents contours	27
4. Protocoles de coopération	28
5. Tableau noir	29
6. Comportement des agents de segmentation	29
7. Conclusion	33
CHAPITRE 4 : CONCEPTION DU SYSTEME	
Partie (1) Méthode de conception	
1. Introduction	34
2. Définition UML (Unified Modeling Language)	34
3. Eléments de modélisation	34
4. Diagrammes	35
4.1. Diagramme de cas d'utilisation	36
4.2. Diagramme de classes	38
4.3. Diagramme d'objet	40
4.4. Les Diagramme de séquence	41
4.5. Diagramme de collaboration	43
4.6. Diagramme d'état-transition	45
4.7. Diagramme d'activités	45
4.8. Diagramme de composants et de déploiement	46
5. Le processus de développement UP	47
5.1. Définition	47
5.2. intérêt de UP	47
5.3. Les caractéristiques de UP	47
6. CONCLUSION	48
Partie (2) Modélisation du système	
1. Introduction	49
2. Fonctionnement du système	49
3. Diagramme de cas d'utilisation et de séquence	50
4. Diagramme de classe	59
4.1. Explication diagramme de classe	60
4.2. Règles de gestion	60
5. Diagramme d'état	62
5.1. Diagramme d'état pour l'agent Moniteur	62
5.2. Diagramme d'état pour l'agent Local	63
5.3. Diagramme d'état pour l'agent Interface	64
5.4. Diagramme d'état pour l'agent Région	65
5.5. Diagramme d'état pour l'agent Contour	66
6. Diagramme d'activité	67

7. Diagramme des composants	68
8. Conclusion	69
CHAPITRE 4 : IMPLEMENTATION ET RESULTATS	
1. Introduction	70
2. Les outils de développement	70
3. Implémentation du système	71
4. Interface du système	78
5. Expérimentation et résultats	79
6. Conclusion	86
Conclusion générale	87
Bibliographies	

TABLES DES FIGURES	Page
Chapitre I	
PARTIE I	
Figure I.01 : correspond à un histogramme bimodal d'une image	08
Figure I.02 : Application de la méthode Ficher sur une image	09
Figure I.03 : Application de la méthode Otsu sur une image	11
Figure I.04 : La pyramide de graphe d'adjacence des régions d'après	14
Chapitre II	
Figure II.01: Représentation d'un Agent réactif	17
Figure II.02 : Représentation d'un Agent cognitif	18
Figure II.03 : interaction entre agents	21
Figure II.04 : Communication par partage d'informations	22
Figure II.05 : Communication par envoi de messages	23
Chapitre III	
Figure III.01 : Différents types de coopération proposés entre agents régions et agents contours.	28
Figure III.02 : Exemple de coopération région-contour et région-région pour la fusion des régions.	30
Figure III.03 : Les sept comportements des agents de segmentation	32
Chapitre IV	
Partie (1).	
Figure IV.01 : Regroupement de cas en paquetages	35
Figure IV.02 : Diagramme de cas d'utilisation modélisant une borne d'accès a une banque	36
Figure IV.03 : Relation entre cas dans un diagramme de cas d'utilisation	37
Figure IV.04 : Représentation d'une classe	38
Figure IV.05 : Représentation d'une relation binaire	38
Figure IV.06 : Représentation d'une relation d'héritage	39
Figure IV.07 : Représentation d'une relation de composition	39
Figure IV.08 : Représentation d'une relation d'agrégation	39
Figure IV.09 : Représentation d'une relation réflexive	40
Figure IV.10 : Représentation des objets et des liens	41
Figure IV .11 : Représentation de diagramme de séquence	42
Figure IV.12 : Représentation de diagramme de communication	43
Figure IV.13 : Première représentation d'un diagramme de collaboration	43
Figure IV.14 : Second représentation d'un diagramme de collaboration	44
Figure IV.15 : Un diagramme de séquence pour illustrer une collaboration	44

Figure IV.16 : Représentation d'un diagramme d'état transition simple	45
Figure IV.17 : Différents représentation d'un composant et interfaces	46
Figure IV.18 : Architecture en couches et vues de Krutchen (vues 4+1)	48
Partie (2).	
Figure IV.19 : Diagramme de cas d'utilisation globale du système	51
Figure IV.20 : Diagramme de cas d'utilisation pour la segmentation de l'image	52
Figure IV.21 : Diagramme de séquence pour L'initialisation de segmentation de l'image	53
Figure IV.22 : Diagramme de séquence pour le début de la segmentation	55
Figure IV.23 : Diagramme de séquence pour la finalisation de la segmentation	57
Figure IV.24 : Diagramme de cas d'utilisation pour la demande de l'état de segmentation	58
Figure IV.25 : Diagramme de séquence pour la demande de l'état de segmentation	59
Figure IV.26 : Diagramme de classe du système globale	60
Figure IV.27 : Diagramme d'état pour l'agent Moniteur	61
Figure IV.28 : Diagramme d'état pour l'agent Local	62
Figure IV.29 : Diagramme d'état pour l'agent Interface	63
Figure IV.30 : Diagramme d'état pour l'agent Région	64
Figure IV.31 : Diagramme d'état pour l'agent Contour	65
Figure IV.32 : Diagramme d'activité du système	67
Figure IV.33 : Diagramme des composants	68
Chapitre V	
Figure V.01 : Classe Otsu	72
Figure V.02 : Classe Dériche	73
Figure V.03 : Classe Agent Moniteur	74
Figure V.04 : Classe Agent Control local	75
Figure V.05 : Classe Agent Région	76
Figure V.06 : Classe Agent Contour	77
Figure V.07 : Classe Agent Interface	77
Figure V.08 : Interface principale	78
Figure V.09 : Les différents sous menus du menu fichier.	78
Figure V.10 : Les différents sous menus du menu histogramme	78
Figure V.11 : Les différents sous menus du menu filtrage	79
Figure V.12 : Les différents sous menus du menu de segmentation	79
Figure V.13 : Les différents sous menus du menu d'Outil.	79
Figure V.14: Application des méthodes de la segmentation sur l'image Test1	81
Figure V.15: Application des méthodes de la segmentation sur l'image Bureau	83
Figure V.16: Application des méthodes de la segmentation sur l'image IRM	84
Figure V.17 Image IRM cérébrale 128x128 décomposé en imagettes	85

CHAPITRE 1

Introduction aux

Techniques De Segmentation D'images

INTRODUCTION

Depuis très longtemps, les chercheurs ont été fascinés par la capacité du système de vision humaine à percevoir l'espace qui l'entoure sans aucune difficulté. Avec la naissance de machines de plus en plus puissantes et les progrès techniques au niveau du traitement automatique des images, une nouvelle discipline est apparue sous le nom : « vision par ordinateur ». Ce nouveau paradigme avait pour ambition de réaliser un système de vision artificielle, capable de reproduire certaines fonctionnalités de la vision humaine à travers l'analyse des images de la scène captée par un dispositif d'acquisition. Pour réaliser un tel système, il est indispensable de passer par la segmentation et l'identification d'attributs caractéristiques de la scène et des objets qui la composent.

PROBLÉMATIQUE

Ce projet aborde l'un des principaux problèmes de la vision assisté par ordinateur (VAO): la segmentation d'image. Celle-ci consiste à localiser dans une image, les régions (ensembles de pixels) appartenant à une même structure (objet ou scène imagée). Aujourd'hui, vue la simplicité de la définition de l'objectif de la segmentation d'images et la richesse des travaux effectués dans ce domaine, on pourrait penser que la segmentation d'images est un problème en grande partie résolu. Il n'en est rien, probablement à cause de la très grande diversité des besoins et des applications, mais également parce que des approches mathématiques (analyse multi résolution, champs de Markov, Logique flous, modèle déformable) permettent de nouveaux développements. L'augmentation de la puissance des ordinateurs, autorise également l'exploration de nouvelles approches et la mise en œuvre de techniques d'optimisation, autrefois trop couteuse en temps de calcule. Par conséquent, on se trouve devant un nombre considérable de méthodes **[Gon, 92]**, qu'il est difficile de classer en dehors du schéma régions versus frontières qui tend à disparaitre avec les approches coopératives.

La recherche d'une méthode performante, pour une application donnée, passe par la comparaison de quelques méthodes bien métrisées et par la modification d'une méthode existante afin de l'adapter suivant le contexte.

La méthode structurelle de résolution de problèmes consiste à décomposer le problème en sous problèmes et à définir dés le départ les étapes de résolution. Si auparavant cette méthode de résolution a donné de bons résultats, son efficacité a été remise en cause ces dernières années avec la complexité croissante des problèmes à résoudre et l'apparition de nouveaux besoins reflétant la nécessité de disposer de systèmes robustes et fiables dans des domaines dynamiques et incertains. Face à ces nouvelles difficultés, il est devenu nécessaire voire inévitable de chercher de nouvelles solutions en explorant de nouveaux paradigmes que ceux habituellement utilisés.

Dans le cadre de notre travail, nous étudiants la segmentation d'image d'un point de vue fonctionnel; Cette approche permet l'intégration de fonctions mathématiques.

OBJECTIF DU TRAVAIL

L'objectif de notre travail est de montrer qu'il est possible de définir un cadre commun de travail pour permettre la mise en oeuvre de coopération entre des approches hétérogènes. L'intérêt d'une telle approche est de pouvoir exploiter la complémentarité d'informations qui résultent de l'application de plusieurs méthodes afin de proposer un système complet de segmentation.

Il nous a été proposé, de développer un système multi agents composé d'un ensemble d'agents réactifs, afin de rendre la segmentation fortement distribuées, coopérative et bien guidée, notre système intègre des blocs fonctionnels à base de détecteurs de contours et région.

Ce système convergerait vers une solution optimale basé sur l'émergence d'un comportement global collectif et complexe à partir de simples interactions entre des agents simples dotés d'une intelligence très réduite et ne possédant qu'une vision très partielle de leur environnement. Ce comportement permettrait aux agents de résoudre collectivement des problèmes très complexes liés à la segmentation.

PRÉÉSENTATION DU MEMOIRE

Le plan de notre travail est structuré en cinq chapitres. Les deux premiers chapitres présentent une brève description des différents domaines auxquels nous avons touché:

Chapitre 1

Nous présentons dans ce chapitre un tour d'horizon des méthodes de segmentation d'images. Un intérêt particulier est porté pour les méthodes de segmentation en classes homogènes et sur les méthodes coopératives.

Chapitre 2

Nous présentons dans ce chapitre une étude des systèmes Multi-agents et leurs structures dans un cadre général.

Chapitre 3

Nous présentons la démarche à suivre et les différents composants de notre système multi-agents, le protocole de coopération ainsi que les comportements des agents seront décrits.

Chapitre 4

Nous nous intéressons ici a la méthode de conception en utilisant le langage de description UML et le processus de développement UP associé. Nous présenterons ensuite différents diagrammes qui modélisent notre système.

Chapitre 5

Nous présentons dans ce chapitre, l'implémentation de l'approche étudiée en utilisant le langage JAVA. Différents tests seront effectués sur des images de taille variable à fin de valider la robustesse de notre système.

I. INTRODUCTION

La segmentation est une étape primordiale en traitement d'image qui a pour but de rassembler des pixels entre eux suivant des critères prédéfinis. Les pixels sont ainsi regroupés en régions, qui constituent un pavage ou une partition de l'image. Il peut s'agir par exemple de séparer les objets du fond.

II. DIFFERENTES TECHNIQUES DE SEGMENTATION

Les techniques de segmentations existantes sont nombreuses mais elles sont généralement regroupées en quatre principales approches qui sont l'approche contour, l'approche région, l'approche classification et l'approche coopérative.

Les méthodes de l'approche contour consistent à identifier les discontinuités du niveau de gris qui séparent les régions. Les méthodes de l'approche région cherchent à regrouper des pixels qui présentent une similarité et une uniformité en niveau de gris. Ces deux approches sont duales l'une de l'autre.

Cependant, elles mènent à des algorithmes différents et ne donnent pas les mêmes résultats. Les méthodes de l'approche classification partitionnent l'image en classes homogènes en considérant les propriétés colorimétriques des pixels. Les informations spatiales peuvent ou non être pris en considération.

II.1. Segmentation par approche « frontières »

Cette approche repose sur la recherche des zones de transition entre deux régions connexes dans l'image **[Pav, 92]**.

Les méthodes les plus anciennes utilisent des opérateurs parallèles de type différentiel calculant la première dérivée partielle (le vecteur Gradient) définie par son module et sa direction, ou bien la deuxième dérivée partielle (le vecteur Laplacien). Une zone de transition dans le signal correspond à un maximum (ou un minimum) local de la dérivée première et un passage par zéro de la dérivée seconde.

L'identification d'une zone de transition du signal peut être faite par seuillage de la norme de sa dérivée première ou du passage par zéro de sa dérivée seconde.

Pour améliorer la qualité des contours et pallier aux problèmes de précision de localisation et d'efficacité de détection, sont apparus les opérateurs de dérivation avec filtrages optimaux.

Le filtre optimal est un dérivateur qui permet de détecter des contours en respectant les 3 critères suivants **[Can, 86]**:

1. Une bonne détection : l'opérateur donne une réponse au voisinage d'un contour.

2. Une bonne localisation : optimisation de la précision avec laquelle le contour est détecté.

3. Unicité de la réponse : le contour doit provoquer une réponse unique de l'opérateur.

Plusieurs opérateurs optimaux sont apparus dans la littérature. Parmi eux on trouve le filtre de Canny et le filtre de Deriche.

Détecteur de contour Dériche

Deriche a proposé un filtre à réponse impulsionnelle infinie pour détecter le contour C d'une image. Ce filtre s'écrit sous la forme **[Ref, 06]** :

$$f(x) - K x e^{-\alpha|x|} \quad avec \quad K - - \frac{(1 \quad e^{-\alpha})^2}{e^{-\alpha}}$$

Le paramètre α de Deriche représente alors l'inverse de l'écart type σ de la gaussienne du filtrage de Canny ($\alpha = \frac{\sqrt{\pi}}{\sigma}$).

Le calcul des contours se fait alors récursivement, avec des conditions initiales nulles aux bords de l'image **[Ref, 06]**.

On va construire les contours droit (C-) et gauche (C+) de notre image I(x) :

$$\begin{cases} C_+(x) = k^2 C_+(x-2) + 2kC_+(x-1) + I(x-1) \\ C_-(x) = k^2 C_-(x+2) + 2kC_-(x+1) + I(x+1) \end{cases}$$

$$avec\ k = e^{-\alpha}$$

$$C(x) = C_+(x) - C_-(x)$$

On obtient alors le contour C(x) de l'image.

II.2. Segmentation par approche « région »

Les méthodes appartenant à cette famille manipulent directement des régions. La segmentation en régions consiste à décomposer l'image en des régions homogènes, Une région est composée de l'ensemble des pixels connexes possédant les mêmes propriétés au sens d'un prédicat d'homogénéité donné [Gon, 92].

Il est courant de définir la segmentation d'une image **I** en terme d'un prédicat d'homogénéité **P** et d'un ensemble de "**n**" régions **R** vérifiant les critères suivants **[Zuc, 76]**:

1. $I = \bigcup_i R_i$;
2. $R_i \neq \phi \quad \forall i$;
3. $R_i \bigcap R_j \quad \forall i \neq j$;
4. R est connexe $\forall$ i ;
5. P(R_i)= vrai $\forall$ i ;
6. P($R_i \bigcup R_j$)=faux $\forall$ i ≠ j et R_i et R_j sont adjacents.

On distingue dans ces classe trois grandes familles de techniques **[Oua, 06]** :

• les méthodes de fusion de régions qui procèdent par agrégation itérative des pixels (ou de régions). L'algorithme s'arrête quand une segmentation optimale est atteinte au sens d'un prédicat d'homogénéité,

• les méthodes de division qui procèdent par division des régions de base en régions plus petites et de plus en plus homogènes. La division s'arrête quand toutes les régions produites vérifient un certain critère d'homogénéité,

• les méthodes de division/fusion qui combinent les deux méthodes précédentes.

II.3. Segmentation par approche globale de l'image

On part ici d'un rapport qu'entretient chaque pixel individuellement avec des informations calculées sur toute l'image, comme par exemple la moyenne des niveaux de gris de l'ensemble des pixels, ou la médiane, permettant de segmenter l'image en 2 régions comportant le même nombre de pixels. Ces informations permettent de construire des classes de pixels. Les pixels appartenant à une même classe étant connexes constituent alors des régions.

Il existe de nombreuse méthodes de segmentation dans l'approche globale; nous allons citer quelque méthodes basés sur l'analyse de l'histogramme **[Ker, 97]**.

II.3.1.Le seuillage

Cette opération segmente une image en plusieurs classes en n'utilisant que l'histogramme. A chaque pic de l'histogramme est associée une classe. Chaque classe est caractérisée par sa distribution de niveaux de gris.

Il existe plusieurs méthodes de seuillage d'un histogramme. Elles sont adaptées à des histogrammes avec des pics séparés.

a)Le Seuillage manuel

Le seuillage manuel consiste à choisir un seuil arbitraire (i : niveau de gris) (ou plusieurs seuils dans le cas d'un histogramme multimodal).

b) Le seuillage automatique

Le seuillage dynamique consiste à déterminer automatiquement le seuil i qui sépare le fond de la forme.

Le but est de chercher la valeur du seuil i correspondant à la vallée entre les 2 pics (**Figure I.01**).

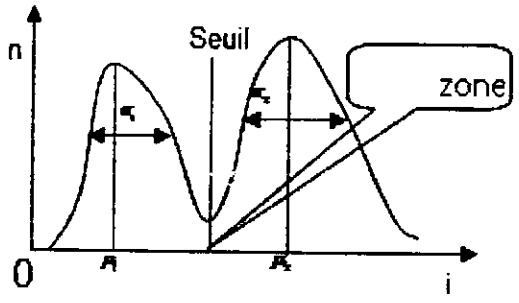


Figure I.01 : correspond à un histogramme bimodal d'une image.

Plusieurs méthodes ont été développée pour la détection automatique de ce seuil, parmi les méthodes les plus utilisées, nous pourrons cites celle de Ficher et celle d'Otsu.

b.1) La méthode de Fisher

La méthode de Fisher est une méthode permettant de découper un histogramme en un certain nombre de classes. Ici, la méthode sera étudiée pour deux classes.

La méthode de Fisher consiste modéliser l'histogramme bimodal en niveaux de gris d'une image par une somme pondérée de distributions gaussiennes et à localiser les seuils comme les séparateurs des distributions [Ker, 95]

Ceci est réalisé à l'aide d'un critère de minimisation de la somme des inerties associées aux différentes classes. Dans le cas où le problème se réduit à la différentiation de deux classes (C1 et C2), il s'agit de trouver leurs bornes afin de minimiser l'inertie I donnée par :

$$I = \sum_{k \in C1} h(k) \times (k - G(C_1))^2 + \sum_{k \in C_2} h(k) \times (k - G(C_2))^2$$

Où:

- G : centre de gravité de la classe
- C1, C2 : classes de niveaux de gris
- k : niveau de gris
- h : poids du niveau de gris dans l'histogramme

Un développement de cette expression permet de formuler le problème en termes de maximisation d'une quantité J :

$$J(p) = \frac{(\sum_{k \in C_1} k \times h(k))^2}{\sum_{k \in C_1} h(k)} + \frac{(\sum_{k \in C_2} k \times h(k))^2}{\sum_{k \in C_2} h(k)}$$

La solution, seuil recherché permettant de distinguer les deux classes, est donc le niveau de gris tel que J soit maximum (Figure I.02 c).

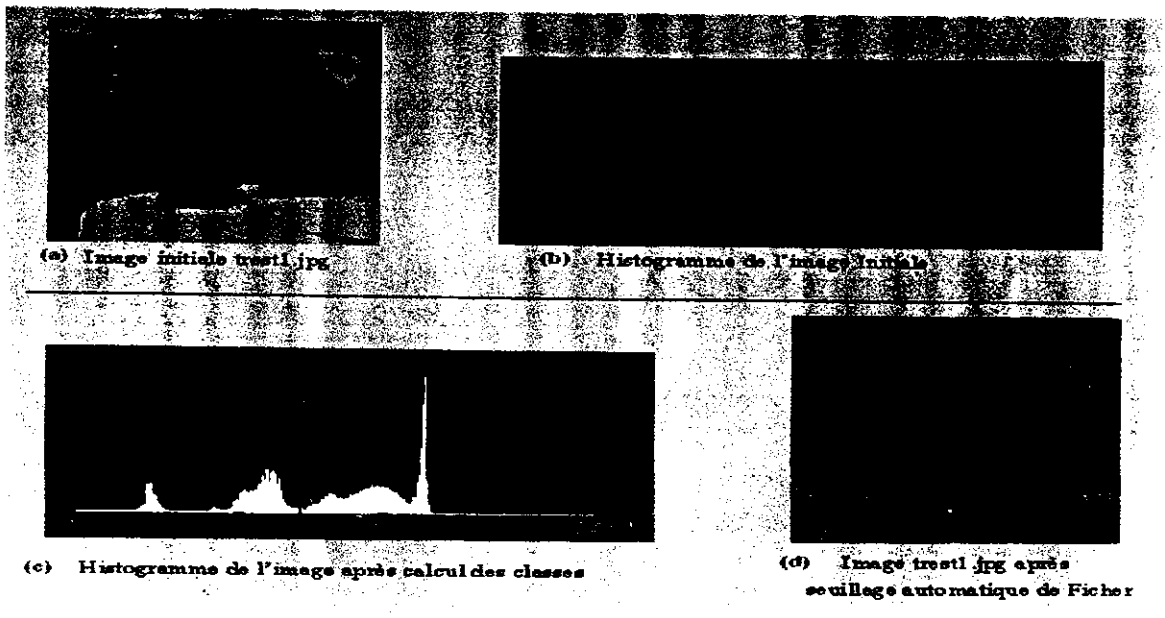


Figure I.02 : Application de la méthode Ficher sur une image

b.2) La méthode d'Otsu

Otsu formule le problème comme une analyse discriminante, pour laquelle il utilise une fonction critère particulière comme mesure de séparation statistique **[Che, 98]**. Des statistiques sont calculées pour les deux classes de valeurs d'intensité séparées par un seuil. Dans le cadre de la binarisation par la méthode d'Otsu, la séparation se fait à partir des moments des deux premiers ordres à savoir la moyenne et la variance. Pour que le procédé soit indépendant du nombre de points dans l'image N, on normalise l'histogramme.

La fonction critère η est donnée par le rapport entre la variance inter classe et la variance total:

$$\eta\ (P) = \frac{\sigma_{\beta}^{2}}{\sigma_{T}^{2}}$$

Où :

- σ_{β}^{2}: la variance inter-classe donnée par : $\sigma_{\beta}^{2} = \omega_0\omega_1(\mu_0\mu_1)$;
- σ_T^2: la variance total donnée par : $\sigma_T^2 = \sum_{i=1}^{l-1}(i-\mu_T)^2 P_i$;
- P_i : l'histogramme normalisé donnée par : $P_i = \frac{ni}{n_t}$;
- ω_0: la variance de la classe C1 donnée par : $\omega_0 = \sum_{i=1}^{t} P_i$;
- ω_1: la variance de la classe C2 donnée par : $\omega_1 = 1-\omega_0$;
- μ_0: la moyenne de la classe C1 donnée par : $\mu_0 = \frac{\mu_t}{\omega_0}$;
- μ_1: la moyenne de la classe C2 donnée par : $\mu_1 = \frac{\mu_T - \mu_t}{1-\mu_0}$;
- μ_T: la moyenne totale donnée par : $\mu_T = \sum_{i=1}^{l-1} i \times P_i$;
- μt : la moyenne donnée par : $\mu_t = \sum_{i=1}^{t} i \times P_i$.

Le niveau qui maximise la fonction critère est choisi comme seuil. Ainsi la valeur du seuil est obtenue pour i tel que le critère à maximiser (Figure I.03 c).

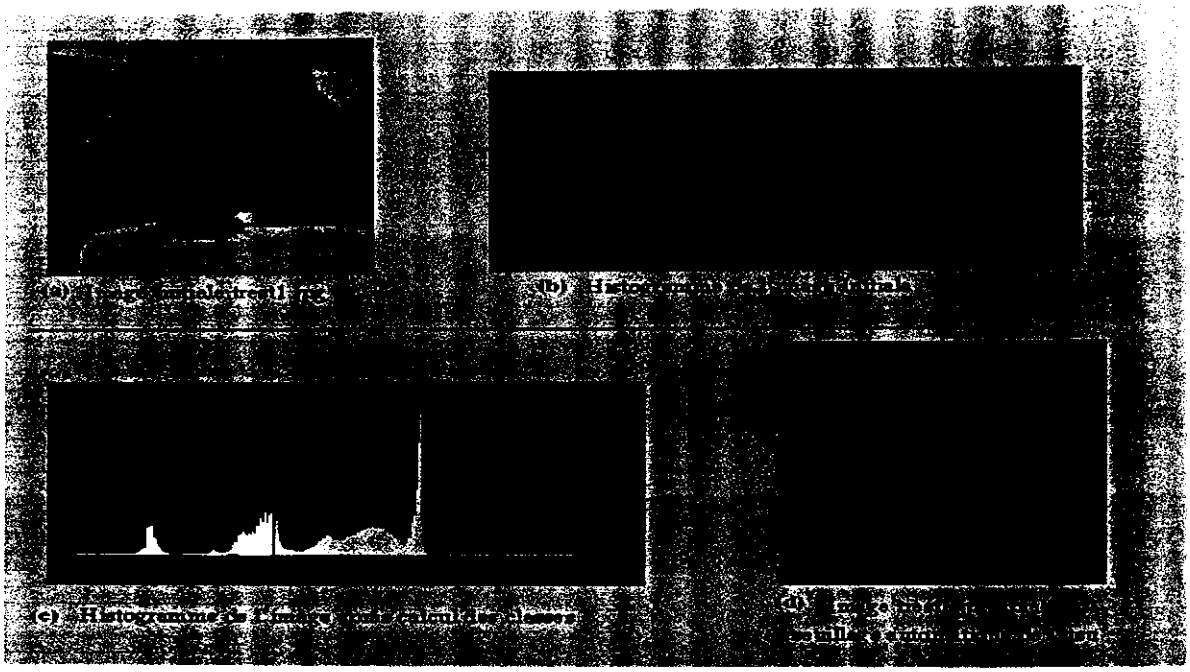


Figure I.03 : Application de la méthode Otsu sur une image

L'Extension de la méthode d'Otsu à plus de 2 classes est possible. Les détails seront donnés dans (chapitre 5).

II.4. Segmentation par approche coopérative

Le but de cette approche est de définir un cadre commun de travail pour permettre la mise en œuvre de la coopération entre des approches hétérogènes. L'intérêt d'une telle approche est de pouvoir exploiter la complémentarité d'informations qui résultent de l'application de plusieurs méthodes afin de proposer un système complet de segmentation.

L'instauration d'une coopération contour-région pour optimiser la segmentation n'est pas une préoccupation récente. Des règles de fusion de régions, ont déjà été proposées dans **[Naz, 84]**. Il s'agit alors d'un véritable système expert de segmentation; la coopération s'exprime par l'intermédiaire des variables des règles de production, qui peuvent par exemple s'instancier avec des couples de régions à fusionner. Un mécanisme de focalisation de l'attention permet à tout instant, de considérer une région courante et un contour courant, sur lesquels on va tenter d'appliquer les règles de fusion, de division, de prolongement et de raccordement

Ces idées ont été reprises dans **[Amm, 95]**, en utilisant une architecture à base de tableau noir, pour pallier le manque d'efficacité des systèmes experts, où toutes les règles sont potentiellement en concurrence et tous les faits de la base de faits sont à plat. Les règles de fusion prennent alors la forme de sources de connaissances. La segmentation est essentiellement considérée comme un processus de croissance de régions dans lequel il s'agit de faire un usage éclairé et favorable de l'information contour.

II.4.1.Approches multi-agents en segmentation d'images

Il existe de multiples manières d'envisager la coopération dans les approches multi-agents. **[Gar, 01]** propose d'en distinguer trois formes :

- **La coopération confrontative** : une tâche est exécutée par plusieurs agents de spécialités différentes œuvrant de manière concurrente sur le même ensemble de données, le résultat étant obtenu par fusion.

- **La coopération augmentative** : une tâche est répartie sur une collection d'agents similaires œuvrant de manière concurrente sur des sous-ensembles disjoints de données, la solution étant obtenue sous forme d'un ensemble de solutions locales.

- **La coopération intégrative** : une tâche est décomposée en sous-tâches, accomplies par des agents de spécialités différentes et œuvrant de manière coordonnée, la solution étant obtenue au terme de leur exécution.

II.4.2.Exemples d'approches coopératives

Dans **[Bou, 98]**, les agents sont spécialisés dans la segmentation et l'interprétation d'images cytologiques. Un des aspects les plus intéressants concerne le traitement de séquences d'images et la manière de lancer des agents dans l'image suivante de la séquence, en fonction de ce qui est en train de se dérouler dans l'image courante.

Dans **[Liu, 99]**, la segmentation d'images est abordée sous l'angle des algorithmes génétiques : il s'agit de générer la population d'agents de segmentation la mieux adaptée à la distribution de points rencontrés dans l'image. Les agents sont dotés de deux types de comportement : la diffusion et la reproduction de manière à s'adapter au mieux aux variations locales de l'image.

Ces idées sont reprises dans **[Bur, 01]**, qui propose de combiner une approche par croissance de régions dans les zones faciles à segmenter et l'approche génétique dans les zones plus délicates. La coopération consiste ici à choisir automatiquement, en fonction de critères d'homogénéité locaux, laquelle des deux méthodes l'agent va appliquer.

Dans **[Ric, 01]** et **[Ric, 02]**, des agents sont situés dans l'image, ils coopèrent pour segmenter des IRM cérébrales. On y trouve diverses catégories d'agents : un agent de contrôle global, des agents de contrôle locaux et au niveau le plus bas, les agents de segmentation, spécialisés dans la détection des trois types de tissus cérébraux (matière blanche, matière grise et liquide céphalorachidien).Un processus de négociation entre régions dans les zones litigieuses est évoqué dans les perspectives, sans toutefois une solution soit apportée.

Toujours dans le domaine de la segmentation d'images, **[Duc, 01]**, s'appuie sur la structure de pyramide irrégulière (Figure I.04) pour gérer le processus de fusion de régions et assurer la convergence de la segmentation; une coopération région-région assez sophistiquée est mise en œuvre, mais qui ne tire pas suffisamment parti de l'information contour. Un des aspects intéressants de son approche est l'utilisation d'une procédure de décimation (récursive) pour le passage du niveau k au niveau $k+1$. La pyramide se construit en plan de la base qui représente l'image pré segmentée jusqu'au dernier niveau de la pyramide comportant le minimum d'information. Les niveaux de cette pyramide sont des graphes d'adjacence de régions.

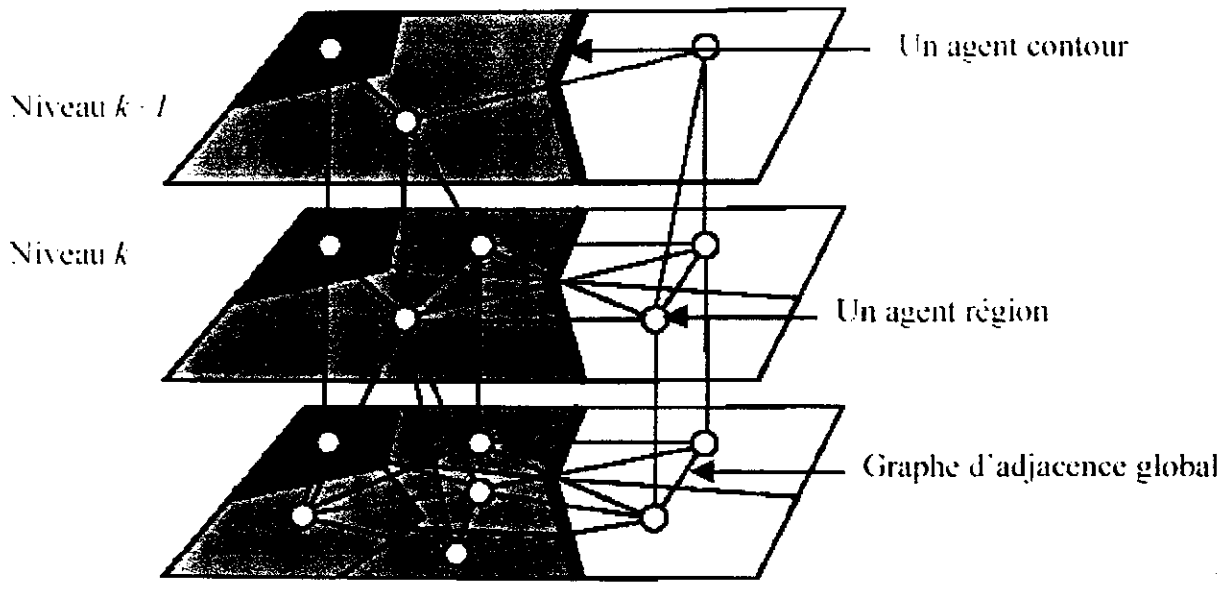


Figure I.04 : La pyramide de graphe d'adjacence des régions

La structure de pyramide irrégulière présente l'avantage d'assurer la convergence du processus de segmentation et nous avons repris certaines de ces idées dans le cadre de notre travail.

III. CONCLUSION

Comme nous venons de le voir, la coopération des agents peut prendre des formes très variées. Notre travail consiste a exploité les systèmes multi-agents dans un processus de segmentation coopératif ce que nous amène à définir est étudier les différents types de SMA dans le chapitre suivant

CHAPITRE 2

Etude des systèmes multi-agents

I.INTRODUCTION

Les systèmes multi-agents (SMA) trouvent actuellement des applications dans les domaines les plus divers. L'approche est en effet tentante, et on ne mesure pas encore très bien les avantages et les inconvénients.

II. HISTORIQUE

En 1980, un groupe de chercheurs s'est réuni pour discuter des défis concernant la résolution « intelligente » de problèmes dans un système comportant plusieurs résolveurs de problèmes. Lors de cette réunion, il a été décidé que l'intelligence artificielle distribuée n'axerait pas de ses travaux sur les délais de bas niveau de paralillisation, ni sur la manière de paralléliser les algorithmes centralisés, mais plutôt sur le fait de savoir comment un groupe de solutionneurs de problèmes pourrait coordonner ses efforts afin de résoudre des problèmes efficacement.

On peut dire que les SMA ont vu le jour avec l'avènement de l'intelligence artificielle distribuée (IAD). A ses débuts toutefois, L'IAD ne s'intéressait qu'à la coopération entre résolveurs de problèmes. On divisait en général un problème en sous problèmes et on allouait ces sous problèmes à différents résolveurs qui devaient coopérer pour élaborer des solutions partielles **[Bri, 01]**. Celles-ci sont finalement synthétisées en une réponse globale au problème initial. Ainsi, l'IAD, au départ, privilégiait le problème à résoudre tout en mettant l'accent sur la résolution d'un tel problème par multiples entités intelligentes. Dans les SMA d'aujourd'hui, les agents sont (entre autres) autonomes, possiblement préexistants et généralement hétérogènes. Dans ce cas, l'accent est plutôt mis sur le fait de savoir comment les agents vont coordonner leurs connaissances, buts et plans pour agir et résoudre des problèmes **[Bri, 01].**

III.GENERALITES SUR LES SYSTEMES MULTI-AGENTS

III.1. Définition d'un agent

Selon les auteurs, un agent peut être défini de différentes manières :

Définition une

«Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents» **[Fer, 95].**

Définition deux

« Un agent est un système informatique, situé dans un environnement, et qui agit d'une façon autonome et flexible pour atteindre les objectifs pour lesquels il été conçu » **[Jen, 98].**

Principales caractéristiques des agents

Il existe différentes sortes d'agent, selon qu'il possède l'une ou plusieurs des caractéristiques suivantes **[Jen, 98]**:

- **Situé** : l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement.
- **Autonome** : l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne.
- **Proactif** : l'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au bon moment.
- **Réactif** : l'agent doit être capable de générer une réponse lorsqu'un stimulus se présente.
- **Social** : l'agent doit être capable d'interagir avec les autres agents (logiciels ou humains) afin d'accomplir des tâches ou d'aider ces agents à accomplir les leurs.

Il existe plusieurs caractéristiques spécifiques au domaine d'utilisation des agents parmi les quelle nous citons

- **la coopération :** la capacité des agents à effectuer un travail collaboratif pour accomplir une tâche donnée.
- **l'adaptation :** un agent capable d'utiliser ses connaissances afin de modifier son comportement (augmentation de l'espace disque, détection d'un seuil,... etc.)
- **la communication :** la possibilité d'échanger des messages entre agents.
- **la reproductive :** capacité de se reproduire ou d'être reproduit.
- **la téléonomique** : comportement tendant à satisfaire un but, qui est défini par ses tendances.
- **le réflexe :** comportement ne faisant pas intervenir de facteurs internes (contrairement au comportement téléonomique).

III.2. Différents types d'agents

Nous distinguons plusieurs types d'agents dont:

III.2.1. Les Agents réactifs

Les agents réactifs n'ont pas de représentation ni d'eux mêmes ni de leur environnement. Leur comportement est régi par des réponses immédiates à des *stimuli* fournis par l'environnement et par leurs buts internes. Ils ne disposent pas de but explicite, ni de mémoire de son histoire, Ces agents sont les plus simples à réaliser et permettent notamment d'observer l'émergence de comportements collectifs **[Bri, 01].** La figure ci dessous représente un agent réactif.

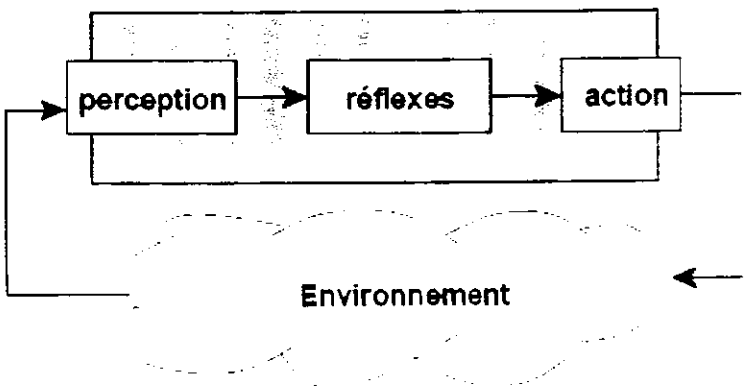


Figure II.01 : Représentation d'un Agent réactif

III.2.2.Les agents proactifs

Les agents proactifs sont des agents généralement associés à une compétence, les agents sont Chargés d'un but, et ils sont capables de planifier leurs actions à partir d'une situation présente. Ils nécessitent une représentation de l'environnement, et une communication avec les autres agents **[Bri, 01].**

III.2.3.Agents cognitifs

Les agents cognitifs ou délibératifs possèdent une représentation, plus ou moins limitée, d'eux-mêmes et de leur environnement, sur laquelle ils peuvent raisonner. Cette représentation regroupe l'ensemble des états mentaux de l'agent (Figure II.02).

Ils sont souvent capables de planifier leurs actions et permettent de simuler des comportements plus complexes. Ils peuvent tenir compte de leur passé et disposent d'un but explicite, Les Interactions entre agents s'établissent en fonction des collaborations nécessaires à la résolution du problème. La figure II.2 représente un Agent cognitif.

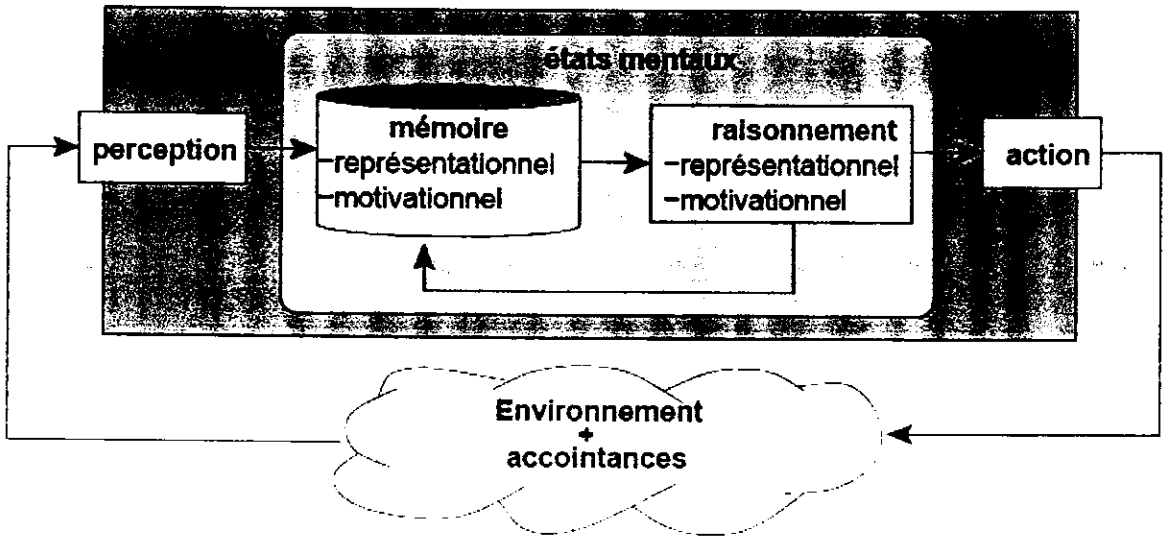


Figure II.02 : Représentation d'un Agent cognitif

Souvent, les agents ne sont ni purement réactifs ni purement cognitifs, mais situés à un niveau intermédiaire entre ces deux extrêmes.

III.2.4.Les agents cognitifs, type BDI (Believe, Desir, Intention)

Les agents BDI ont leurs propres croyances, désirs et intention **[Bra, 88]**.Cette architecture est basée sur le raisonnement pratique. Dans ce type d'architectures, les agents sont généralement représentés par un « état mental » ayant les attitudes mentales suivantes :

- les Croyances : sur le monde, sur les autres agents (leurs compétences, leurs croyances propres), et sur soit même,
- les Désirs : les états possibles envers lesquels l'agent peut vouloir s'engager,
- les Intentions : la volonté consciente d'effectuer un acte, et la planification de buts dans l'objectif d'atteindre la satisfaction du désir.

III.3. Définition de système multi agents

En parcourant la bibliographie, nous avons rencontré plusieurs définitions, nous proposons les définitions suivantes:

Définition une

Selon Ferber et Ghallab : «Un système multi-agents est une communauté d'agents travaillants en commun, selon des modes parfois complexes de coopération, conflit, concurrence, pour aboutir à un objectif global tel que la résolution d'un problème, rétablissement d'un diagnostic...» **[Fer, 95] .**

Définition deux

« Un SMA est un ensemble d'agents situés dans un certain environnement et interagissant selon une certaine organisation » **[Ref, 05].**

Définition trois

Un système multi-agents est un système distribué composé d'un ensemble d'agents, qui simulent dans une certaine mesure les capacités du raisonnement humain. Les SMA sont conçus et implantés idéalement comme un ensemble d'agents interagissant, le plus souvent, selon des modes de coopération, de concurrence ou de coexistence **[Bri, 01].**

III.4.Différents types d'SMA

Il existe cinq types de systèmes multi agents :

- **ouvert** : le nombre d'agents n'est pas fixe,
- **fermé** : l'ensemble d'agents reste le même,
- **homogène** : tous les agents sont construits sur le même modèle,
- **hétérogène** : des agents de modèles différents, de granularités différentes,
- **mixte (ou non)** : les agents « humains » sont partie intégrante du système.

III.5.Structure d'un SMA

La structure des systèmes multi agents regroupe trois composants principaux dont:

III.5.1 L'environnement

Un agent est situé dans un environnement (spatio-temporel), dynamique et évolutif. Cet environnement présente un espace partagé par tous les agents, il introduit des stimulations, des contraintes aux agents et un médium d'interactions.

Il existe 2 classes de contraintes, une géographiques (Accessibilité à certaines informations, connaissance partielle du monde à un instant, Possibilité d'interactions) et la deuxième temporelles.

Différents types d'environnements

On peut citer quatre types d'environnement :

a) Environnements continus

- ✓ disposent d'une métrique,
- ✓ location de l'agent de l'environnement.

b) Environnements discontinus (réseaux)

- ✓ le réseau peut être modifié,
- ✓ ancrage (présence) de l'agent dans son environnement,
- ✓ accès partiel d'un agent au contenue de l'environnement.

c) Environnements non structuré

- ✓ environnement contenant des agents et des objets et informations disponibles du système,
- ✓ abstrait,
- ✓ pas d'ancrage de l'agent,
- ✓ pas de modification possible de l'environnement.

d) Absence d'environnement

Pour certains SMA avec des agents qui communiquent en mode directe, il n'existe pas d'environnement.

III.5.2. Les interactions

Dans un Système Multi-Agents à partir de 3 agents (2 agents + un environnement), il est nécessaire de mettre en œuvre un mécanisme de communication entre les agents, pour prendre en charge les échanges de connaissances et la transmission de ses croyances ce qui facilite la coordination entre agents.

Cette communication de message se caractérise par (Figure II.03):

- le contenu du message,
- l'intention de l'expéditeur,
- les conséquences du message dans l'environnement et les agents receveurs,
- les agents impliqués dans la communication,
- son mode (direct ou indirect).

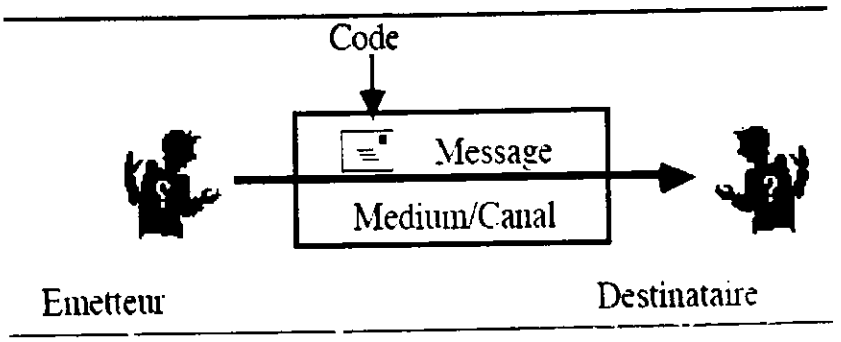


Figure II.03 : Interaction entre agents

Il existe trois types de communication :

III.5.2.1. La communication par partage d'informations :

Les agents utilisent une zone commune pour communiquer (Figure II.04). Cette zone contient les données initiales du problème et les conclusions tirées par les agents pendant la construction progressive de la solution globale **[Fer, 92]**. Elle peut être considérée comme une mémoire globale accessible à tous les agents, ou divisée en plusieurs niveaux d'abstraction, dont certains ne sont pas accessibles par l'agent. Cette zone est appelée tableau noir (blackboard), et les agents sont appelés sources de connaissances (knowledge sources) **[Fer, 95]** . Le tableau noir est le seul moyen de communication entre les agents, ce qui signifie que les agents ne se connaissent pas mutuellement **[Fer, 92]**. Ce modèle est efficace, en termes de coût de communication. Cependant, la gestion de la mémoire partagée et la résolution des conflits d'accès conduisent à des problèmes d'intégrité et de cohérence des données.

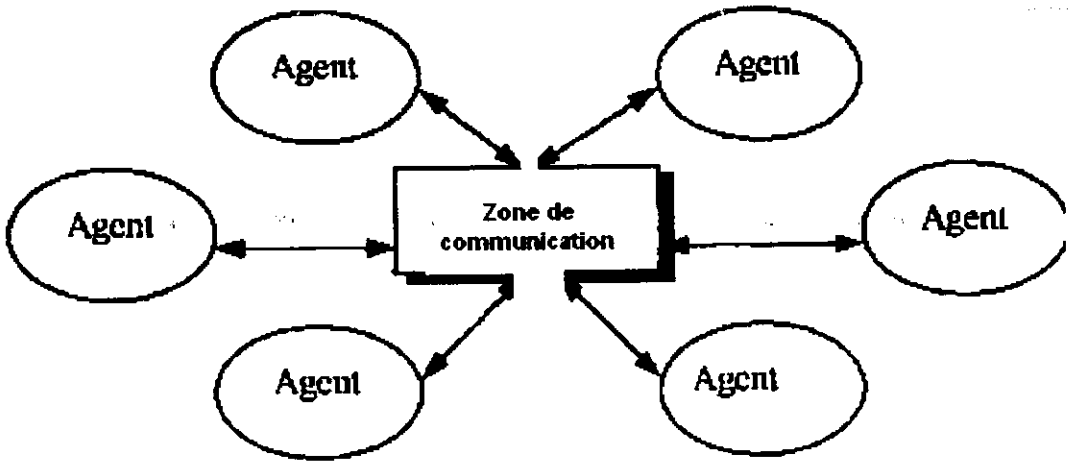


Figure II.04 : Communication par partage d'informations

III.5.2.2. La communication par envoi de messages

Les échanges entre les agents se font par envoi de messages, suivant un protocole bien défini. Deux modes de transmission sont possibles : la transmission directe (point à point), et la transmission par diffusion (broadcast) **[Fer, 95]**. Chaque agent a sa propre base de faits locale, dans la quelle il représente une partie de la solution courante du problème global (Figure II.05). Les messages peuvent comporter des demandes de renseignement ou des informations servant à un échange de résultats partiels. Ce type de communication est bien adapté aux architectures distribuées, et particulièrement aux systèmes d'acteurs **[Fer, 95]**.

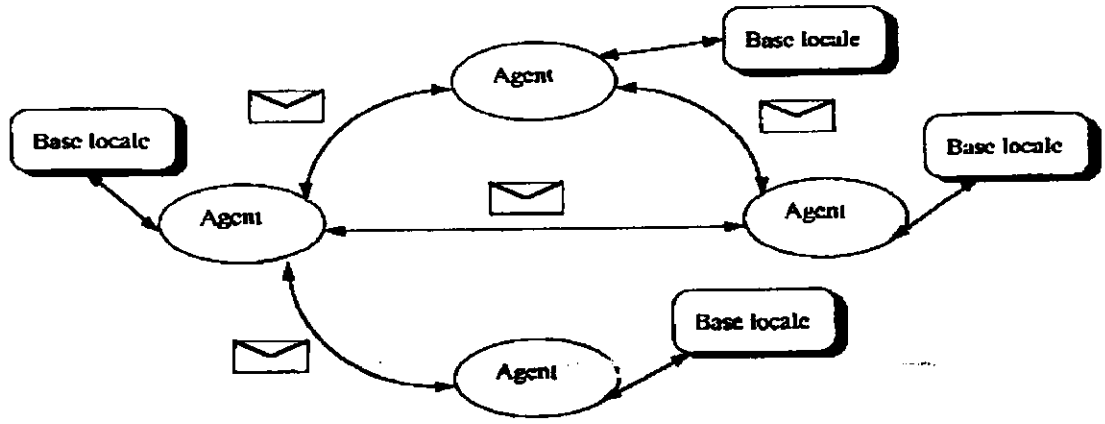


Figure II.05 : Communication par envoi de messages

III.5.2.3. Communication par actes de langages

Ce mode de communication est utilisé pour les agents cognitifs, il utilise une logique de représentation.

Dans les systèmes multi-agents, l'acte de communication n'est pas réduit à un simple envoi de messages entre les agents. L'acte de communication est un acte intentionnel caractérisé par un type et plusieurs composantes.

Il se traduit par une modification des croyances des autres, c'est à dire qu'un agent n'envoie un message à un autre que s'il le juge nécessaire et qu'il lui permet d'apporter une information pour lui-même (cas d'une interrogation) ou pour l'accointance (cas d'une assertion) **[Fer, 95]** .

Un Type peut être :

- ✓ assertif: affirmer quelque chose sur le monde,
- ✓ directif: donner des directives au destinataire,
- ✓ interrogatifs: poser une question,
- ✓ exercitif: demander d'accomplir une action,
- ✓ promissif: s'engager à accomplir certains actes dans l'avenir,
- ✓ expressif: donner des indications concernant son propre état mental.

Une Composantes peut être :

- ✓ composante locutoire (l'énoncé),
- ✓ illocutoire (force associée au message par l'émetteur),
- ✓ perlocutoire (effet sur le destinataire eg persuader).

Il existe plusieurs Langages de modélisation des connaissances d'un agent dont **[Bri, 01]:**

- ❖ FIPA ACL (Foundation for Intelligent Physical Agents, Agent Communication Language).
- ❖ KQML (Knowledge Query and Manipulation Language).
- ❖ KIF (Knowledge Interexchange Format).

III.5.3. L'organisation

L'Organisation de base consiste à prendre en compte des réseaux d'agents (leurs accointances) pour la réalisation d'une tâche commune :

- délégation (Relations hiérarchiques),
- complémentarité (des compétences, des biens ou des connaissances),
- simultanéité (plusieurs agents réalisent la même tâche pour être plus efficaces),
- partage de ressources (producteur/consommateur).

Les organisations permettent de maîtriser et structurer, à la conception :

- ✓ les coordinations entre agents du système.
- ✓ les interactions entre les agents.

IV.CONCLUSION

Nous venons de voir que les systèmes multi agent sont applicables dans les domaines les plus divers. Le chapitre suivant va nous permettre de décrire la structure de SMA exploité dans le processus de segmentation coopératif.

CHAPITRE 3

La Démarche à suivre

I. INTRODUCTION

Pour le bute d'élaboré notre système nous allons décrire les différentes composantes de sa structure et les prés requis nécessaires.

II. PRÉ REQUIS

Le système à développer doit exploiter des informations issues de la segmentation d'une image. Il s'agit ici de réaliser la segmentation selon les deux approches:

Une segmentation par extraction de régions: afin de déterminer le nombre de classes. Les informations seront considérées comme des entrées dans le SMA; elles permettent d'initialiser le nombre d'agents région et leurs emplacements.

Une segmentation pas extraction de contour: afin de produire la carte des contours; ce pré permet de définir les agents contour et leurs emplacements.

L'image est divisée en plusieurs partitions. A l'intérieur de chaque partition sont mis en oeuvre des agents d'interprétation locaux, coordonnés et s'échangeant des informations afin de segmenter leur zone du travail.

III. LES AGENTS DE SMA

Nous distinguons trois catégories d'agents :

- des agents nommés agents de gestion du système, contiennent les informations nécessaires au fonctionnement du système multi agent,
- des agents nommés agents de segmentation, chargé de l'amélioration du pré segmentation initiale et dont le comportement est défini par un automate,
- un agent interface.

Les caractéristiques de ces trois catégories d'agents sont :

- situé,
- autonome,
- réactif,
- social.

III.1. Les agents de gestion du système

Dans cette catégorie d'agents, nous avons les agents suivants :

III.1.1. L'agent moniteur

C'est le centre de contrôle de notre système, il est responsable de l'ordonnancement et du lancement des tâches des autres agents. C'est l'agent moniteur qui crée l'agent interface et les agents de contrôle locaux du système et qui les initialise en utilisant les informations obtenues lors de l'étape de pré traitement. C'est lui qui s'occupe de la suppression des agents inactifs.

III.1.2. Les agents de contrôle locaux

Leurs rôles est de créer et d'organiser les agents de segmentation dans une seule partition de l'image (imagette), et d'informer l'agent moniteur de l'accomplissement de ça tache à la fin de segmentation de l'imagette.

III.2. L'agent interface

Sa tâche principale est de gérer l'affichage de l'état de la segmentation d'image.

III.3. Les agents de segmentation

Nous détaillons maintenant le fonctionnement des agents régions et contours.

III.3.1. Les agents régions

Un agent région représente une primitive région, cette primitive correspond à une classe dans l'image initiale. Chaque agent de ce type marque de son identifiant la zone de l'environnement correspondant à sa primitive. Ces informations locales contiennent les attributs photométriques et géométriques caractérisant la région représentée par l'agent.

L'agent R stocke ses attributs dans une structure de donnée contenant :

- IdR: le numéro d'identification de l'agent;
- RVBR : la valeur d'intensité de chaque pixel dans la région.
- SurfR : la surface de la région (nombre de pixels);
- MoyR: la moyenne des niveaux de gris de la région.

III.3.2. Les agents contours

Chaque agent contour représente une carte de contour d'une partition de l'image, il marque avec son identifiant tout les pixels appartenait à cette carte.

IV. PROTOCOLES DE COOPERATION

Nous souhaitons étudier divers types de coopération (Figure III.01):

- Coopération région-région pour la fusion de régions: il doit exister un désir mutuel de fusion entre les deux régions pour que la fusion soit autorisée.
- Coopération contour-région pour la fusion de régions : il ne doit pas exister de contour entre les deux régions qui souhaitent fusionner. L'agent contour dispose ainsi d'un droit de veto : il peut s'opposer à une fusion.

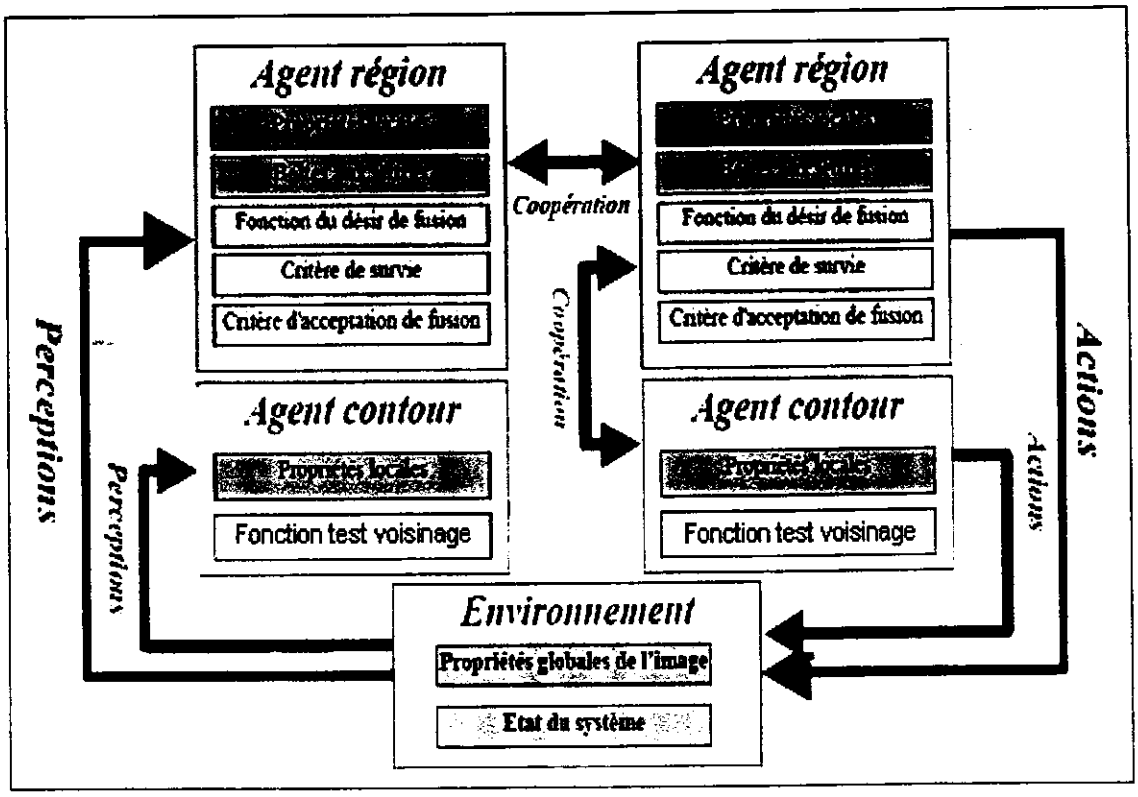


Figure III.01 : Différents types de coopération proposés entre agents régions et agents contours.

V. TABLEAU NOIR

Le tableau noir (blackboard) est une zone partagée qui détient les différentes images et leurs caractéristiques, qui servent de base au travail des agents. Il représente aussi un moyen de communication pour ces agents car toutes les informations qu'il détient sont accessibles à tous les agents de segmentation.

Les informations présentes dans le tableau noir sont :

- image source (c'est l'image d'entrée),
- liste des agents voisins (c'est une liste qui contient tout les voisins de chaque agent de segmentation),
- carte de contours (permet de définir le nombre d'agents contour et leurs emplacements),
- carte de régions (permet de définir le nombre d'agents région et leurs emplacements),
- informations statistiques sur les images et les images segmentées associées (nombre de région, la surface et moyenne d'intensité pour chaque région),
- les boites aux lettres des agents (Pour permettre la communication entre agents régions, l'agent possède une boite aux lettres des messages qui contient les propositions de fusion émis par les agents régions à leurs voisins).

VI. Comportement des agents de segmentation

Pour la gestion de la coopération et de la négociation, nous avons choisi les propositions comme catégorie de message. Une proposition peut être une demande de fusion d'une région avec ses voisines; elle va permettre d'engager un processus de négociation.

Toutes les mises à jour nécessaires doivent être faites après la fusion de deux régions. Pour simplifier et optimiser les traitements, il est nécessaire d'associer des priorités aux messages.

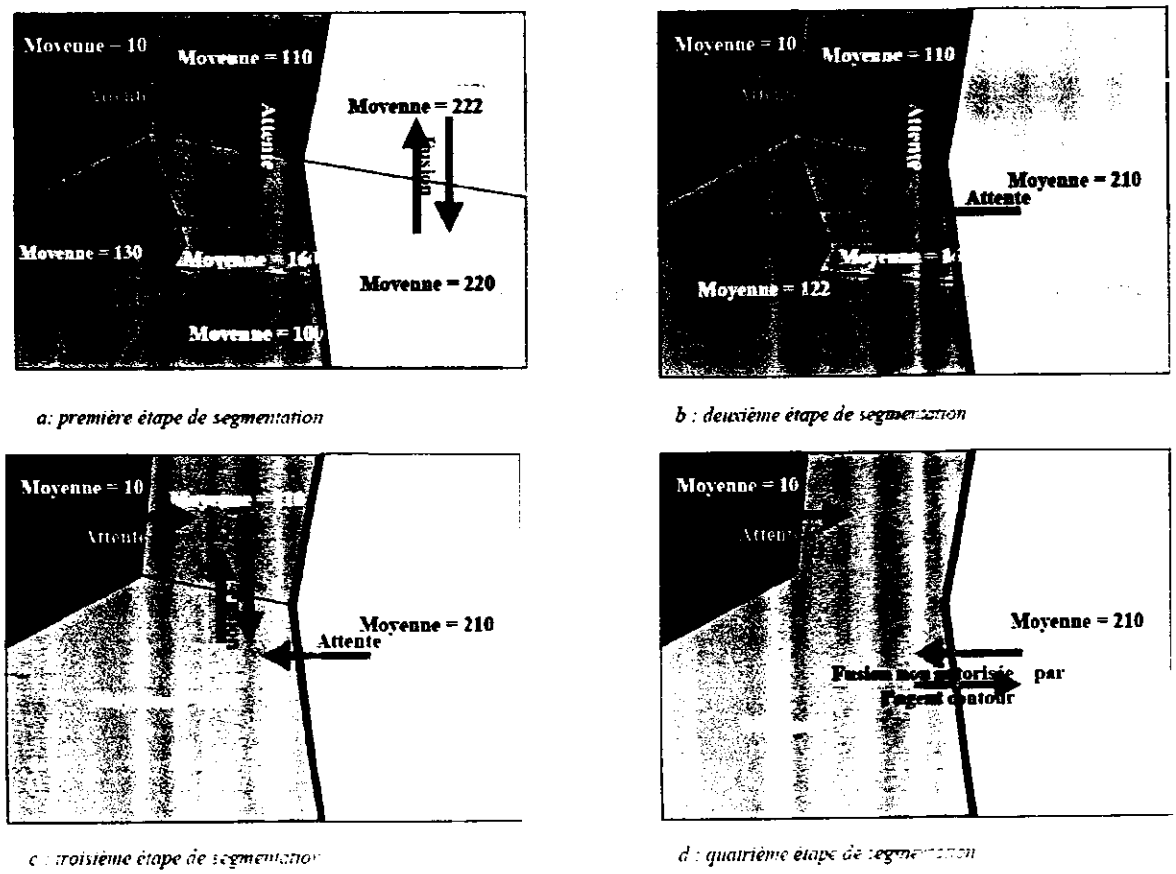
Le comportement d'un agent est défini sous forme d'un automate, ce qui présente un double intérêt : d'une part, l'automate permet d'avoir une vue d'ensemble du comportement d'un agent; d'autre part, les états et les transitions sont clairement définis et peuvent plus facilement être modifiés. Par exemple, le critère de fusion entre deux régions est défini comme une transition de l'automate; ainsi, on peut aisément changer de critère; il en est de même pour le critère de survie

(qui consiste pour un agent région à décider lequel parmi les deux agents régions à fusionner va survivre).

Pour mettre en œuvre la fusion de deux régions, chaque agent région connaît son « voisinage ». Pour qu'une fusion ait lieu, le désir de fusion doit être réciproque : si R1 et R2 désignent le couple de régions candidate à la fusion, le meilleur voisin de R1 doit être R2 et le meilleur voisin de R2 doit être R1.

La (Figure III.02) illustre la coopération région-région et région contour pour la fusion de régions. Au départ toutes les régions calculent leur désir de fusion avec chacun de leur voisins et ces désirs sont communiqués au meilleur voisin par un message (par exemple une flèche sur la (Figure III.02 a) entre R10 et R110). Une opération de fusion est exécutée quand le désir de fusion est mutuel (Figure III.02 b) entre R160 et R122) et la région se met alors dans un état d'attente jusqu'à exécution de la fusion de son meilleur voisin (par exemple la région R110 dans la (Figure III.02 b)).

Comme on peut voir sur la (Figure III.02 c) la fusion des deux régions R210-R130 est stoppée par l'agent contour qui se situe aux frontières de ces deux régions.



a: première étape de segmentation

b : deuxième étape de segmentation

c : troisième étape de segmentation

d : quatrième étape de segmentation

Figure III.02: Exemple de coopération région-contour et région-région pour la fusion des régions

Les sept comportements d'un agent sont:

1. **marquage de territoire :** il consiste à ancrer l'agent dans l'image et à définir la partie (primitive contour ou région) de celle-ci dont il est le représentant.
2. **exploration :** elle permet aux agents de découvrir leur voisin 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. **planification de la fusion :** l'objectif d'un agent est d'interagir avec ses voisins d'adjacences afin de déterminer avec lesquels il souhaite fusionner. Il va donc construire un plan de fusion qui est équivalant à un graphe de similarité des pyramides irrégulières.
4. **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 un comportement région/région et région/contour. Les modifications entraînées sur le plan de fusion sont équivalentes à un traitement qui vise l'amélioration du graphe de similarité.
5. **décimation :** elle vise à sélectionner des survivants parmi les agents de niveau courant de la pyramide. Cette sélection est basée sur deux aspects le premier est de satisfaire un critère de survie. L'agent qui dispose de la plus grande surface c'est celui qui va être le survivant. Le deuxième aspect est l'utilité d'associer le plan de fusion précédemment calculé. Ceci revient dans le cadre des pyramides irrégulières à favoriser la survivance des sommets ayant des nombreux arcs sortants dans le graphe de similarité.
6. **rattachement :** il permet aux agents non survivants de ce rattacher à un survivant afin d'être représenter dans le niveau suivant de la pyramide.
7. **reproduction :** elle permet aux agents survivants de crées un nouvel agent dans le niveau suivant de la pyramide. Ce nouvel agent représente tous les agents rattaches au survivant.

La (Figure III.03) illustre les sept comportements d'un agent :

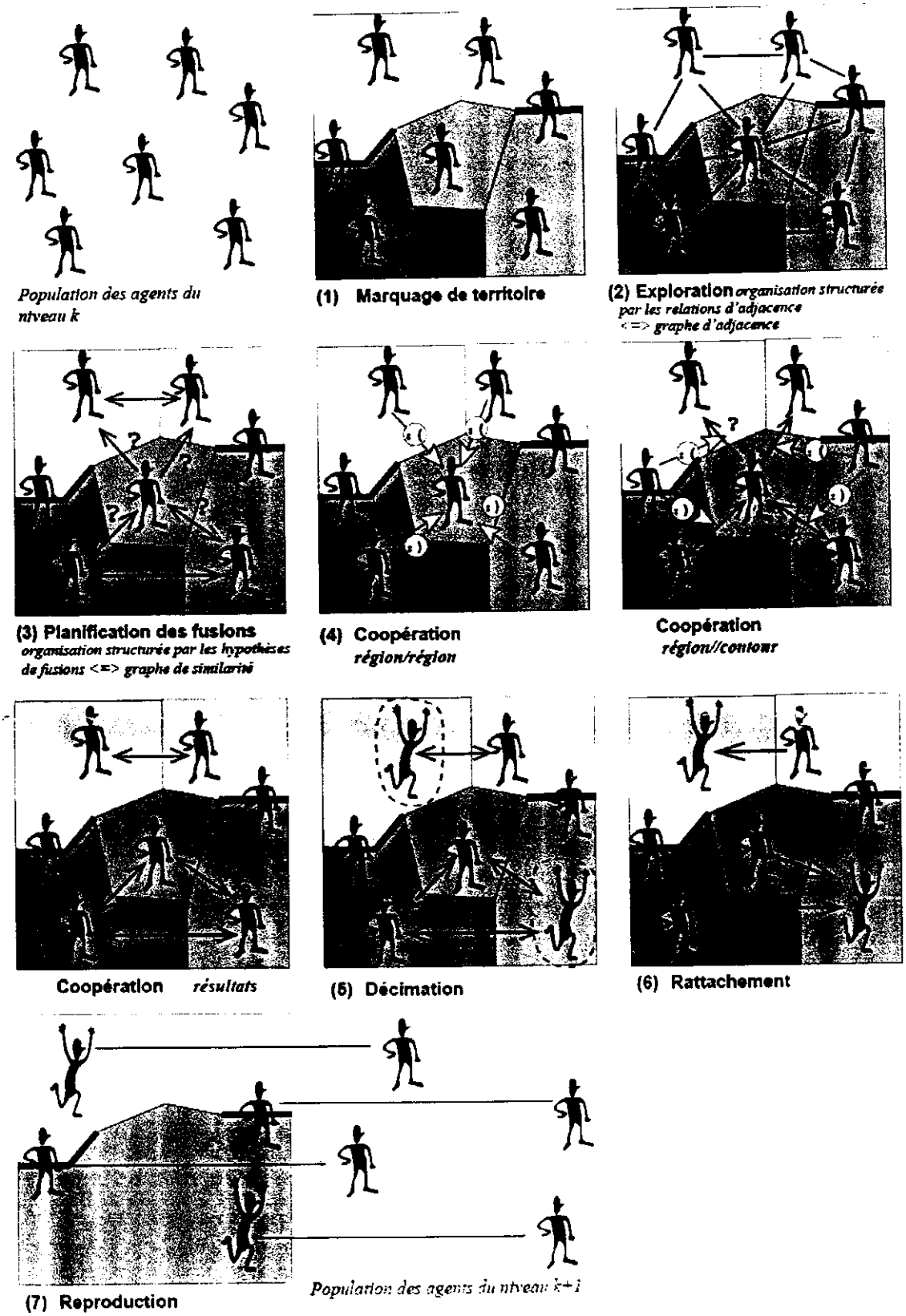


Figure III.03 : Les sept comportements des agents de segmentation

VII. CONCLUSION

Dans ce chapitre nous avons présenté d'une manière générale la démarche à suivre pour réaliser notre système. Une description formelle plus détaillée sera étalée dans le chapitre suivant.

CHAPITRE 4

Conception Du Système

I. Langage de Description UML Et Le processus unifié UP

I. INTRODUCTION

Pour tout projet informatique d'envergure il est indispensable de parler d'un langage commun qui permet de formaliser les besoins d'une part et les exigences de l'utilisateur d'autre part. Il s'agit aussi de trouver des solutions techniques envisager par le développeur. Le secteur informatique à adopter UML comme langage pivot, il est très complet, basé sur le concept orienté objet et qui est indispensable de tous les langages de programmation.

Nous avons utilisé UML2 comme langage de modélisation en se basant sur le processus de développement UP. Ainsi nous présentons un rappel sur ces deux concepts.

II. DEFINITION UML

UML (Unified Modeling Language) est né de la consolidation de trois méthodes objet : *OMT* (Rumbaugh), *Booch*, *OOSE* (Jacobson). Cette consolidation a été marquée par trois étapes :

- le regroupement des trois équipes au sein de la société Rationnel,
- le recentrage du projet de standardisation sur le langage de modélisation, les aspects purement méthodologiques étant laissés de coté,
- la décision de l'Object management group en 1997.

Le succès a été immédiat et UML est aujourd'hui universellement accepté et supporté par l'ensemble des outils de développement [Ref, 01]. L'architecture d'UML est Basée sur la notion des cas d'utilisation. UML est élaboré en modèles, il est constitué d'éléments de modélisation et d'un ensemble de diagrammes.

III. ELEMENTS DE MODELISATION

Les éléments de modélisation représentent les abstractions du système en cours de modélisation. Ils représentent toutes les propriétés du langage. Parmi les éléments de modélisation, il y a un groupe de concepts permettant l'extension d'UML pour l'adapter à des utilisations et des domaines particuliers. Ce groupe est constitué des concepts de stéréotypes.

Les éléments de modélisation sont regroupés en paquetages. Le paquetage permet d'organiser des éléments de modélisation en groupes. Il peut contenir des classes, des cas d'utilisations, des interfaces,...etc **[Ger, 06]**.

Chaque paquetage est représenté graphiquement par un dossier (Figure IV.01).

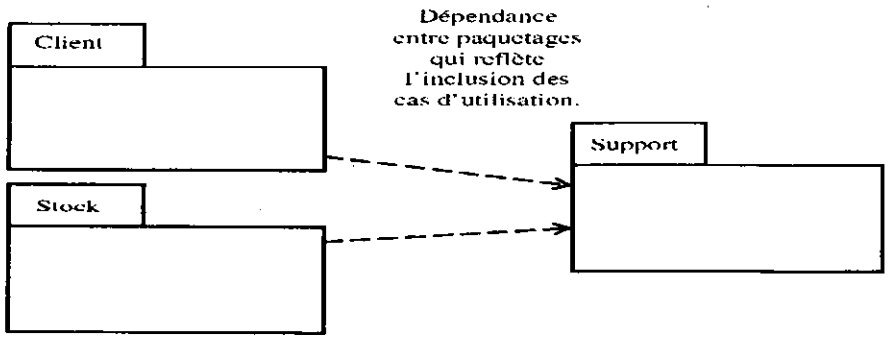


Figure IV.01 : Regroupement de cas en paquetages

IV. DIAGRAMMES

UML permet de construire plusieurs modèles d'un système: certains montrent le système du point de vue des utilisateurs, d'autre montrent sa structure interne, d'autre encore en donnent une vision globale ou détaillée **[Ref, 01]**. Les modèles se complètent et peuvent être assemblés. Ils sont élaborés tout au long du cycle de vie du développement d'un système (depuis le recueil des besoins jusqu'à la phase de conception). UML définit neuf diagrammes répartis on trois catégories : structurel, comportemental et d'interaction.

- Diagrammes Structurels ou Diagrammes statiques (Deployment diagram)
  - Diagramme de classe
  - Diagramme de composant
  - Diagramme de déploiement
  - Diagramme d'objet
- Diagrammes Comportementaux (Behavior Diagram)
  - Diagramme de cas d'utilisation
  - Diagramme d'état transition
  - Diagramme d'activité

- Diagramme d'interactions ou Diagrammes dynamiques (Interaction Diagram)

  Diagramme de séquence

  Diagramme de communication **[Ref, 01]**.

Nous allons commencer par un des modèles en l'occurrence le premier à construire: le diagramme de cas d'utilisation qui permet de recueillir, d'analyser et d'organiser les besoins. Ce modèle déclenche le processus d'analyse d'un système **[Cha, 05]**.

IV.1. Diagramme de cas d'utilisation

Un cas d'utilisation est une manière spécifique d'utiliser un système. Les acteurs sont à l'extérieur du système; ils modélisent tout ce que interagit avec lui. Un cas d'utilisation réalise un service de bout en bout, avec un déclanchement, un déroulement et une fin, pour l'acteur qui l'initie **[Cha, 05]**.

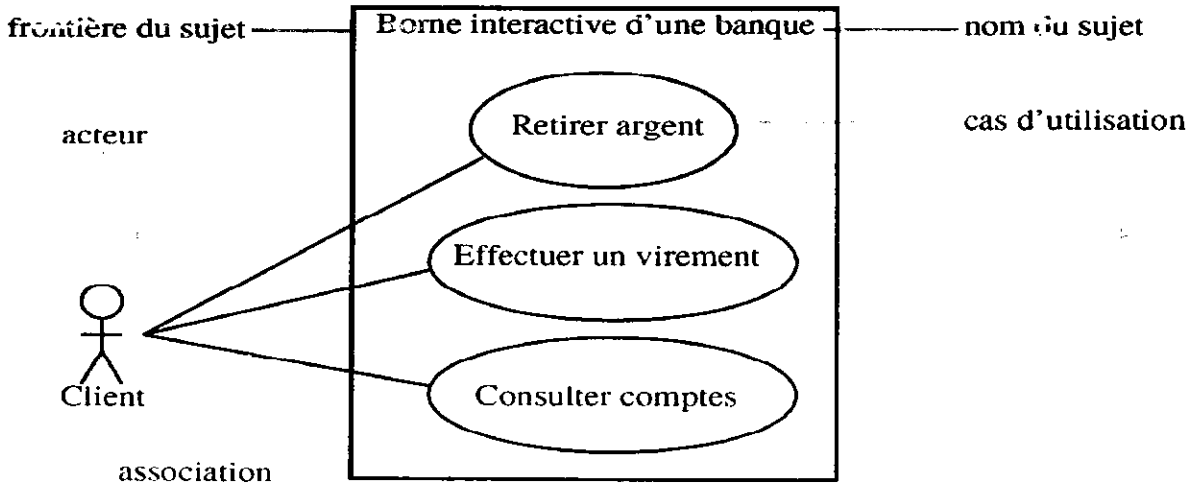


Figure IV.02: Diagramme de cas d'utilisation modélisant une borne d'accès à une banque.

Les relations qui peuvent exister dans un diagramme de cas d'utilisation sont :

- **Utilisation :** elle signifie qu'une instance du cas source comprend également le comportement décrit par le cas d'utilisateur destination.

- **Inclusion :** un cas A inclut un cas B si le comportement décrit par le cas A inclut le comportement du cas B. Lorsque A est sollicité, B l'est obligatoirement, comme une partie de A.

Exemple : la connexion à un serveur inclut une identification. Il n'y a jamais de connexion sans identification.

- **Extension :** si le comportement de A peut être étendu par le comportement de B, on dit alors que B étend A. Exécuter A peut éventuellement entraîner l'exécution de B. Dans le déroulement de A, le moment où l'on peut avoir besoin d'exécuter A est un point d'extension. Une extension est souvent soumise à condition. Contrairement à l'inclusion, l'extension est optionnelle.

- **Généralisation :** un cas A est une généralisation d'un cas B si A est un cas particulier de B **[Ger, 06]**.

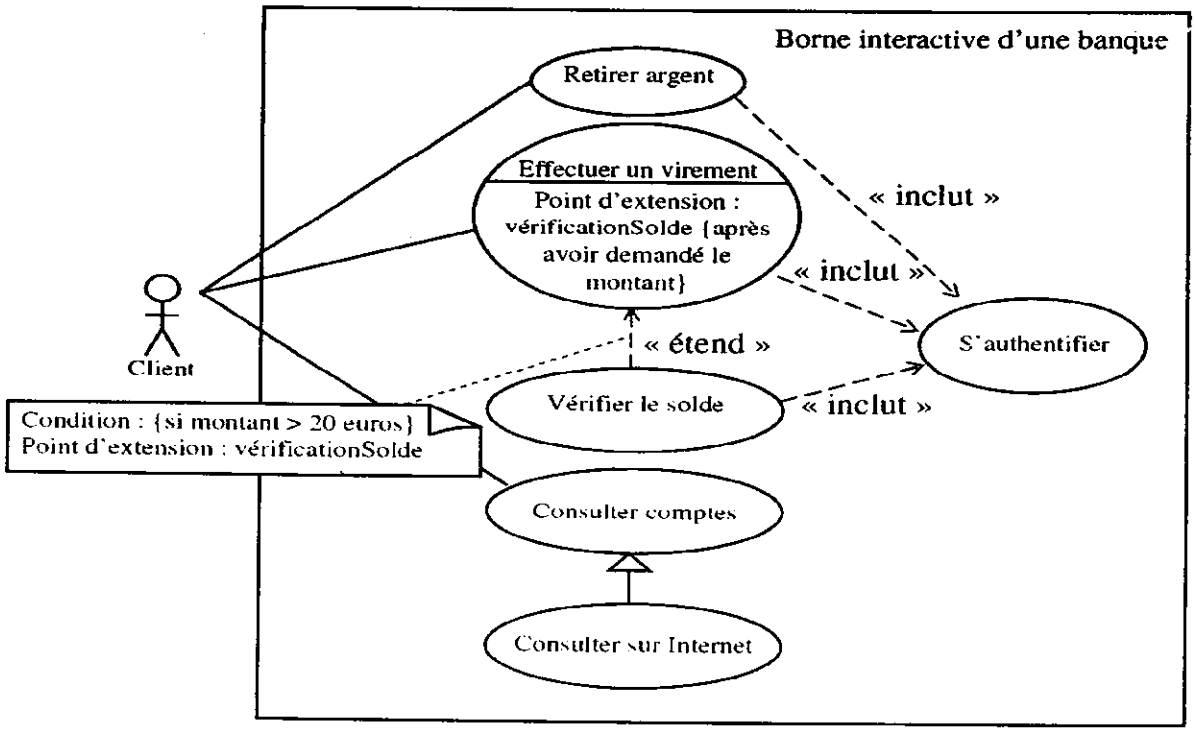


Figure IV.03: Relation entre cas dans un diagramme de cas d'utilisation.

IV.2. Diagramme de classes

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il fournir une représentation abstraite et générale des objets du système, en termes de classes et de relations entre ces classes. Une classe permet de décrire un ensemble d'objets (attributs et comportement), tandis qu'une relation ou association permet de faire apparaître des liens entre ces objets. On peut donc dire **[Ger, 06]**:

- un objet est une instance de classe,
- un lien est une instance de relation.

Une classe est la description d'un ensemble d'objets ayant une sémantique, des attributs, des méthodes et des relations en commun (Figure IV.04).

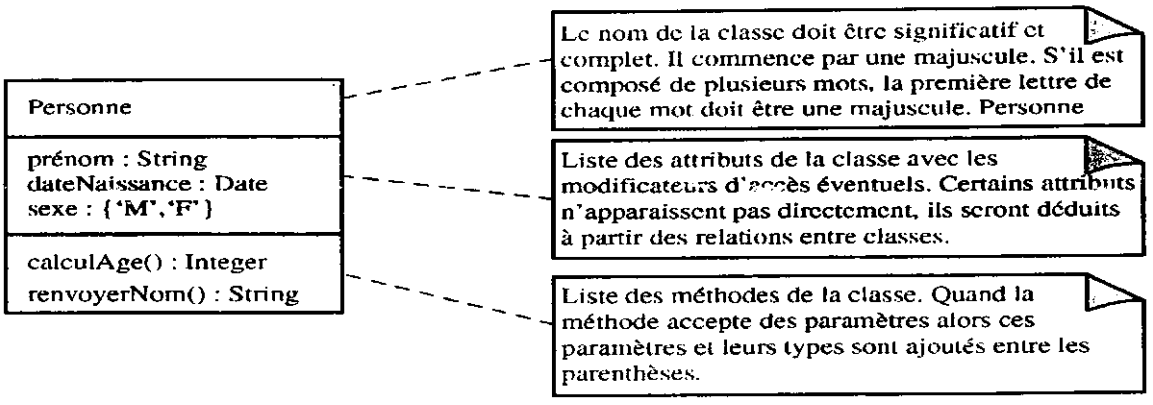


Figure IV.04: Représentation d'une classe

Types de relation entre classes

* Relation binaire c'est la plus utilisée, elle relie deux classeurs (Figure IV.05).

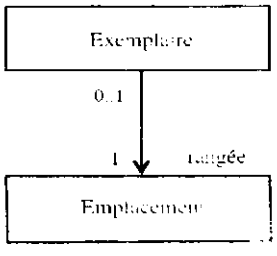


Figure IV.05: Représentation d'une relation binaire

* Relation n-air elle relie plus de deux classes, on la représente par un losange central pouvant éventuellement accueillir une classe-association.

* Relation d'héritage est une relation de spécialisation/généralisation, les éléments spécialisés héritent de la structure et du comportement des éléments plus généraux.

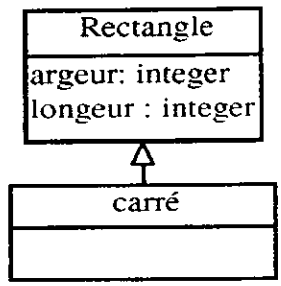


Figure IV.06: Représentation d'une relation d'héritage.

* Relation de composition décrit une contenance structurelle entre instances. On utilise un losange plein.

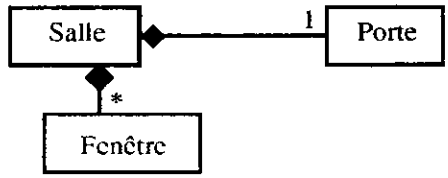


Figure IV.07: Représentation d'une relation de composition

* Relation d'agrégation est une forme particulière d'association. Elle représente la relation d'inclusion structurelle ou comportementale d'un élément dans un ensemble. On représente l'agrégation par l'ajout d'un losange vide du côté de l'agrégat.

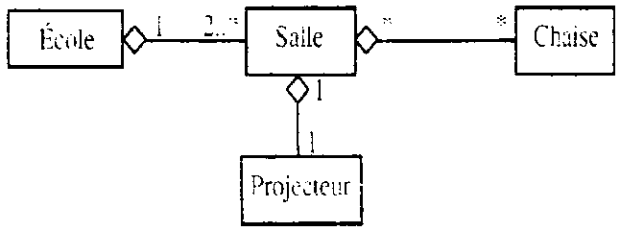


Figure IV.08: Représentation d'une relation d'agrégation

* Relation réflexive, les deux extrémités de l'association pointent vers le même classeur

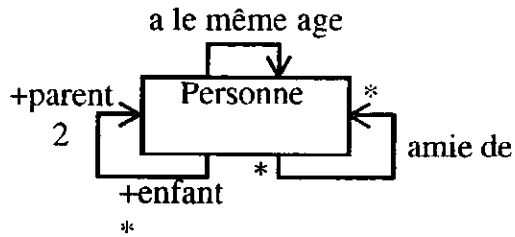


Figure IV.09: Représentation d'une relation réflexive

IV.3. Diagramme d'objet

Le diagramme d'objets représente les objets d'un système à un instant donné. Il permet de :

- **illustrer le modèle de classes (en montrant un exemple qui explique le modèle),**
- préciser certains aspects du système (en mettant en évidence des détails imperceptibles dans le diagramme de classes),
- **exprimer une exception (en modélisant des cas particuliers, des connaissances non généralisables. . .).**

Le diagramme d'objets modélise des faits, il est essentiellement utilisé pour comprendre ou pour illustrer des parties complexes du diagramme de classes **[Ger, 06]**.

Représentation des objets

- on utilise des cadres compartimentés,
- les noms des objets sont soulignés et on peut rajouter son identifiant devant le nom de sa classe,
- les valeurs (a) ou l'état (f) d'un objet peuvent être spécifiées (Figure IV.10),
- les instances peuvent être **anonymes** (a,c,d), **nommées** (b,f), **orphelines** (e), **multiples** (d) ou **stéréotypées** (g) (Figure IV.10),
- un lien est une instance d'une association,
- un lien se représente comme une association mais s'il a un nom, il est souligné,
- naturellement, on ne représente pas les multiplicités.

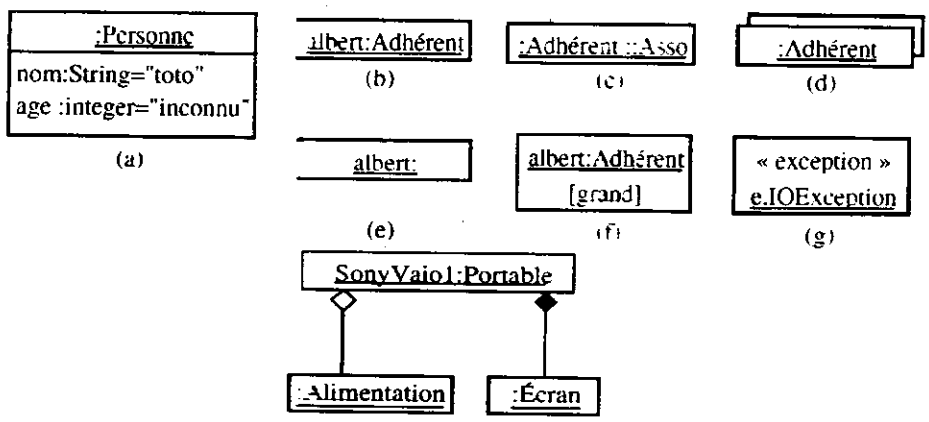


Figure IV.10: Représentation des objets et des liens

IV.4. Diagramme d'interaction

Les diagrammes d'interaction montrent comment des instances ou cœur du système communiquent pour réaliser une certaine fonctionnalité, ils permettent d'établir un lien entre la vision fonctionnelle et externe (diagramme de cas d'utilisation) et la vision statique et structurelle (diagramme de classes) **[Cha, 05].**

Les diagrammes de communication et les diagrammes de séquences sont deux types de diagramme d'interaction.

- Un **diagramme de séquence** montre des interactions sous un angle temporel, en mettant l'emphase sur le séquencement temporel de messages échangés entre des lignes de vie
- Un **diagramme de communication** montre une représentation spatiale des lignes de vie.
- Ils représentent la même chose, mais sous des formes déférentes.

A ces diagrammes, UML 2.0 en ajoute un troisième : le **diagramme de timing,** Son usage est limité à la modélisation des systèmes qui s'exécutent sous de fortes contraintes de temps, comme les systèmes temps réel.

Un diagramme d'interaction se représente par un rectangle contenant, dans le coin supérieur gauche, un pentagone accompagné du mot-clé **sd** lorsqu'il s'agit d'un diagramme de séquence (Figure IV.11), et **com** lorsqu'il s'agit d'un diagramme de communication (Figure IV.12). Le mot clé est suivi du nom de l'interaction. Les messages sont représentés par une flèche horizontale.

Fragment combiné

Un fragment combiné permet de décomposer une interaction complexe en fragments suffisamment simples pour être compris. Il est représenté un rectangle dont le coin supérieur gauche contient un pentagone accompagné d'un de ces mots-clé (alt, loop, break, par, ref, ...ect).

- **alt** Fragment " Alternatif " (IF - THEN - ELSE)
- **loop** Fragment " Loop " utilisé pour décrire un ensemble d'interactions qui s'exécutent en boucle.
- **break** Fragment " break " utilisé pour représenter des scénarios d'exceptions.
- **par** Fragment " Parallel " utilisé pour représenter des interactions en parallèle.
- **ref** Références à un pointeur ou un raccourci vers un autre diagramme de séquences existant.

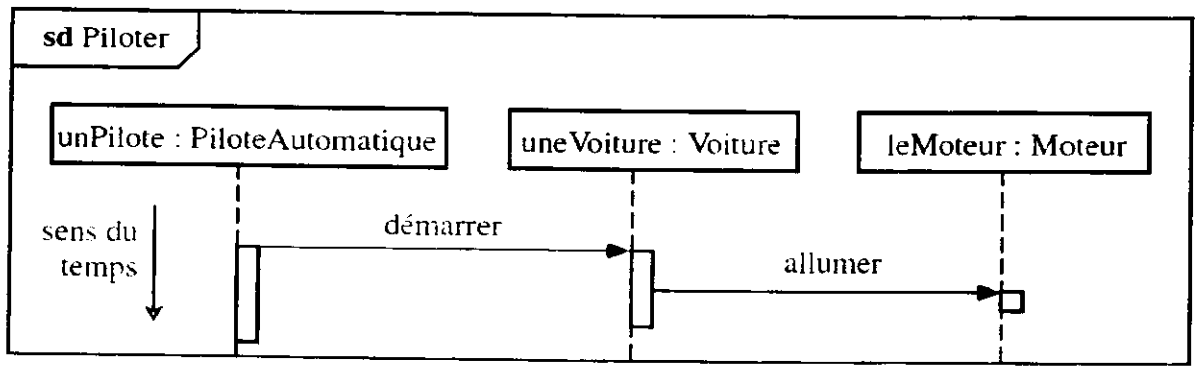
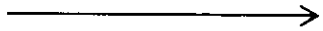


Figure IV.11: Représentation de diagramme de séquence

Sur les diagrammes de séquence, on distingue plusieurs types de messages :

- **Un message synchrone** bloque l'expéditeur jusqu'au retour du destinataire. Le flot de contrôle passe de l'émetteur au récepteur.

- **Un message asynchrone** n'interrompt pas l'exécution de l'expéditeur. Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré.



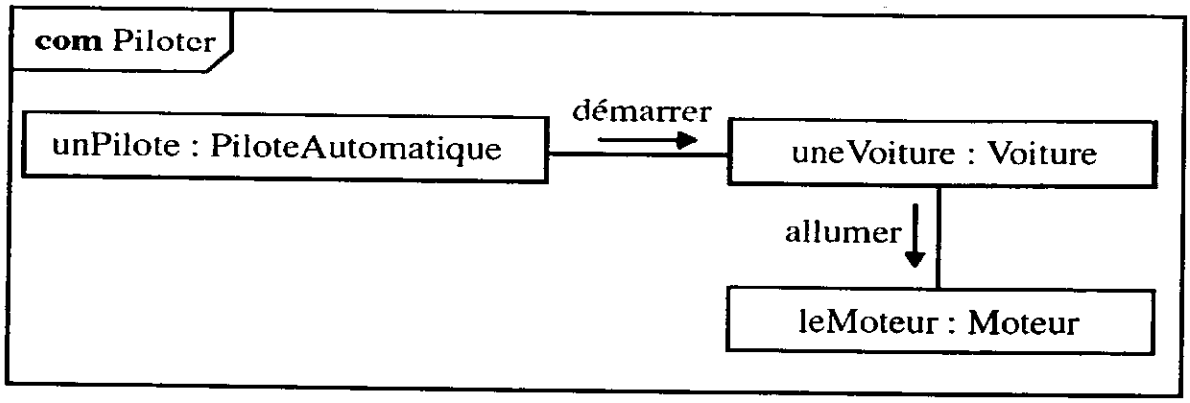


Figure IV.12: Représentation de diagramme de communication

IV.5. Diagramme de collaboration

Une collaboration montre des instances qui collaborent dans un contexte donné pour mettre en œuvre une fonctionnalité d'un système.

Une collaboration est représentée par une ellipse en traits pointillés comprenant deux compartiments. Le compartiment supérieur contient le nom de la collaboration, et le compartiment inférieur montre les participants à la collaboration **[Ger, 06]**.

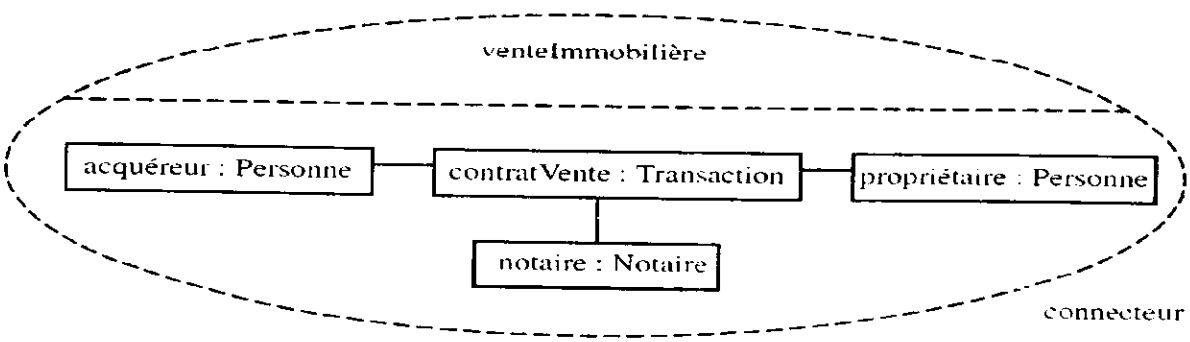


Figure IV.13: Première représentation d'un diagramme de collaboration

Une collaboration peut aussi se représenter par une ellipse sans compartiment, portant le nom de la collaboration en son sein. Les instances sont reliées à l'ellipse par des lignes qui portent le nom du rôle de chaque instance.
On peut ainsi former des diagrammes de collaborations.

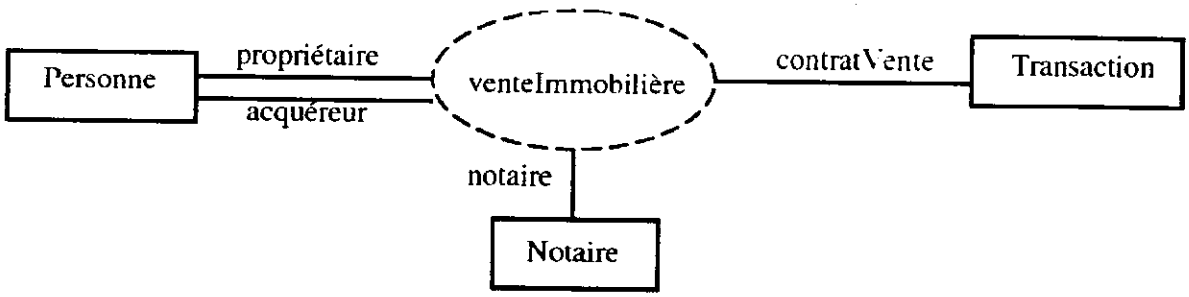


Figure IV.14: Seconde représentation d'un diagramme de collaboration

Les collaborations donnent lieu à des interactions

Les interactions documentent les collaborations

Les collaborations organisent les interactions.

Les interactions se représentent indifféremment par des diagrammes de **communication** ou de **séquence** (Figure IV.15).

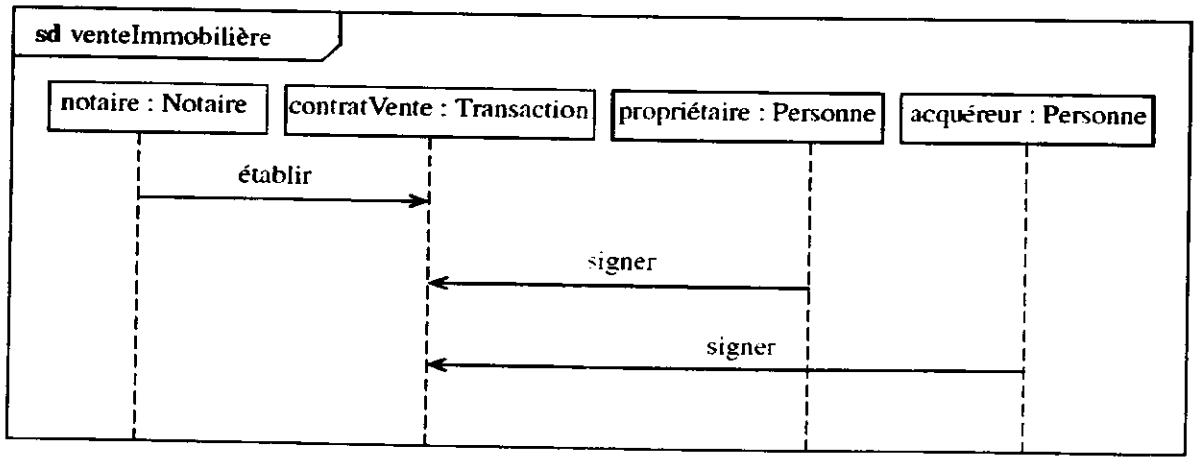


Figure IV.15: Un diagramme de séquence pour illustrer une collaboration.

IV.6. Diagramme d'état-transition

Un diagramme d'état transition est un graphe qui représente le comportement interne d'un objet à l'aide d'un automate à état finis, c'est-à-dire une machine dont le comportement de sortie ne dépend pas seulement de l'état de ses entrées, mais aussi d'un historique des sollicitations passées:cet historique est caractérisé par un état.

Ainsi, l'effet d'une action sur un objet dépend de son état interne, qui peut varies au cours du temps. Par exemple, considérées une lampe munie de deux bouton poussoirs: une pression sur ON allume la lampe et une pression sur OFF l'éteint. Une pression sur ON ne produit pas l'effet si la lampe est déjà allumée; la réaction d'une instance de lampe à cet événement dépend de son état interne **[Cha, 05].**

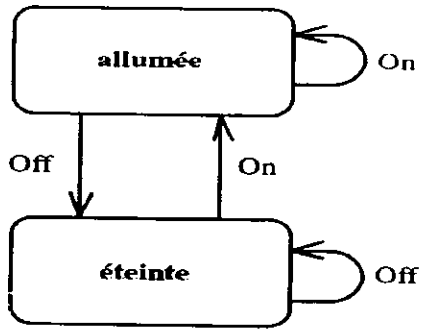


Figure IV.16: Représentation d'un diagramme d'état transition simple.

On représente les états par des rectangles aux coins arrondis, et les transitions par des arcs orientés liant les états entre eux. On trouve Certains états, dits composites, qui peuvent contenir des sous-diagrammes **[Ger, 06].**

IV.7. Diagramme d'activités

Les diagrammes d'activités permettent de spécifier des traitements à priori séquentiels. Ils offrent un pouvoir d'expression très proche des langage de programmation objet : spécification des actions de base (déclaration des variables, affectation..), structure de contrôles (conditionnelles, boucles..), ainsi que les instruction particulières à la programmation orientée objet (appels d'opération, exception..). Ils sont bien donc adaptés à la spécification détaillée des traitements en phase de réalisation.

On peut aussi les utiliser de façon plus informelle pour décrire des enchaînements d'action de haut niveau, en particulier pour la description détaillée des cas d'utilisation.
Ces diagrammes sont assez semblables aux états-transitions mais avec une interprétation différente **[Cha, 05]**.

IV.8. Diagramme de composants et de déploiement

- Diagramme de composants

Les diagrammes de composants permettent de décrire l'architecture physique et statique d'une application en terme de modules : fichiers sources, librairies, exécutables,...etc. Ils montrent la mise en oeuvre physique des modèles de la vue logique avec l'environnement de développement.

Les dépendances entre composants permettent notamment d'identifier les contraintes de compilation et de mettre en évidence la réutilisation de composants. **[Ref, 04]**

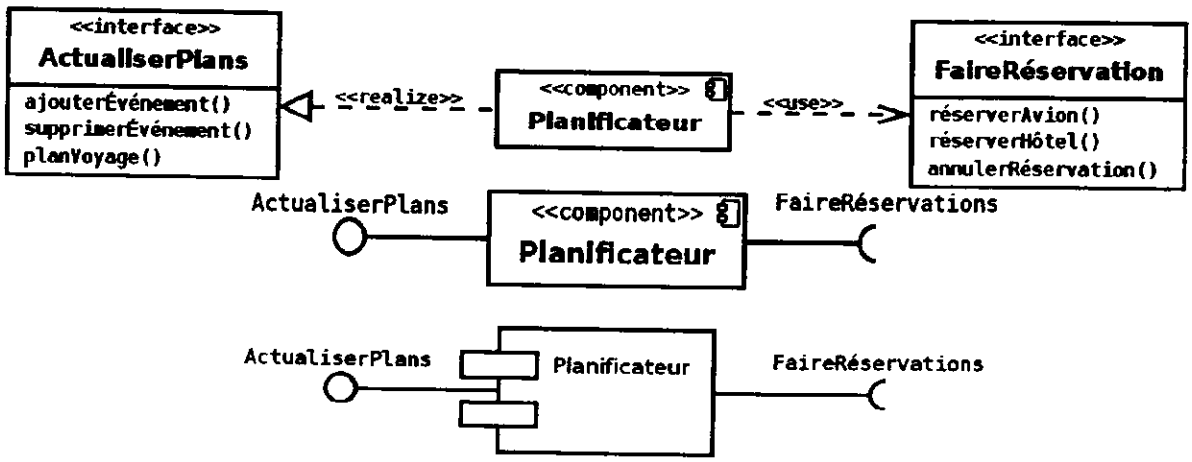


Figure IV.17: Différents représentation d'un composant et interfaces

- Architecture matérielle [Ger, 06].

En dernier lieu, un système doit s'exécuter sur des ressources matérielles dans un environnement matériel particulier.
UML permet de représenter un environnement d'exécution ainsi que des ressources physiques (avec les parties du système qui s'y exécutent) grâce aux diagrammes de déploiement.

* L'environnement d'exécution ou les ressources matérielles sont appelés « nœuds ».
* Les parties d'un système qui s'exécutent sur un noeud sont appelées « artefacts ».

V. Le processus de développement UP

V.1. Définition

Processus unifié (PU ou UP en anglais pour Unified Process) est une méthode de prise en charge du cycle de vie d'un logiciel et donc du développement, pour les logiciels orientés objets. C'est une méthode générique, itérative et incrémentale, contrairement à la méthode séquentielle Merise (ou SADT) **[Ref, 02]**.

Ce processus n'est pas applicable directement, il définit des principes et une architecture, mais doit être adapté à l'organisation et au projet visés.

V.2. intérêt d'UP

Le processus unifié est un processus de développement logiciel : il regroupe les activités à mener pour transformer les besoins d'un utilisateur en système logiciel. UP est une démarche de développement qui est souvent utilisé conjointement au langage UML, elle vient compléter la systémique des modèles UML **[Ger, 06].**

V.3. Les caractéristiques d'UP

Caractéristiques essentielles du processus unifié **[Gal, 01] :**

- **Le processus unifié est à base de composants,**
- **Le processus unifié utilise le langage UML** (ensemble d'outils et de diagramme),
- **Le processus unifié est piloté par les cas d'utilisation,**

La principale qualité d'un logiciel est son utilité; c'est à dire l'adéquation du service rendu par le logiciel avec les besoins des utilisateurs. Le processus de développement sera donc centré sur l'utilisateur.

Les cas d'utilisation font apparaître les besoins fonctionnels et leur ensemble constitue le modèle des cas d'utilisation qui décrit les fonctionnalités complètes du système **[Ger, 06]**.

- Centré sur l'architecture,

UP utilise le model des 4+1 vues pour Modélisation de différentes perspectives indépendantes et complémentaires, ce modèle comprend quatre vues principales et une vue coordinatrice :

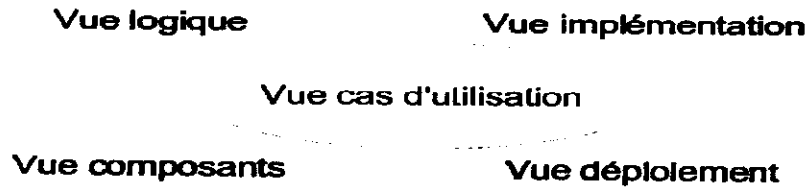


Figure IV.18: Architecture en couches et vues de Krutchen (vues 4+1)

* **Vue cas d'utilisation:** Description du système comme un ensemble de transactions du point de vue de l'utilisateur
* **Vue logique:** Créée lors de la phase d'élaboration et raffinée lors de la phase de construction utilisation de diagrammes de classes, de séquences...
* **Vue composants:** Description de l'architecture logicielle
* **Vue déploiement:** Description de l'architecture matérielle du système
* **Vue implémentation:** Description des algorithmes, code source **[Ger, 06].**

- Itératif et incrémental.

Le développement d'un produit logiciel destiné à la commercialisation donc on doit favoriser une diminution progressive des risques des le début du projet, c'est-à-dire on ne va pas tout développer d'un coup. On peut découper le travail en plusieurs parties qui sont autant de mini projets. Chacun d'entre eux représentant une itération qui donne lieu à un incrément. Une **itération** désigne la succession des étapes de l'enchaînement d'activités, tandis qu'un **incrément** correspond à une avancée dans les différents stades de développement.

VI. Conclusion

Dans cette section nous avons présenté les différents diagrammes de langage de modélisation UML et le processus de développement UP. Ces derniers vont être utilisés dans la conception de notre systéme présenté dans le chapitre suivant.

CHAPITRE 4

Conception Du Système

II. Modélisation du Système

I. INTRODUCTION

Apres avoir présenté un rappel sur le langage UML et le processus unifié UP, on passe dans cette partie là à la présentation des différents diagrammes qui modélisent notre système avec la description détaillée de chaque diagramme.

En se basant sur le processus de développement unifié UP, les diagrammes qui modélisent notre système sont :

- diagrammes de cas d'utilisation,
- diagrammes de séquence et scénario,
- diagramme de classes,
- diagramme d'état,
- diagramme d'activité,
- diagramme de composant.

II. FONCTIONNEMENT DU SYSTEME

1- Créer un agent Moniteur qui va diriger la segmentation, cet agent décompose l'image en petites parties (imagette), puis crée un agent local sur chacune de ces parties.

2- Pour chaque (imagette) l'agent local doit créer :

2.1- un agent contour,

2.2- des agents région (le nombre d'agents dépend de nombre des pics de la carte région).

3- L'agent région commence par la construction de son voisinage pour qu'il puisse chercher le meilleur voisin, il coopère avec les agents région de son voisinage et l'agent contour pour fusionner.

4- Dés que la segmentation est terminée (c'est-à-dire après plusieurs fusions successive) le résultat sera affiché sur l'écran par l'agent interface qui a été crée par l'agent moniteur.

III. DIAGRAMMES DES CAS D'UTILISATION ET DE SEQUENCES

III.1. Diagramme de cas d'utilisation Globale du système

Il contient les tâches principales de notre système (Figure IV.19):

- Ouvrir une image,
- Segmenter une image,
- Enregistrer une image,
- Etat de la segmentation.

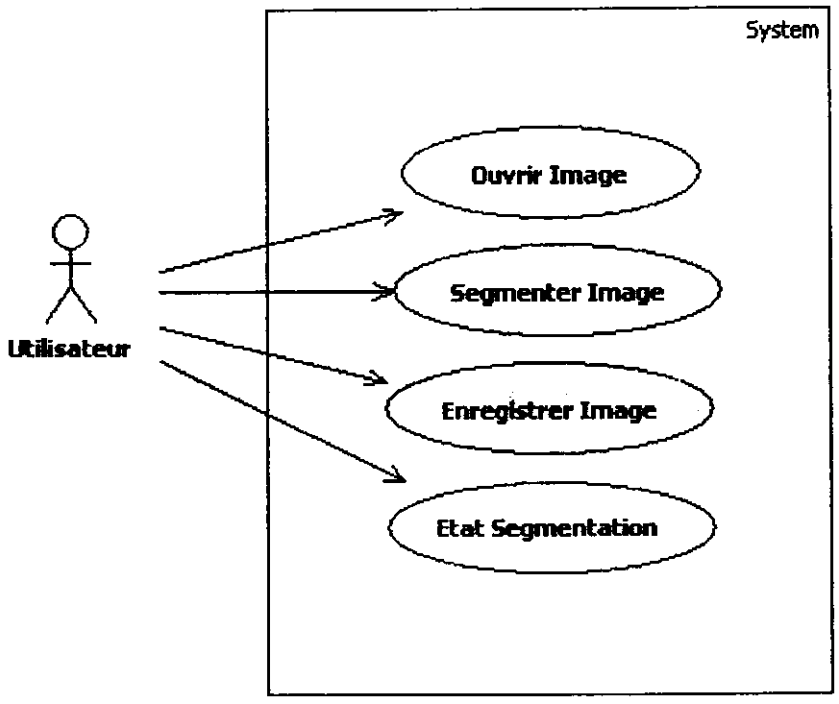


Figure IV.19 : Diagramme de cas d'utilisation globale du système

Chaque tâche de ce diagramme de cas d'utilisation globale sera détaillée dans ce qui suit, sauf les deux tâches élémentaires ouvrir image et Enregistrer image.

III.2. Diagramme de cas d'utilisation pour la segmentation de l'image

Ce diagramme est destiné aux phases de la segmentation de l'image.

Pour segmenter l'image choisie il faut d'abord initialiser l'image puis commencer la segmentation en utilisant les SMA. A la fin, le système procède à la finalisation de segmentation et l'affichage de résultat. La (Figure IV.20) schématise les différentes étapes.

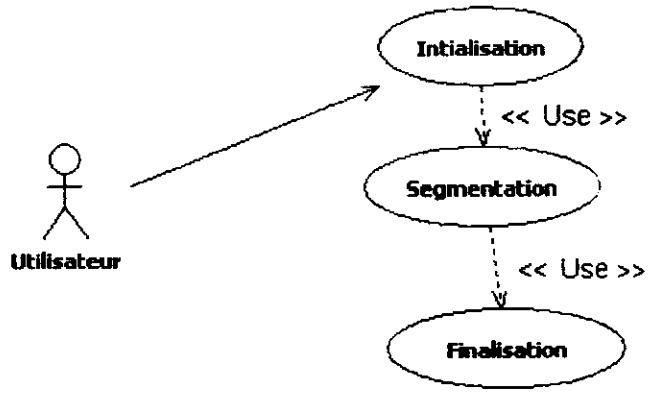


Figure IV.20 : Diagramme de cas d'utilisation pour la segmentation de l'image.

Les Diagramme de séquence

- **Le Scénario de l'initialisation de segmentation de l'image**

  1- L'utilisateur lance la segmentation de l'image choisie.
  2- Le système crée un agent moniteur.
  3- L'agent moniteur prend l'image choisie et calcule le nombre d'imagettes possibles.
  4- L'agent moniteur décompose l'image initiale en petites images selon le nombre qu'il a trouvé
  5- L'agent moniteur crée un agent local dans chaque petite image.
  6- L'agent moniteur crée un agent interface.
  7- L'agent local crée la carte de régions de l'imagette associée.
  8- L'agent local extrait les pics de ça carte de région.
  9- L'agent local crée la carte de contours de l'imagette associée.

La (Figure IV.21) correspond à un Diagramme de séquence qui illustre l'initialisation de la segmentation de l'image

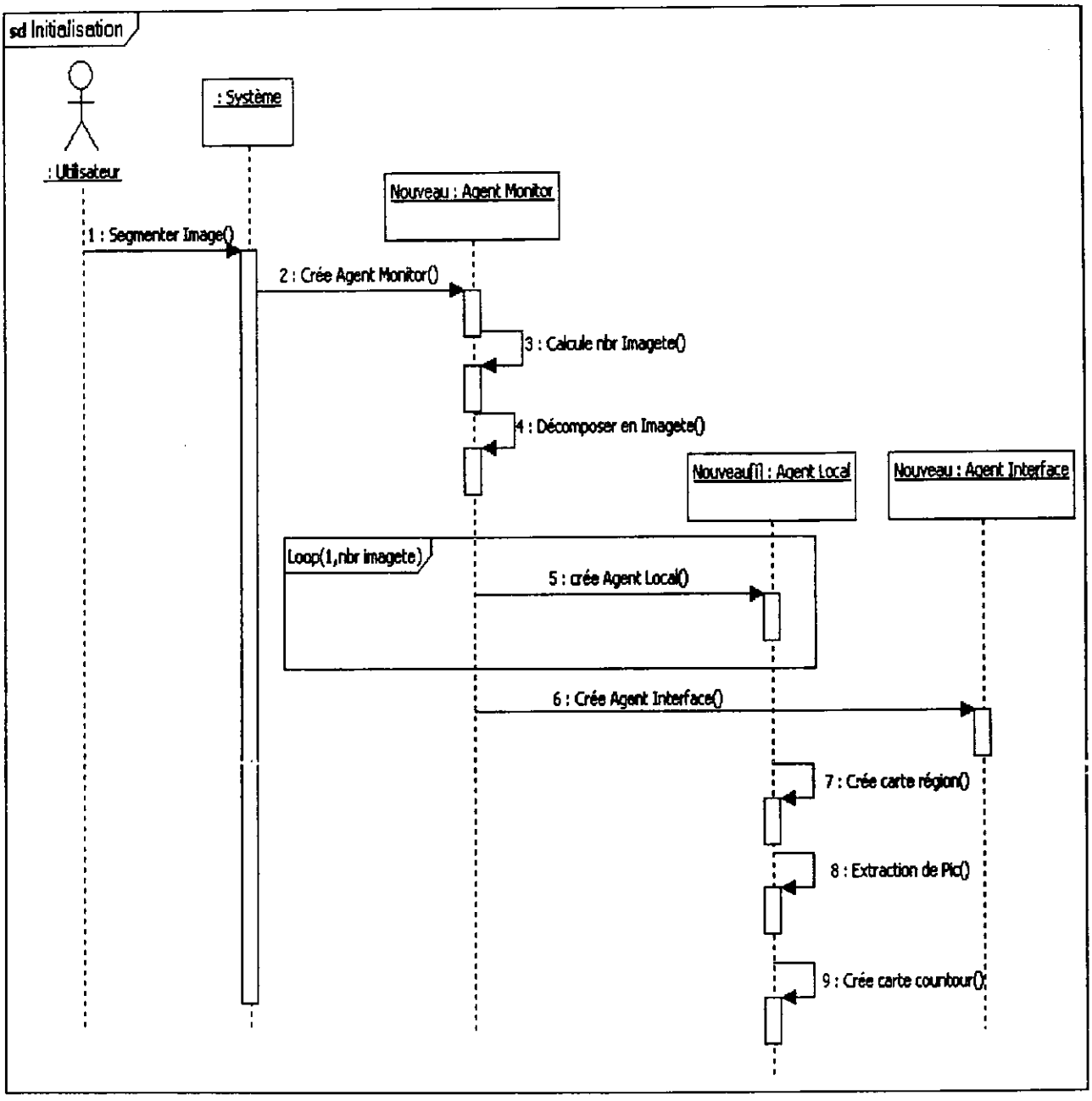


Figure IV.21 : Diagramme de séquence pour l'initialisation de la segmentation de l'image

- **Le Scénario de début de la segmentation**

1- L'agent local crée un agent contour au niveau de l'imagette.
2- L'agent local crée des agents région autant le nombre de pics extraits.
3- Chaque agent région calcule la liste des pixels voisins.
4- L'agent région crée un agent région au niveau de chaque pixel voisin.
5- L'agent région cherche le meilleur voisin en calculant le critère de désir.
6- L'agent région envoie une demande de fusion au meilleur agent voisin.
7- L'agent région (le meilleur voisin du premier agent région) teste si l'agent contour est dans son voisinage ou pas, S'il est dans son voisinage alors (8), (9). Sinon (11), (13).
8- L'agent région envoie une demande d'autorisation à l'agent contour.
9- L'agent contour calcul son voisinage s'il trouve que les deux régions qui veulent fusionner appartiennent a son voisinage alors (10). Sinon (12).
10- L'agent contour en voie un refus de fusion pour l'agent région.
11- L'agent région refus de fusionner.
12- L'agent contour envoie une autorisation de fusion pour l'agent région.
13- L'agent région accepte de fusionner.
14- L'agent région calcul le critère de survie (la superficie de la région et l'agent région délégué).
15- L'agent région fait la fusion.
16- L'agent région met à jour sa liste des pixels voisins.
17- L'agent région refus de fusionner.
18- L'agent région accepte de fusionner.
19- L'agent région calcul le critère de survie (la superficie et l'identifiant de la région).
20- L'agent région fait la fusion.
21- L'agent région met à jour sa liste des pixels voisins.
22- Après la fusion, l'agent région qui a survécu envoie une demande à son agent Local pour lui demander d'éliminer agent région qui a échoué de survivre.
23- L'agent local élimine l'agent région qui n'a pas été survécu.

La (Figure IV.22) correspond à un Diagramme de séquence qui illustre le début de la segmentation.

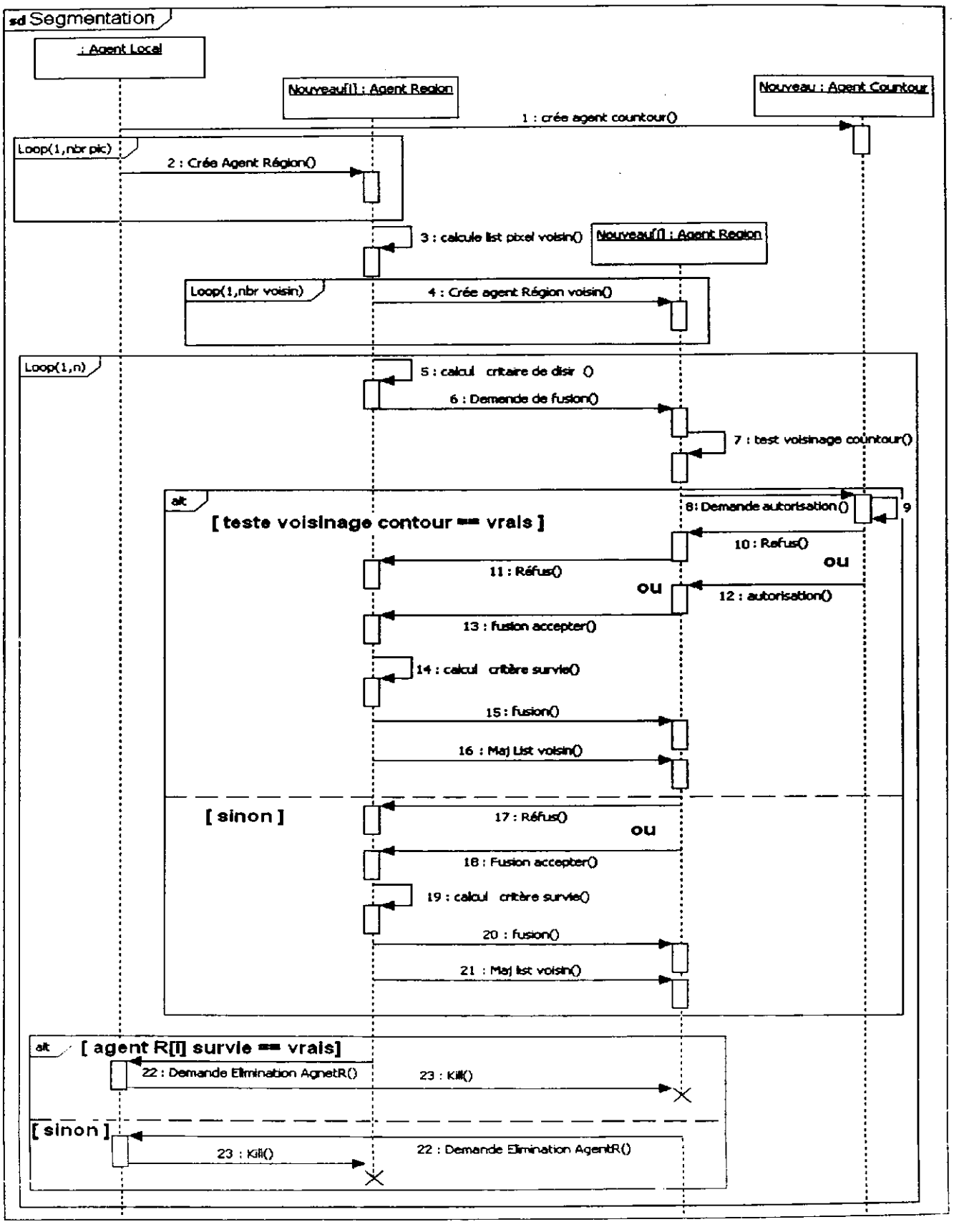


Figure IV.22 : Diagramme de séquence pour le début de la segmentation

- **Le Scénario de la finalisation de la segmentation**

1. L'agent Contour reste sur l'écoute, il teste à chaque fois si la boite de réception est vide, si elle est vide alors (2), Sinon il ne fait rien.
2. L'agent Contour envoie à son agent local un message de fin de segmentation.
3. L'agent Région reste sur l'écoute, il teste à chaque fois si la boite de réception est vide, si elle est vide alors (4), Sinon il ne fait rien.
4. L'agent Région envoie à son agent local un message de fin de segmentation.
5. L'agent Local attend jusqu'à ce qu'il reçoit les messages de fin de segmentation de la totalité des agents région et l'unique agent contour, puis il envoie à son tour un message de fin de segmentation à l'agent Moniteur.
6. L'agent Moniteur envoie un message à l'agent interface pour l'informer que la segmentation est finie et qu'il peut la récupérer au niveau du Blackboard.

La (Figure IV.23) correspond à un Diagramme de séquence qui illustre Diagramme de séquence pour la finalisation de la segmentation

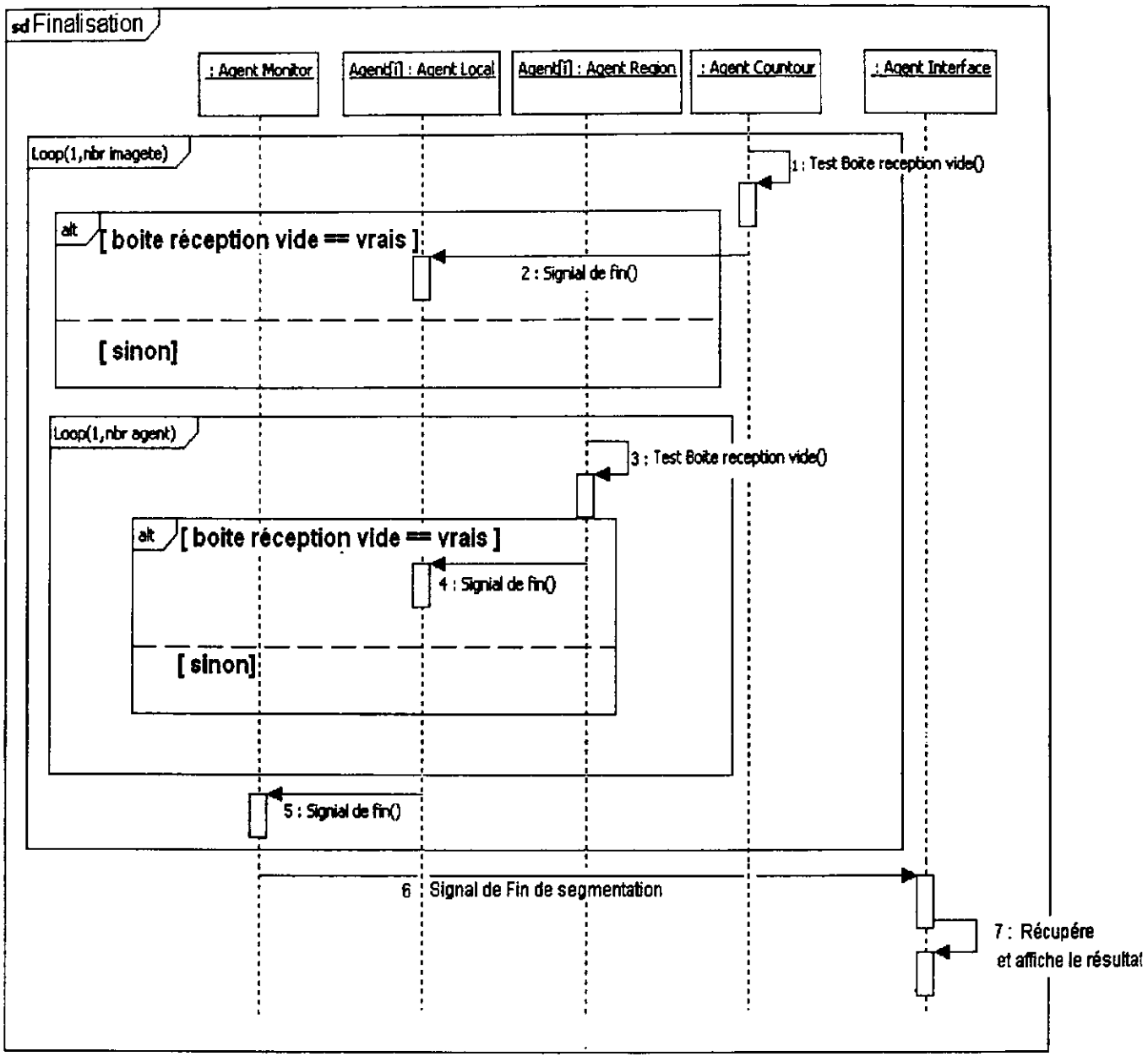


Figure IV.23 : Diagramme de séquence pour la finalisation de la segmentation

III.3. Diagrammes de cas d'utilisation pour la demande de l'état de la Segmentation

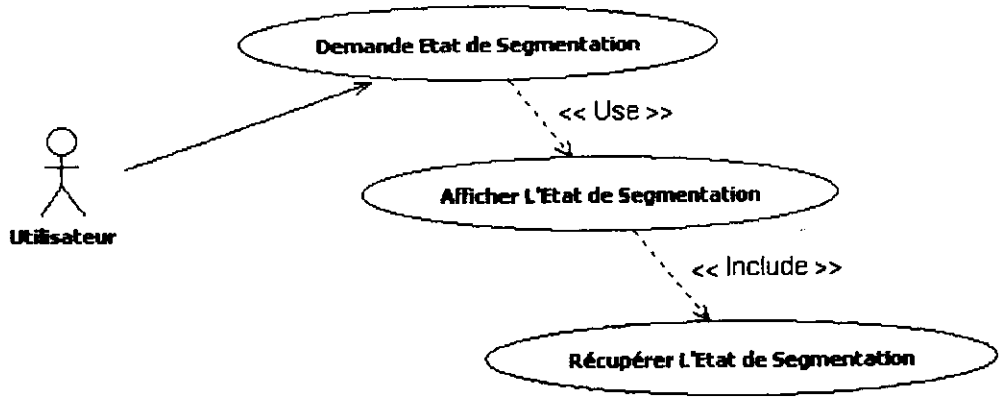


Figure IV.24 : Diagramme de cas d'utilisation pour la demande de l'état de segmentation.

Diagramme des séquences

- **Le Scénario demande Etat de segmentation**

1- L'utilisateur demande un état de segmentation de l'image, donc il lance l'agent interface.
2- L'agent interface envoie un message à l'agent moniteur pour lui demander l'état d'avancement de la segmentation.
3- L'agent moniteur accède au blackboard pour récupérer l'état de la segmentation.
4- L'agent moniteur envoie un message pour l'agent interface pour l'informer que l'état est disponible.
5- L'agent interface accède au blackboard pour récupérer l'état puis il l'affiche à l'utilisateur.
6- L'agent interface affiche le résultat sur l'écran.

La (Figure IV.25) correspond à un Diagramme de séquence qui illustre la demande de l'état de la segmentation

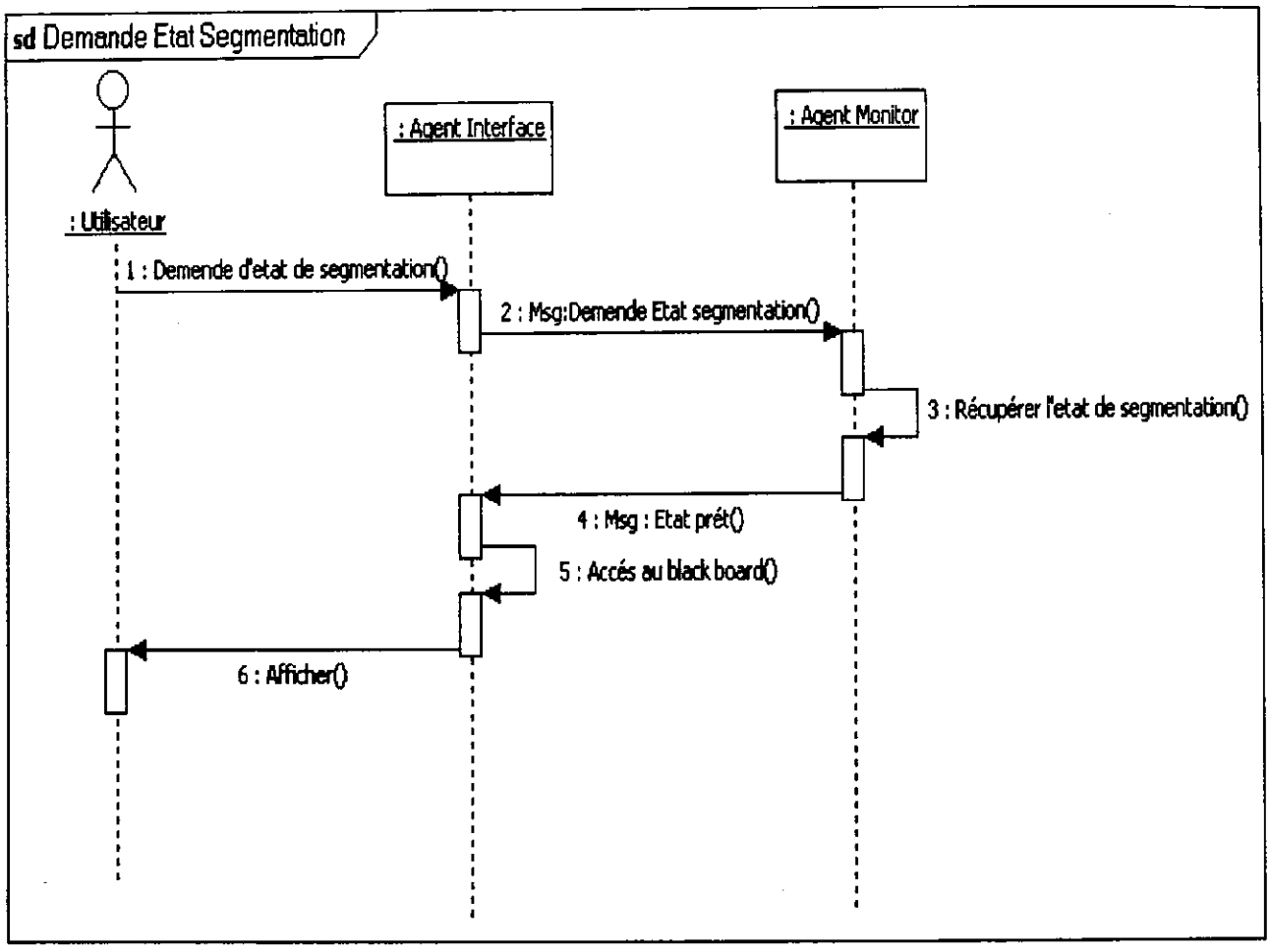


Figure IV.25: Diagramme de séquence pour la demande de l'état de la segmentation

IV. DIAGRAMME DE CLASSE

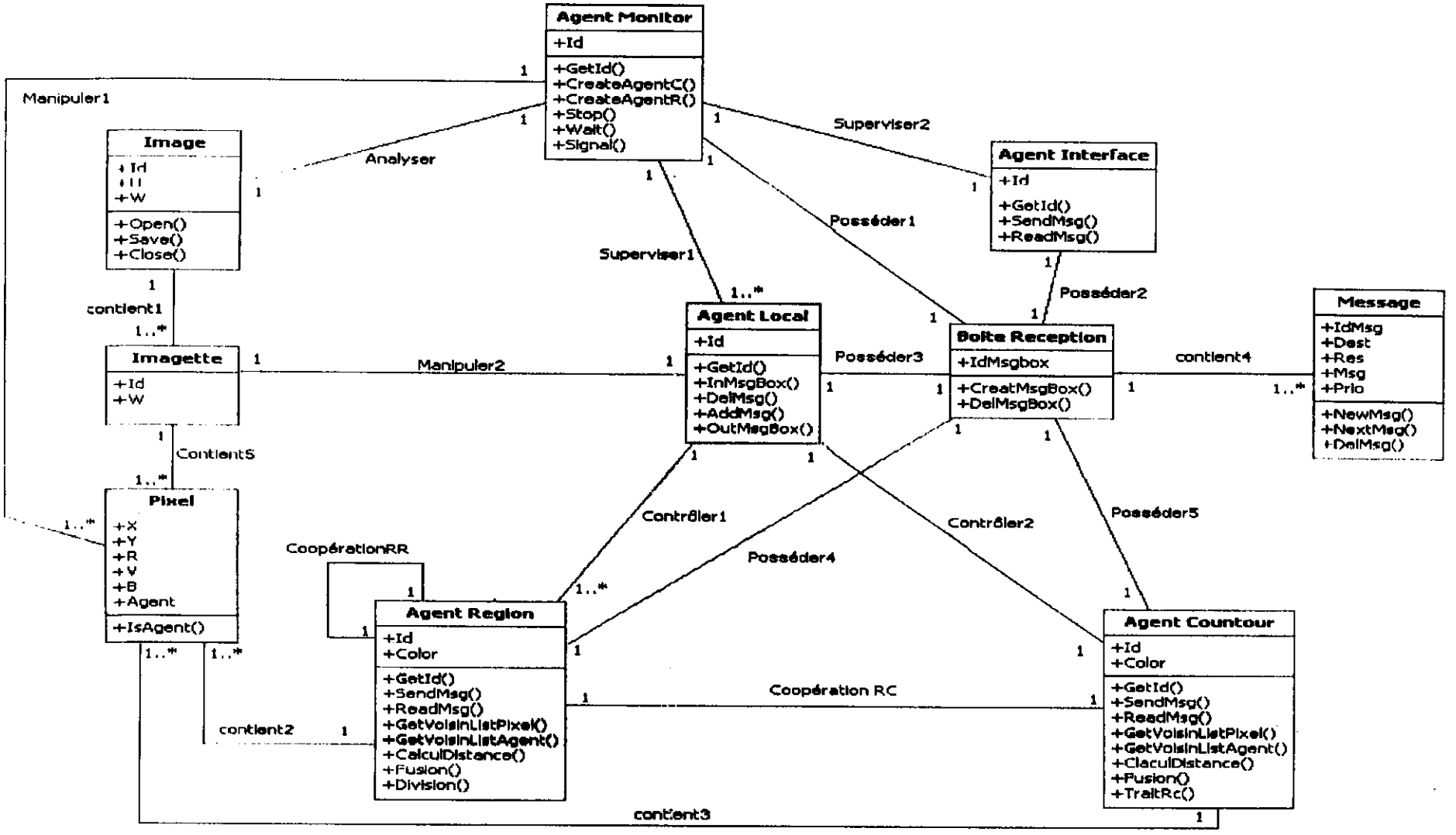


Figure IV.26 : Diagramme de classe du système globale.

IV.1. Explication du diagramme de classes

Notre système contient les classes suivantes :

- La classe Agent Monitor pour superviser tout les agents qui vont participer à la segmentation.
- La classe Agent Local pour contrôler les agents région et les agents contour de chaque imagette.
- La classe Agent Region pour faire la segmentation région
- La classe Agent Contour pour faire la segmentation contour
- La classe Agent Interface permet d'afficher le résultat de la segmentation
- La classe Boite Réception permet aux différents types d'agent de communiquer entre eux.
- La classe Message pour le stockage des message envoyés
- La classe Image contienne la taille de l'image
- La classe Imagette contienne la taille de le petite image (imagette) après décomposition
- La classe Pixel permet de localiser le pixel dans l'image, connaître son intensité et l'agent pour le quel le pixel l'appartient.

IV.2. Règles de gestion

- Un agent moniteur peut superviser un ou plusieurs agents locaux, un agent local peut être supervisé par un et un seul agent moniteur.
- Un agent moniteur peut superviser un et un seul agent interface, un agent interface peut être supervisé par un et un seul agent moniteur.
- Un agent moniteur peut posséder une et une seul boite de réception, une boite de réception peut être possédée par un et un seul agent moniteur.
- Un agent moniteur peut analyser une et une seule image, une image peut être analysée par un et un seul agent moniteur.
- Un agent moniteur peut manipuler un plusieurs pixels, un pixel peut être manipulé par un et un seul agent moniteur.
- Un agent local peut posséder une et une seul boite de réception, une boite de réception peut être possédée par un ou plusieurs agents locaux.

- Un agent local peut manipuler une et une seule imagette (petite image), une imagentte peut être manipulé par un et un seul agent local.
- Un agent local peut contrôler un ou plusieurs agents région, un agent région peut être contrôlé par un et un seul agent local.
- Un agent local peut contrôler un et un seul agent contour, un agent contour peut être contrôlé par un et un seul agent local.
- Un agent interface peut posséder une et une seul boite de réception, une boite de réception peut être possédée par un et un seul agent interface.

- Un agent région peut posséder une et une seul boite de réception, une boite de réception peut être possédée par un et un seul agent région.
- Un agent contour peut posséder une et une seul boite de réception, une boite de réception peut être possédée par un et un seul agent contour.
- Une boite de réception peut contenir rien ou plusieurs message, un message est appartient à une et une seule boite de réception.
- Un agent région peut coopérer avec un autre agent région.
- Un agent région peut coopérer avec un agent contour.

- Un agent région peut contenir un ou plusieurs pixels.
- Un agent contour peut contenir un ou plusieurs pixels.
- un pixel peut être occupé par un et un seul agent région ou un et un seul agent contour.
- Une image peut être décomposée en une ou plusieurs imagette, une imagette appartient à une et une seule image.
- Une imagette peut contenir un ou plusieurs pixels, un pixel appartient à une et une seule imagette.

V. DIAGRAMMES D'ETATS

Pour chaque agent nous allons représenter le diagramme d'état suivant:

V.1. Diagramme d'état pour l'agent Moniteur

L'agent moniteur est responsable du déclanchement du système. Il est crée et lancé par l'utilisateur. Dés qu'il est prêt, il crée un agent interface qui va être responsable de l'affichage du résultat. Selon la taille de l'image il calcule le nombre de divisions possibles. L'image est décomposée ensuite en imagettes pour les quelles un agent local est crée pour chacune.

L'agent moniteur change d'état, il part de celle de création vers l'état d'écoute. Il est en phase de lecture continue des messages envoyés par les agents locaux ou l'agent interface. La figure ci-dessous illustre le diagramme d'état pour l'agent Moniteur.

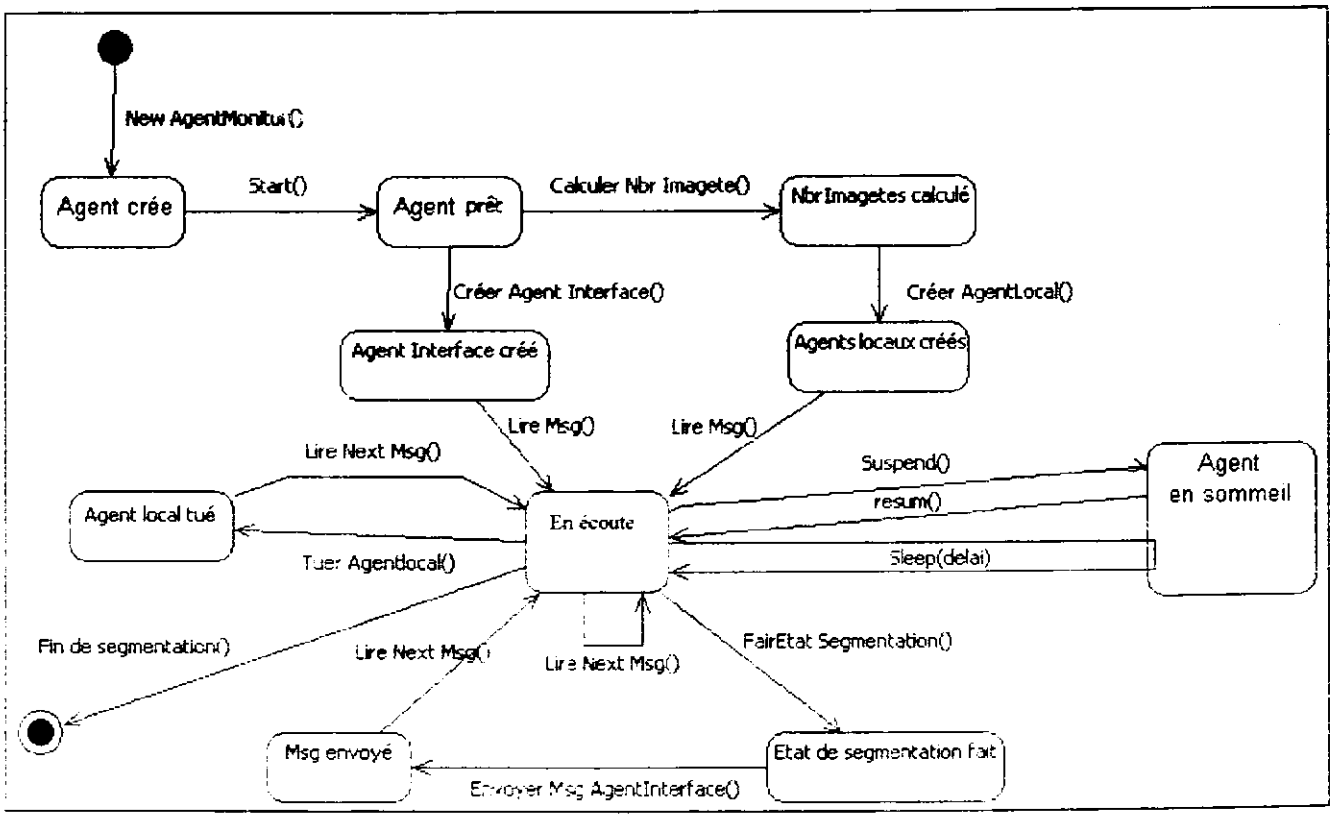


Figure IV.27 : Diagramme d'état pour l'agent Moniteur

V.2. Diagramme d'état pour l'agent Local

Comme on l'a dit précédemment l'agent Local a été crée par l'agent moniteur dans le but de manager des agents délégués qui vont collaborer à leur tour pour segmenter chaque imagette.

Dés que l'agent local est prêt, Il construit la carte de contour de sa petite image puis il crée un agent contour; ce dernier est responsable des contours de la petite image. En même temps, l'agent local construit aussi la carte régions de l'imagette, il extrait les pics des régions de la carte puis il crée un agent région délégué sur chacun des pics.

Apres tout ce traitement, l'agent local passe à l'état d'écoute pour qu'il puisse géré ces agents délégué soit par l'élimination des agents régions après les fusions, ou bien par l'envoi d'un message de fin de segmentation à l'agent moniteur, sinon il reste a l'écoute. La figure ci-dessous illustre le diagramme d'état pour l'agent du contrôle local.

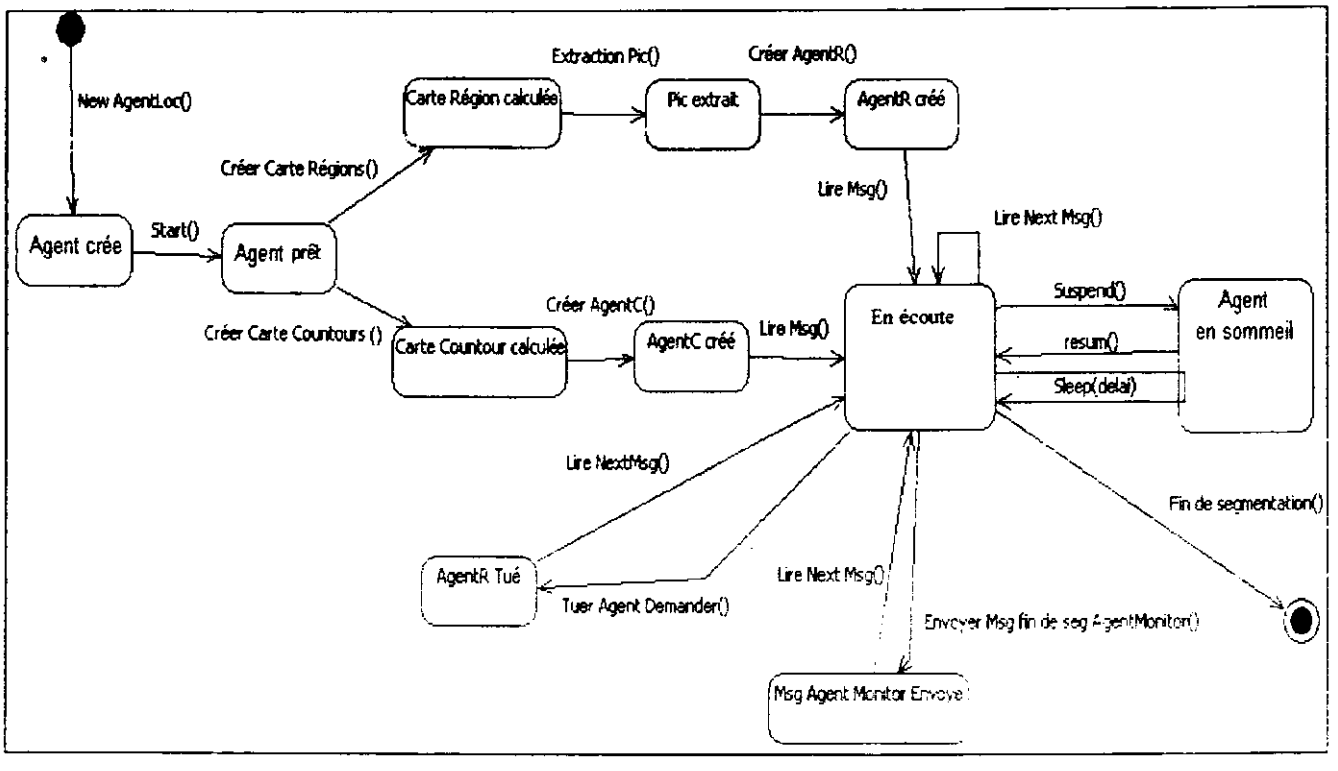


Figure IV.28 : Diagramme d'état pour l'agent du contrôle local

V.3. Diagramme d'état pour l'agent Interface

L'agent interface a été crée pour qu'il reste à l'écoute. Il reçoit des messages soit de l'extérieur (utilisateur) qui demande l'état de segmentation, soit de l'agent moniteur qui lui demande de récupère l'état ou niveau du blackboard. L'unique traitement que l'agent interface fait c'est de demander l'état de la segmentation à l'agent moniteur puis il récupère l'état et il affiches l'image sur écran. La figure ci-dessous illustre le diagramme d'état pour l'agent interface.

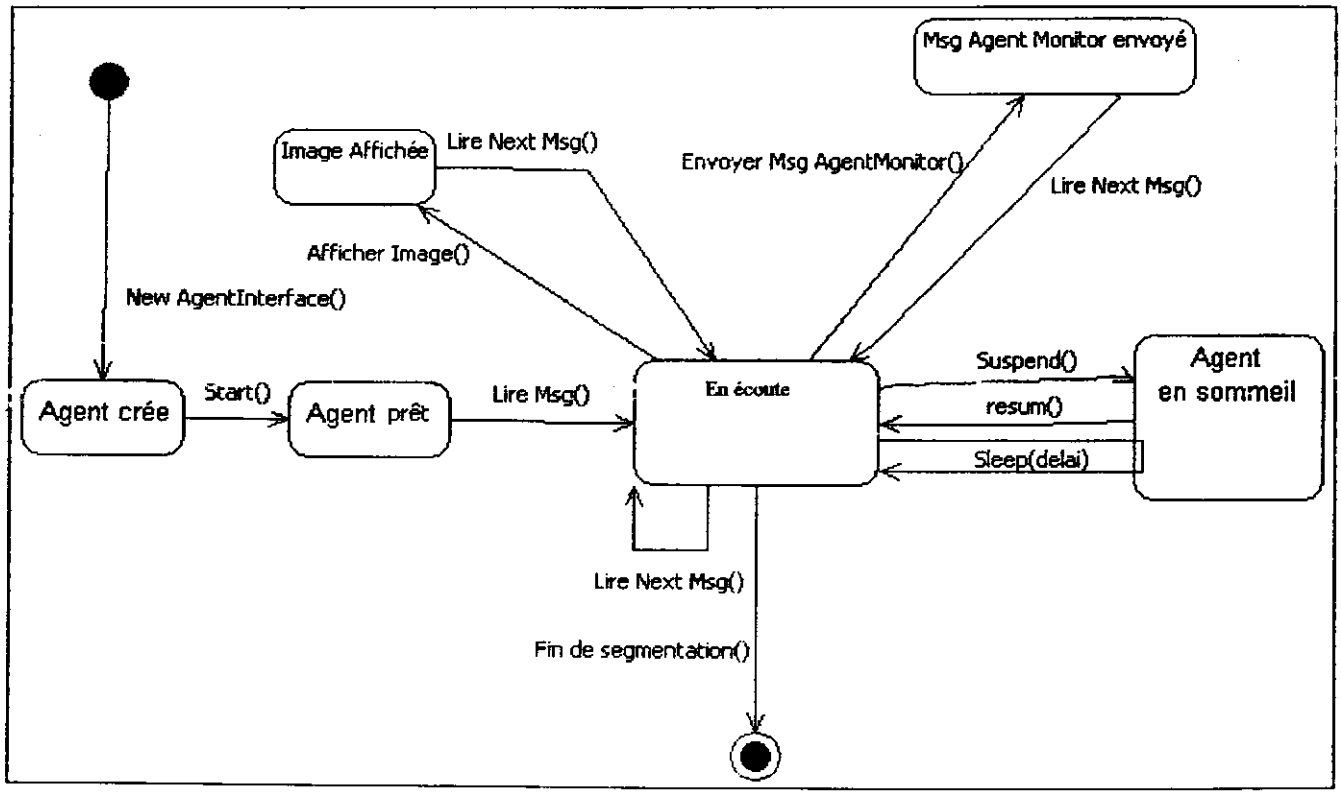


Figure IV.29 : Diagramme d'état pour l'agent Interface

V.4. Diagramme d'état pour l'agent Région

Après le lancement de l'agent région il sera prêt à construire sa liste de pixels voisins, il crée un agent région sur chaque pixel voisin. En calculant les critères de désire l'agent région peut choisir le meilleur voisin et lui envoyer une demande de fusion puis il passe a l'état d'écoute. Durant cet état l'agent région peut recevoir (Figure III.35):

- une demande de fusion,
- un refus à sa demande de fusion,
- un accord à sa demande de fusion,
- une autorisation de fusion,
- une interdiction de fusion.

Dans chaque fusion, l'agent région calcule le critère de survie. En cas de survie, il doit mettre à jour sa liste de pixels voisins pour continuer le processus, sinon il sera tué par son créateur (agent local). La figure ci-dessous illustre le diagramme d'état pour l'agent région.

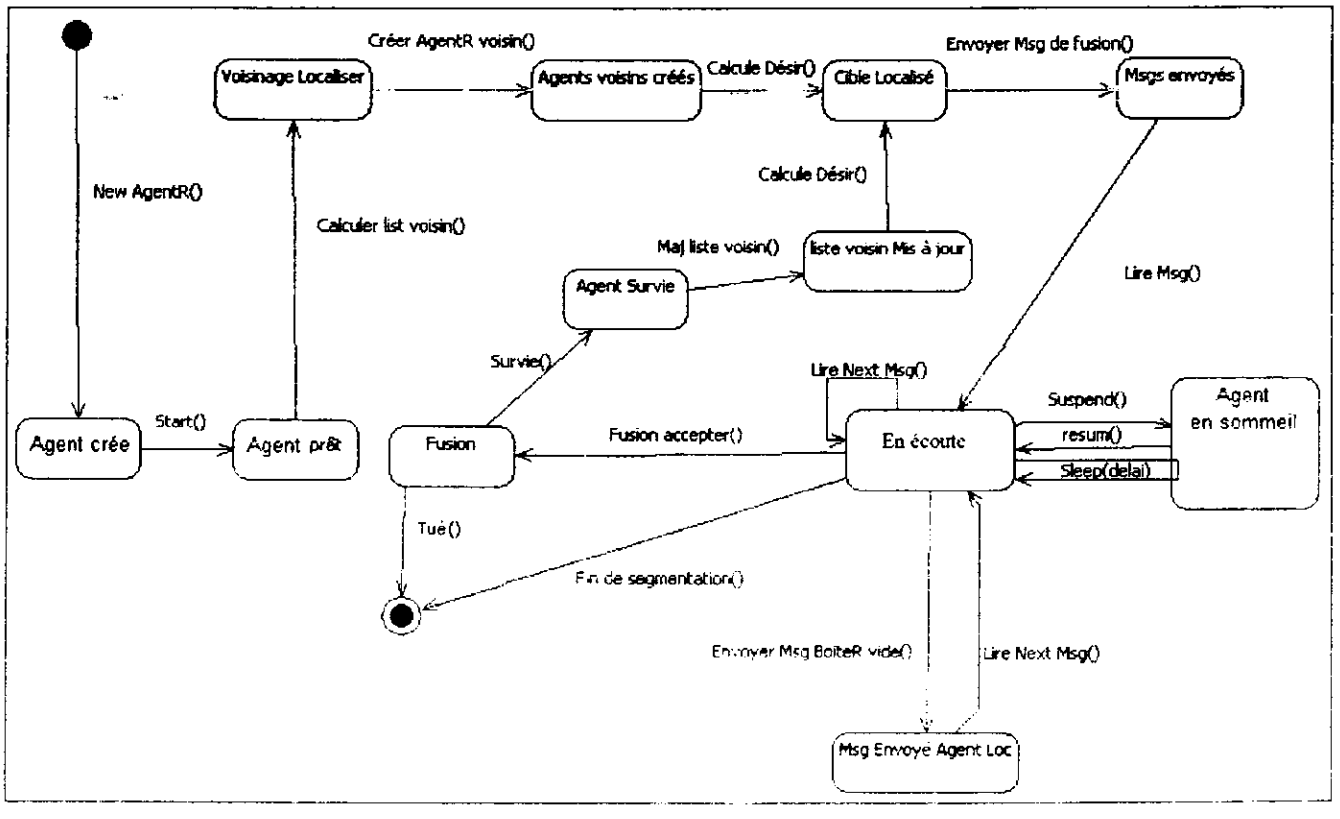


Figure IV.30 : Diagramme d'état pour l'agent Région

V.5. Diagramme d'état pour l'agent Contour

L'agent contour dés qu'il est prêt, il passe à l'état d'écoute pour voir s'il y a une demande d'autorisation de fusion. Dans ce cas il vérifie son voisinage s'il trouve qu'il est voisin des deux régions qui demandent la fusion, alors il interdit la fusion, sinon il l'autorise. Puis il retourne à sont état d'écoute. Dans le cas contraire, c'est-à-dire pas de message (boite de réception vide) il envoie un message à l'agent local pour l'informer que les fusions sont achevées (Figure IV.31).

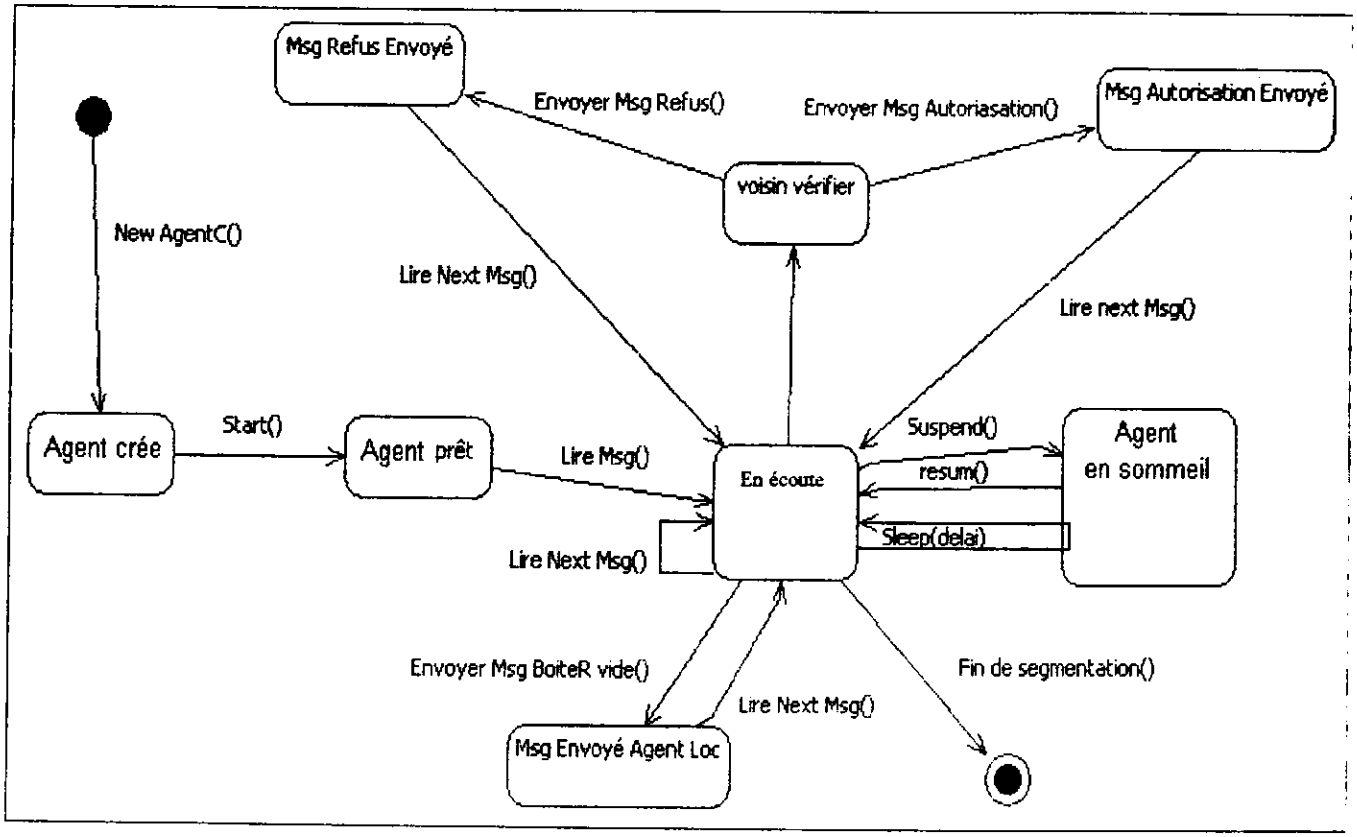


Figure IV.31: Diagramme d'état pour l'agent Contour

IV. DIAGRAMME D'ACTIVITE

Ce diagramme d'activité représente les différents composants de système. Il se divise en quatre activités principales:

- **initialiser la segmentation** : pour réaliser cette tâche il faut créer un agent moniteur qui va créer à son tour un agent interface et le mettre à l'écoute et des agents locaux qui vont gérer la segmentation.
- **segmentation** : pour réaliser cette tâche il faut créer des agents contours et des agents région qui vont collaborer pour finaliser la segmentation.
- **finalisation de la segmentation** : après la réalisation de cette étape il faut faire appel à l'agent interface qui est en écoute, pour qu'il récupère l'état de segmentation et l'afficher par la suite.
- **demande l'état de la segmentation** (cette étape peut se dérouler en parallèle avec les autres étapes, comme elle peut se présenter à la fin. Pour réaliser cette tâche, il suffit d'appeler l'agent interface qui est en écoute, pour qu'il récupère l'état de segmentation et l'afficher par la suite. Toutes ces activités sont résumées au niveau de la (Figure IV.32).

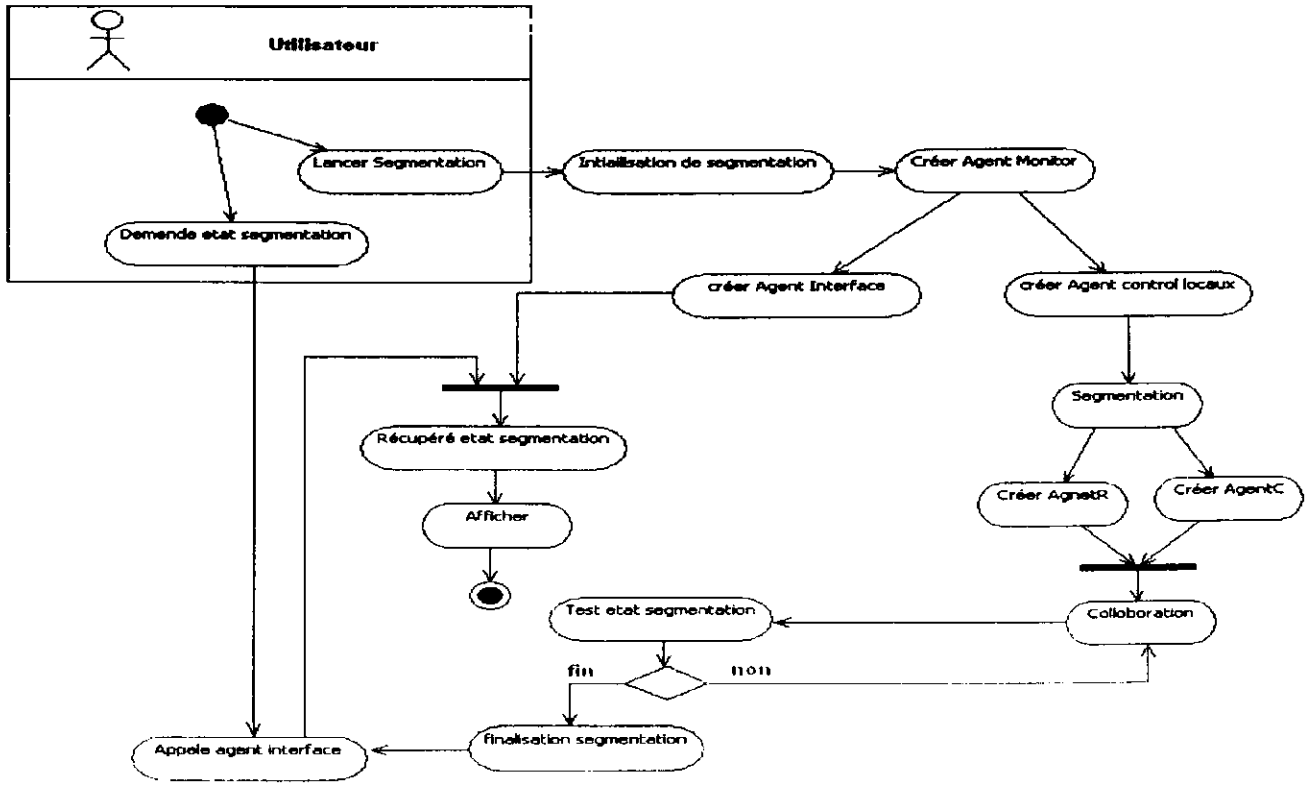


Figure IV.32 : Diagramme d'activité de système.

VII. DIAGRAMME DES COMPOSANTS

Ce diagramme va nous permettre de montrer les différentes parties du système. Notre système est composé de cinq composants qui représentent les agents du système, et un blackboard pour partager les ressources entre agents. Le Blackboard ou tableau noir, contient les pixels qui composent l'image, la boite de réception de chaque agent et l'état du système (Figure IV.33).

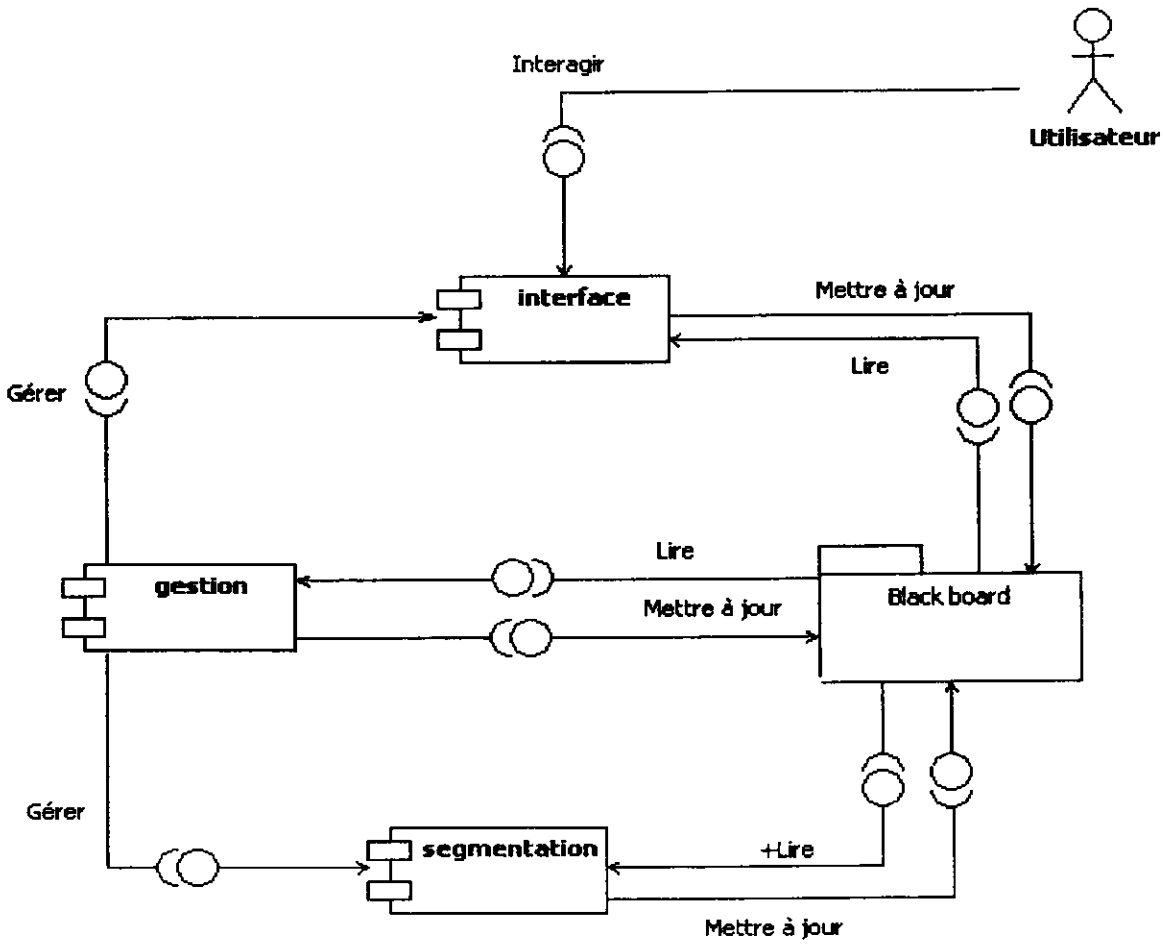


Figure IV.33 : Diagramme des composants.

VIII. CONCLUSION

Dans ce chapitre nous avons présenté les trois premiers modèles du processus unifié UP (modèle de cas d'utilisation, modèle d'analyse, modèle de conception) qui montrent les différents diagrammes qui modélise notre système. Dans le chapitre suivant on découvre ensembles les modèles: implémentation, déploiement et teste du processus UP.

CHAPITRE 5

Implémentation et Résultats

I.INTRODUCTION

Dans ce chapitre, nous allons mettre en œuvre le système multi-agents qui réalise les différentes étapes de notre approche.

Ce système comporte des agents réactifs qui se divisent en deux catégories: gestion et segmentation. Ces agents communiquent entre eux via un tableau noir (BlackBoard) pour réaliser la segmentation de l'image.

II. LES OUTILS DE DEVELOPPEMENT

II.1.Matériel utilisé

Notre application a été développé sur un micro-ordinateur muni essentiellement de :

- ❖ Un processeur Pentium 4 3.4Ghz avec 2Mb de cache
- ❖ Une carte mère Gigabyte 945 Fsb 1066
- ❖ Une Ram de capacité 2Go
- ❖ Un disque dur de 250 Go
- ❖ Une carte graphique GeForce 7100 Gs

II.2. Langage de programmation

Plusieurs langages sont destinés à la conception d'agents, dont Java qui est le langage adopté dans la mise en œuvre de notre application.

II.2.1. Pour quoi choisir JAVA ?

Nous avons choisi le langage java comme langage de développement pour plusieurs raisons dont :

- Java 100 % portable sur toute plate forme, il est possible d'utiliser le même code pour Windows 98/NT/Vista, Solaris UNIX, Macintosh, etc.
- JAVA est libre et gratuit, offrant de très nombreuses librairies, facilement réutilisables.
- l'allocation et la libération de mémoire se fait d'une manière automatique dans java.

- Java est fiable, il intègre un modèle de pointeur qui écarte les risques d'écrasement de la mémoire et d'endommagement des données.
- Java possède des fonctions destinées à éliminer la possibilité de créer du code contenant les types d'erreurs les plus courants.
- Java est un langage de programmation orienté objet qui répond aux différents aspects du concept orienté objet tel que *L'encapsulation, Le polymorphisme et L'héritage*.
- Java prend en charge les multithread qui offrent une meilleure interactivité et un meilleur comportement en temps réel.
- Java permet la construction de systèmes inaltérables et sans virus.

Vu que La plupart des plateformes multi agents sont développées en JAVA, nous avons pensé à implémenter notre système multi agents avec java Myeclips.

Myeclips est un IDE (Integrated Development Environment), c'est-à-dire un environnement de développement intégré dont le but est de fournir une plate forme modulaire pour permettre de réaliser des développements informatiques **[Ref, 03].**

III. IMPLEMENTATION DU SYSTEME

Dans ce qui suit, nous allons présenter l'implémentation de l'approche de segmentation coopérative en utilisant les systèmes multi agents.

Notre système est composé de plusieurs modules dont :

- module analyse de l'histogramme,
- module détection contour,
- module gestion de SMA,
- module Segmentation,
- module interface.

III.1. Module analyse de l'histogramme

Nous avons utilisé Une analyse de l'histogramme par la méthode d'Otsu afin de définir le nombre de classes existantes dans l'image originale. Ce module permet aussi de localiser les pixels qui désignent les pics de chaque classe, ces informations seront utilisées pour initialiser le nombre d'agents région ainsi que leurs emplacements.

La figure ci-dessous illustre la forme de la classe Otsu.

```
package Logi;

import java.awt.*;

import com.sun.org.apache.xpath.internal.operations.Div;

public class Otsu extends Frame
    {
        nbreClasses = i;
        menu = menu1;

        otsuGraph = new OtsuGraphClass();
        otsuGraph.Init(this);

        CalculHisto();

        otsuGraph.Init();
        CalculClasses();
        SegmenterImage();
        menu1.RetFenetreIm().ValiderImage(matrice);
    }
```

Figure V.01: Classe Otsu

Les méthodes utilisées dans la classe Otsu:

- CalculHisto(): elle permet de calculer l'histogramme de l'image,
- CalculClasses(): Elle constitue principale de la segmentation d'image pour l'analyse d'histogramme. Elle calcule les seuils séparateurs en maximisant le rapport entre la variance inter-classe et la variance totale,
- SegmenterImage(): Elle affecte des pseudos couleurs aux classes extraites,
- ValiderImage(): Elle permet de valider l'image segmenter, cette méthode a comme paramètres la matrice de l'image initiale.

Extension de la méthode d'OTSU

Nous avons présenté précédent la méthode Otsu, qui donne un résultat satisfaisant dans de cas d'une binairisation, mais elle est trop limitée pour les histogrammes multi classes (plus que 2 classes) d'où la nécessité d'une extension pour la méthode.

Pour atteindre ce but on a choisi comme critère de seuillage la maximisation de la somme des variances interclasses, et ce là nécessite que chaque variance inter classe soit maximale, ce qui satisfaire le critère d'OTSU. La nouvelle fonction critère à maximiser est donnée par la formule suivante:

$$J(p) = \frac{\sigma_{\beta}^{2}}{\sigma_{T}^{2}}$$

D'où:

- p = histogramme normalisé,
- σ_{β}^{2} = variance interclasse,
- σ_{T}^{2} =variance total,
- n = nombre de classe.

III.2. Module détection contour

Une carte de contours est calculée par la méthode de Dériche, pour définir les l'agent contour et son emplacement.

La figure ci-dessous illustre la forme de la classe Dériche.

```
package Logi;
import java.awt.*;

public class Deriche extends Frame{

    menu = menu1;
    largeur = menu1.RetFenetreIm().RetLargeur().intValue();
    hauteur = menu1.RetFenetreIm().RetHauteur().intValue();

    Norm(d);
    MajPix();
    affiche();

    menu1.RetFenetreIm().ValiderImage(matrice);

}
```

Figure V.02: Classe Dériche

Les méthodes utilisées dans la classe Dériche:

- Norm(): elle permet d'extraire les contours de l'image
- MajPix(): mise à jour de la liste des pixels affecté aux contours,
- affiche(): affichage de la carte contour.

III.3. Module gestion de SMA

Ce module contient deux classes, une pour la gestion et la coordination de système (FigureV.03), et l'autre pour garantir la cohérence globale de la segmentation à partir des modèles locaux (Figure V.04).

La figure ci-dessous illustre la forme de la classe de l'agent moniteur.

```
package ALLAgent;
import java.awt.Menu;
import java.lang.reflect.Type;
import Logi.FenetreIm;
import Logi.Menu1;
import java.util.*;

public class MonitorAgent extends Thread{

    public static int w;
    public static int h;
    public static int [][] Img;
    public static int [][] TT;
    public static  int im_nbr;
    public static int Aloc[][];
    public static int Abox1 [][];public static int Abox2 [][];public static int Abox3 [][];
    public static String Abox4 [][];public static int Abox5 [][];
    public static int Aprior[][];public static int Temp[];
    public static int [] vois;
    public static int [][] Avois;

    public void run(){
        Abox1 = new int[500][w*h+1];Abox2 = new int[500][w*h+1];Abox3 = new int[500][w*h+1];
        Abox4 = new String[500][w*h+1];Abox5 = new int[500][w*h+1];
        Aprior = new int[w*h+1][w*h+1];Temp=new int[w*h+1];
        vois = new int[9];
        Avois =new int[w*h+1][w*h+1];

        Int_Pixel(FenetreIm.RetLargeur(),FenetreIm.RetHauteur());
        init_Msgbox();
        cali_nbr_imag();
        Aloc=new int[im_nbr+1][3];
        InitAgentLoc();
        createAgentLoc(im_nbr);

            while(true){
            readmsg();
            HandelMsg(Msg1);
            }
    }
```

Figure V.03: Classe agent Moniteur

Les méthodes utilisées par cette classe sont:

- Int_Pixel(): Elle permet d'initialiser la liste des pixels, leurs position(x,y) et valeurs d'intensité.
- Init_Msgbox(): elle initialise les boites aux lettres.
- Cali_nbr_imag(): elle calcule le nombre d'imagettes dans l'image principale.
- initAgentloc(): elle permet de marque les zones de travail de chaque agent de control local.
- createAgentLoc(): elle est utilisé pour créer les agents des contrôles locaux.
- Readmsg(): elle permet la lecture des messages.
- HandeMsg(): généré un fait (action) ,selon le message reçu.

La figure ci-dessous illustre la forme de la classe de l'agent control local.

```
package ALLAgent;

public class AgentCtrlLocal extends Thread {
    public int P1;  public int P2;  public int P3;

    public void run(){

       Segotsu(this,P1,P2,P3);
       Segdrich(this,P);
       index_pix_imagette(P1,P2,P3);

       try {
          this.sleep(1000);
       } catch (InterruptedException e) {
          e.printStackTrace();
       }

       synchronized(this.getClass()){
          CreateAgentR(P1);
          CreateAgentR(P2);
          CreateAgentR(P3);
          CreateAgentR(P);
       }

       while(true){

       readmsg();
       HandelMsg(Msg1);
       }
       }
```

Figure V.04: Classe agent contrôle local.

Les méthodes utilisées par cette classe sont:

- Segotsu(): c'est un appel de la classe Otsu pour segmenter l'imagette,
- Segdrich(): c'est un appel de la classe Deriche pour détecter les contours de l'imagette,
- Index_pix_imagette(): elle permet de calculée l'indice du pixel dans l'image initiale,
- CreateAgentR(): cette méthode crée les agents région, elle a comme paramètres l'indice du pixel cible,
- CreateAgentC(): cette méthode crée un agent contour.

III.4. Module Segmentation

Ce module représente le noyau de la segmentation coopérative, Il est base sur deux classes :
La figure ci-dessous illustre la forme de la classe de l'agent région.

```
package ALLAgent;

public class AgentR extends Thread{

    private static final String Msg = null;
    int id;
    String color;
    public static int I;
    public static int M;
    public static int M5;
    public static int F5;
    public static int G=1;
    public static int IdMsg=1;
    public static int IdMsg1;
    public static int F1;
    public static int M1;
    public static String Msg1;
    public static int Prio1=1;

    public void run(){

        getindicpix();
        getvoisnlistpixel(MonitorAgent.II[M][2],MonitorAgent.II[M][1]);
        CalculDist();

        while(MonitorAgent.II[M][7]==1){

            readmsg();
            HandelMsg(Msg1);

            if (MonitorAgent.II[M][7]==1){
            System.gc();
            return;
            }

        }

    }
```

Figure V.05: Classe agent Région.

Les méthodes utilisées par cette classe sont:

- getindicpix(): cette méthode retourne l'indice de l'agent région courant ,
- getvoisnlistpixel():cette méthode calcule la liste des pixels voisin de l'agent courant, elle fait appelle a une autre méthode pour crée un agent région sur chacun des pixels voisins,
- CalculDist(): cette méthode calcule la distance euclidien entre l'agent courant et ces agents voisins.

La figure ci-dessous illustre la forme de la classe de l'agent contour.

```
package ALLAgent;

public class AgentC extends Thread {


    public void run(){

        while(true){

            readmsg();
            HandelMsg(Msg1);
        }
        }
```

Figure V.06: Classe agent Contour.

III.5. Module interface

Ce module est responsable de l'affichage de l'image segmenté.

La figure ci-dessous illustre la forme de la classe de l'agent interface:

```
package ALLAgent;

public class AgentInerface extends Thread  {

    public void run(){

      while(true){

            readmsg();
            HandelMsg(Msg1);
        }
      }


}
```

Figure V.07: Classe agent Interface.

IV. INTERFACE DU SYSTEME

L'interface principale est illustrée par la figure ci-dessous :

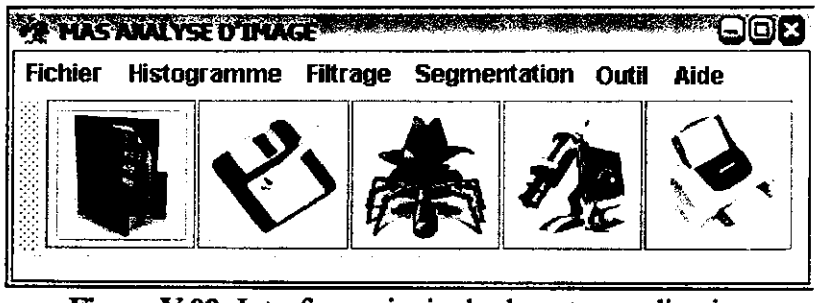


Figure V.08: Interface principale de notre application

On distingue quatre menus illustrés par les figures suivantes :

1. Menu fichier : il contient trois sous menus, tel que :
   - Ouvrir Image,
   - Fermer Image
   - Quitter l'application.

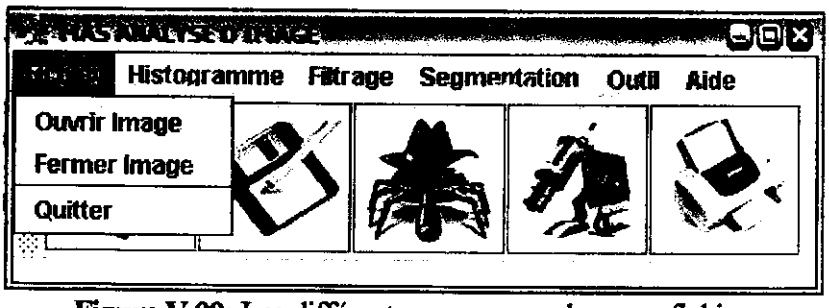


Figure V.09: Les différents sous menus du menu fichier.

2. Menu histogramme : il contient quatre sous menus, tel que :
   - Info sur l'image,
   - Histogramme,
   - Histogramme Cumulé,
   - Egalisation de l'Histogramme.

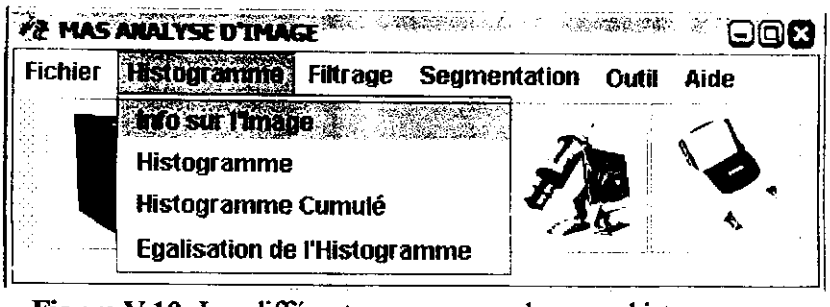


Figure V.10: Les différents sous menus du menu histogramme.

3. Menu filtrage : il contient trois sous menus correspondent a trois filtres optimaux, tel que :
   - filtre Sobel,
   - filtre Prewit,
   - filtre Median.

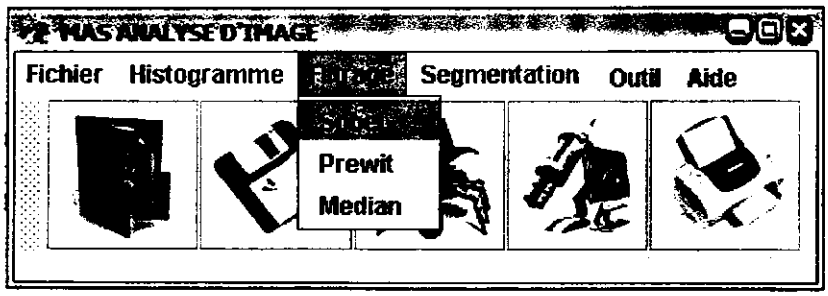


Figure V.11: Les différents sous menus du menu filtrage.

4. Menu segmentation : il contient deux sous menus, tel que :
   - Segmentation manuelle,
   - Segmentation de SMA.

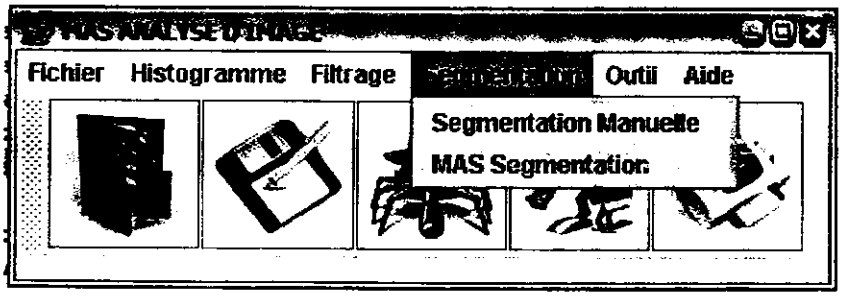


Figure V.12: Les différents sous menus du menu de segmentation.

5. Menu Outils : il contient deux sous menus, tel que :
   - Détecteur de contour Deriche,
   - Classification d'Otsu.

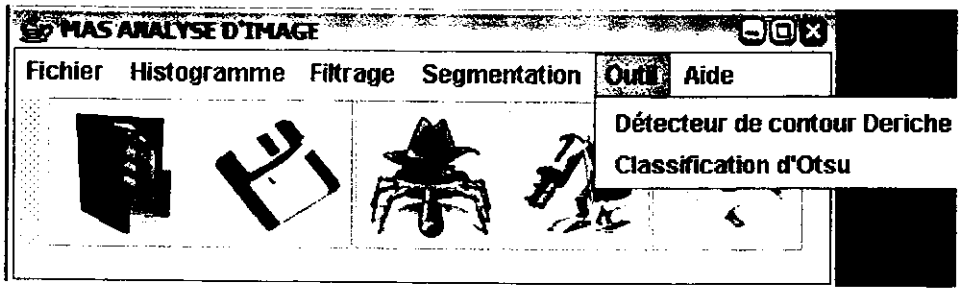


Figure V.13: Les différents sous menus du menu d'Outil.

V. EXPERIMENTATION ET RESULTATS

Nous allons définir ci-dessous quelques paramètres obtenus par expérimentation qui sont utilisés dans notre système:

V.1. Choix et définition des paramètres

- La taille de l'imagette

Le bon choix de la taille des partitions est essentiel, si la taille est trop grande, le nombre d'agents de la segmentation sera très élevé dans l'imagette. Vu qu'on est limité par l'architecture de la machine utilisée et la taille de la mémoire centrale, nous avons choisi une partition de 16x16 pour limiter le nombre d'agents lancé dans une imagette à 256 au maximum.

- Le nombre de classes

Le nombre de classes est un paramètre important pour l'exécution de notre système. L'analyse de l'histogramme de l'image à traiter donne une idée sur la distribution des niveaux de gris ; les maxima locaux correspondent aux centres de classes. Cette opération est faite de manière manuelle.

Les seuils obtenus par cette analyse vont être utilise comme un critère d'arrêt pour le processus de la segmentation coopérative.

- Le paramètre alpha du détecteur de Dériche

L'application de l'opérateur optimal de Dériche sur l'image à traiter, nécessite la définition de paramètre alpha. Ce dernier donne l'intensité du lissage associé à la détection du gradient. Les valeurs sont typiquement dans l'intervalle [0..10]. Plus la valeur est faible, plus le lissage est fort. Une valeur typique est 1,8 obtenu par expérimentation.

V.2. Application des méthodes de segmentation

Après avoir faire une analyse de l'histogramme (Figure V.14 (b)) sur l'image **Test1** (Figure V.14 (a)), on obtient le nombre de classes **Nb=4**, ce dernier sera utiliser dans la méthodes d'**Otsu** le résultat obtenu représente une carte de régions illustré dans la figure(Figure V.14 (d)). Ensuite on applique le détecteur optimale de **Dériche** avec un paramètre **alpha = 1,8** sur l'image originale (Figure V.14 (a)), on obtient une carte de contour illustré dans la figure (Figure V.14 (c)) .Finalement les cartes de contours et de régions sont utiliser par les **SMA** afin d'effectué une segmentation coopérative le résultat est illustré dans la figure (Figure V.14 (e)).

- **Application sur l'image Test1 (Figure V.14).**

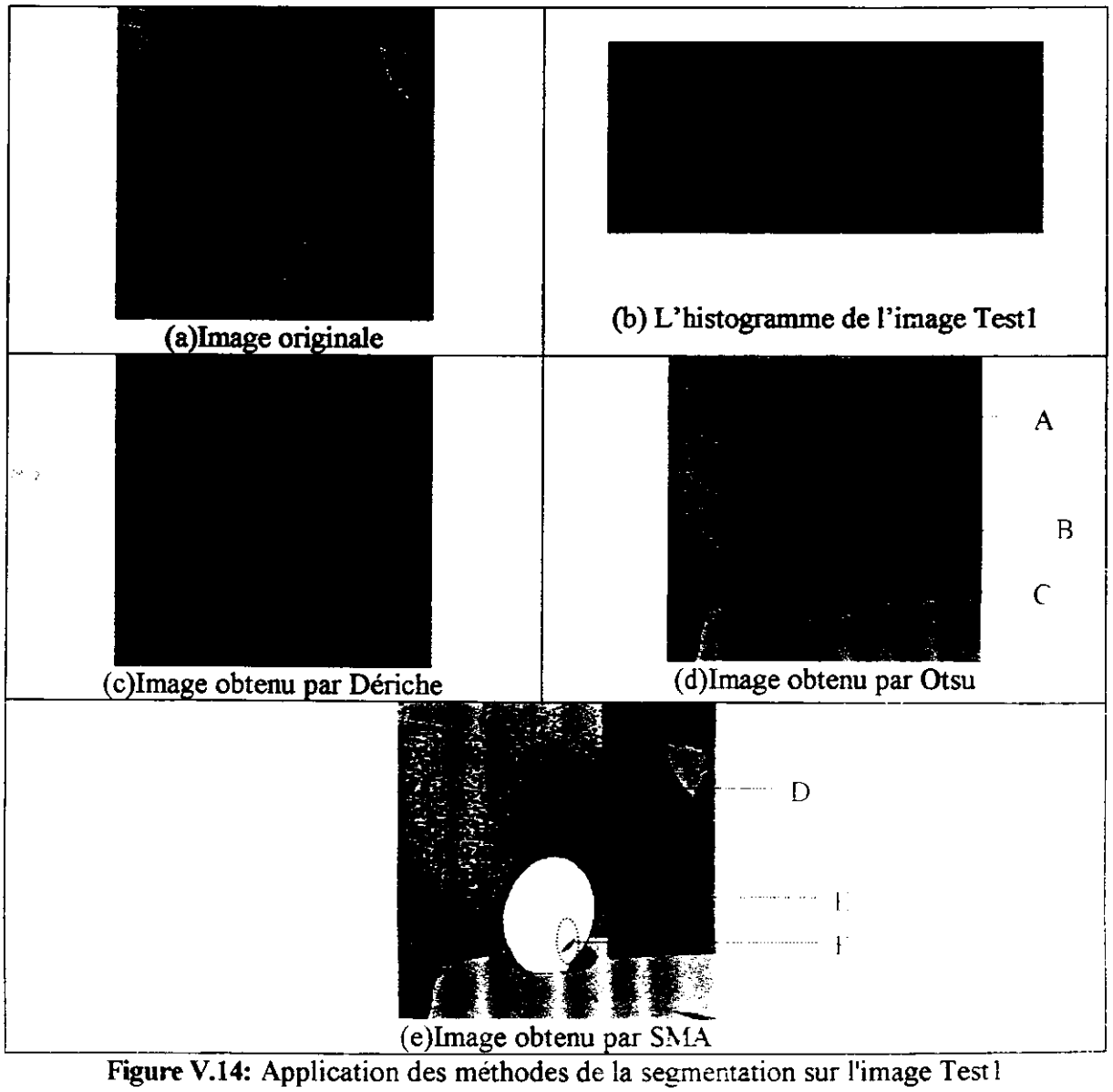


Figure V.14: Application des méthodes de la segmentation sur l'image Test1

Interprétation

En comparant l'image originale et l'image résultante de la segmentation d'Otsu, on trouve qu'il y a une grande perte d'information dans les zones (A), (B) et (C) (Figure V.14 (d)). Par contre l'image segmentée par les SMA présente une bonne affectation des pixels de la zone (D) et (F) (Figure V.14 (e)). Ce qui revient à l'utilisation de l'information contour dans la segmentation par SMA.

L'information contour obtenu par la carte de contour (Figure V.14 (c)) est insuffisante, ce qui induit une perte d'information au niveau de la zone (E) dans l'image segmenté par les SMA (Figure V.14 (e)).

Afin de montré la robustesse de notre système, nous avons appliqué les le processus de segmentation sur les images BUREAU et IRM cérébrale. Les résultats obtenus sont présentés sur les figures suivantes (Figure V.15, Figure V.16).

- **Application sur l'image Bureau (Figure V.15).**

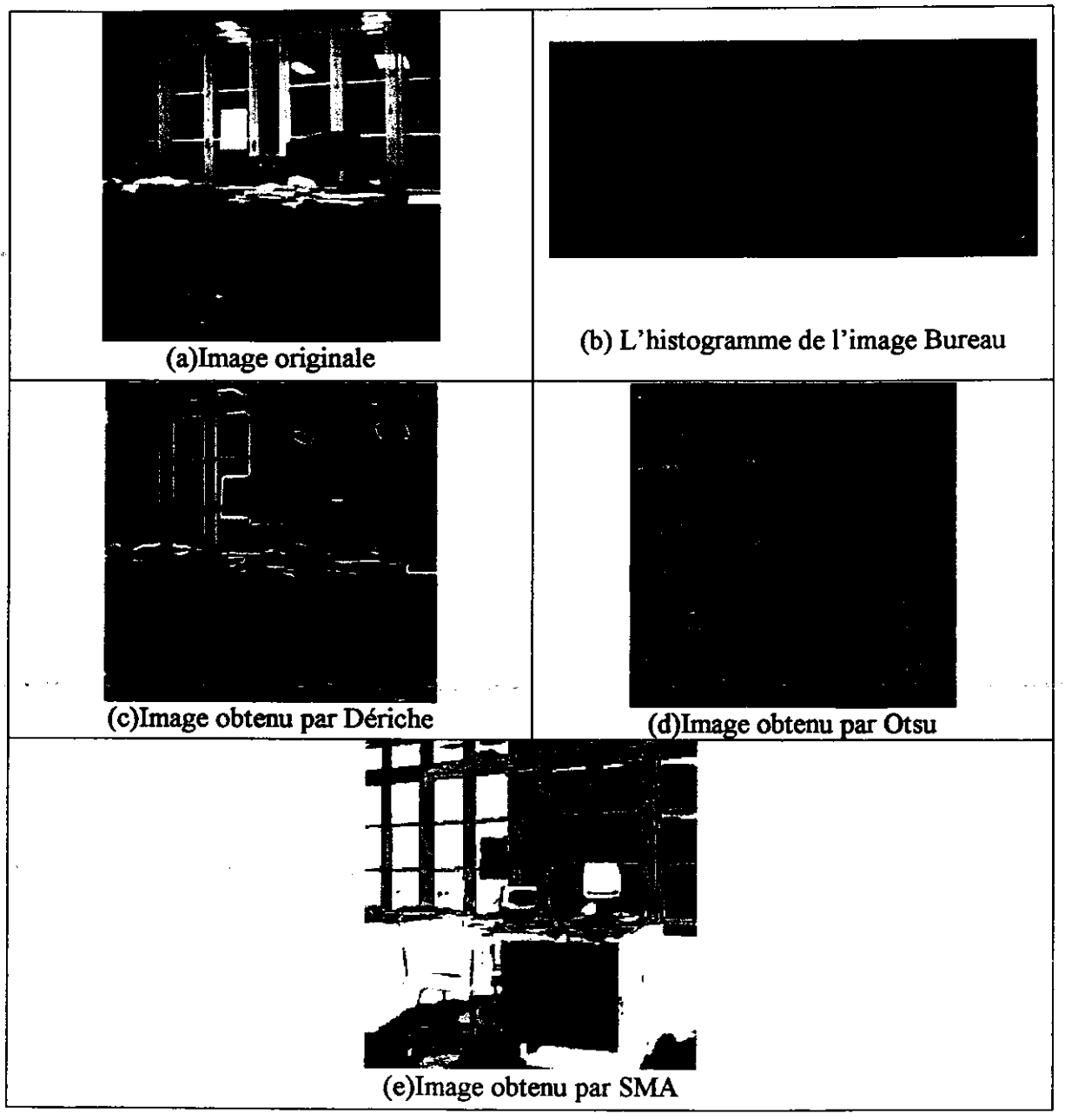


Figure V.15: Application des méthodes de la segmentation sur l'image Bureau

Application sur une image d'IRM cérébrale

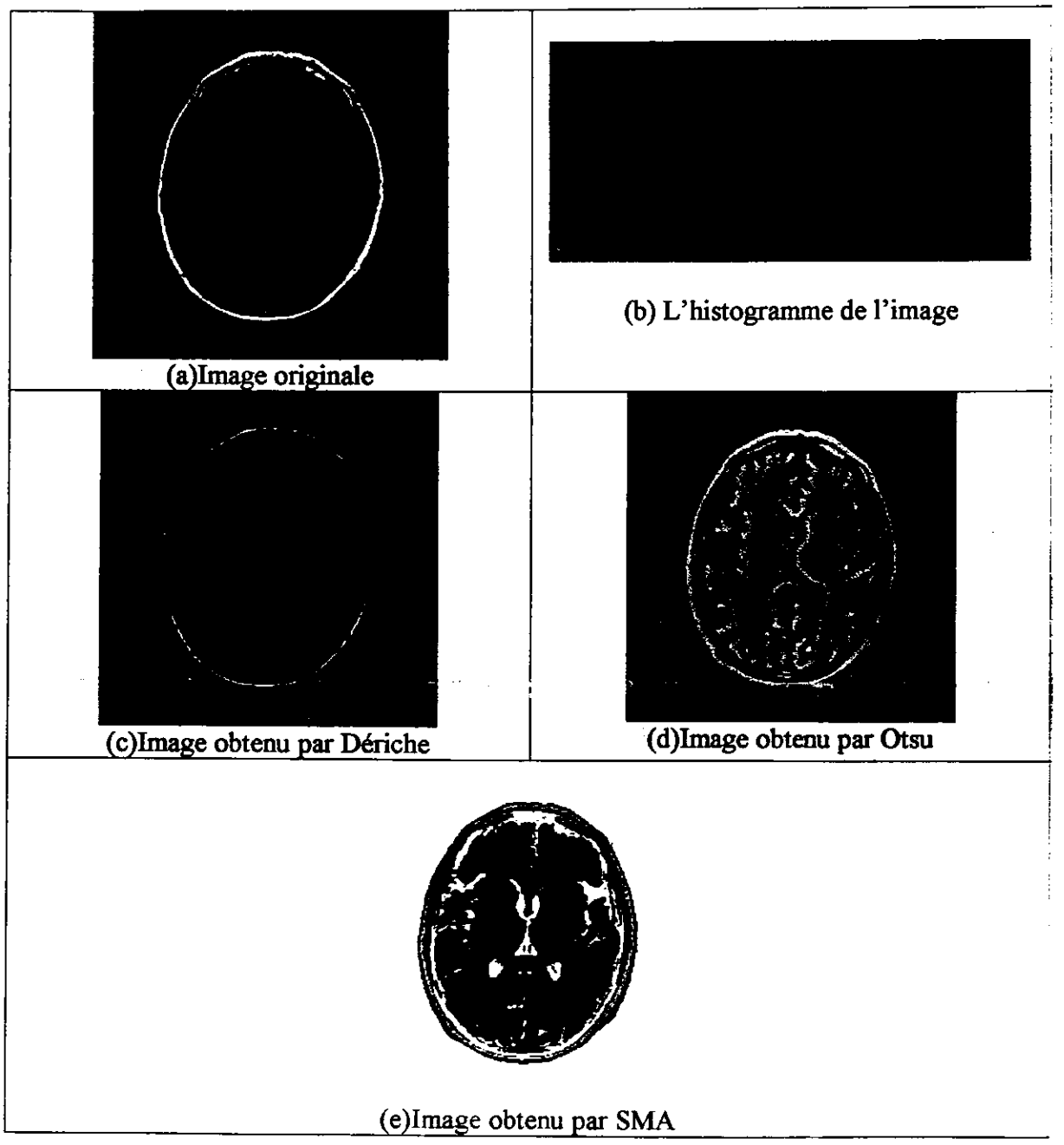


Figure V.16: Application des méthodes de la segmentation sur l'image IRM cérébrale

V.5. Estimation du temps de calcul

Apres avoir effectué plusieurs teste sur des images de taille différentes. On a constaté que le contenu de l'imagette influe sur le temps d'exécution qui varie entre 11 et 15 secondes. La figure ci-dessous illustre une image IRM cérébrale décomposée en imagettes de 16x16.

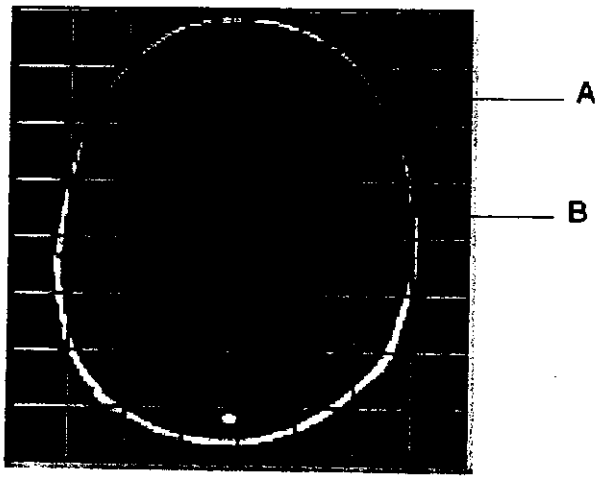


Figure V.17: Image IRM cérébrale 128x128 décomposée en imagettes

Le tableau ci-dessous montre les temps de segmentation lorsqu'on applique l'approche SMA sur des images de tailles différentes

Taille de l'image	Nombre d'imagette	Temps d'exécution	Temps d'exécution
16x16	1	11 sec	15 sec
32x32	4	44 sec	1 min
48x48	9	1.38 min	2.25 min
64x64	16	2.52 min	4 min
80x80	25	4.58 min	6.25 min
96x96	36	6.33 min	9 min
112x112	49	8.35 min	12.25 min
128x128	64	11.29 min	16 min
256x256	256	46.57min	1h ,4 min
512x512	1024	3h ,18 min	4h ,26 min

La fusion des pixels homogènes de l'imagette (A) (Figure V.17) va prendre plus de temps que l'imagette (B) parce que si on compare les deux images A et B on trouve que l'image B contient une seul classe par contre L'image A contient 3 classe ; le processus de fusion va alors progresser rapidement dans l'imagette (B) que dans (A).

VI. CONCLUSION

Dans ce chapitre nous avons présenté l'implémentation de notre système ainsi que les résultats obtenu par deux méthodes dual segmentation par extraction de contours et segmentation par extraction de régions. Et les améliorations apporter moyennant la segmentation de système multi agents

CONCLUSION GENERALE

Le travail présenté dans ce mémoire consiste à exploiter les systèmes multi-agents dans un processus de segmentation coopératif.

Ce travail s'est confiné dans un domaine très pointu de la vision artificielle. A l'intérieur de ce domaine, pourtant, un travail important a été effectué. L'intérêt de ce dernier repose sur trois éléments. Tout d'abord, il définit un cadre commun du travail pour permettre la mise en œuvre de coopération entre des approches hétérogènes de la segmentation. Ensuite, il explore un nouveau paradigme (SMA) composé de différents modules :

- module pour la segmentation des régions moyennant la méthode d'Otsu,
- module pour l'extraction des contours Moyennant le détecteur de Dériche,
- module pour la gestion et le contrôle de SMA baser sur les modèles locaux,
- module de la segmentation coopérative,
- module d'interface utilisateur.

Ces modules permettent de rendre la segmentation fortement distribuée, fiable, coopérative et bien guidée. Finalement, il propose une démarche qui coopère entre les différents types d'agents pour permettre de résoudre collectivement des problèmes complexes liés à la segmentation et qui offert une solution optimale base sur l'émergence d'un comportement global.

Le principal inconvénient de notre approche c'est le temps d'exécution.

Perspectives

Plusieurs amélioration peuvent être apportée à notre système, pour les quelles nous pouvons citer :

- Elaborer une plate forme multi agents, qui offre aux utilisateurs selon leurs besoins d'effectue leurs expérimentation en paramétrant le système.
- Implémenté d'autres types de coopérations tel que la coopération contour- contour pour le raccordement des contours et la coopération contour-région pour l'attachement des contours.
- Développer un système de segmentation complet pour la détection des contours.

Enfin, si nous sommes conscients que la contribution de ce travail est modeste au niveau du progrès de la vision artificielle, nous osons néanmoins prétendre qu'elle peut être utile aux personnes s'intéressant au traitement des images

Références Bibliographiques

[Amm, 95] : Z. AMMAR « Système de segmentation d'images à base de connaissances ». Thèse de Doctorat de l'Université de Caen (1995).

[Ber, 02] : F. Bernardi, « Méthode d'analyse orienté objet UML » , Dondo , 2002.

[Bou, 98] : A. BOUCHER, C. GARBAY « Des agents spécialisés pour la compréhension de séquences d'images ». RFIA'98, vol . II – p.275-284 (1998).

[Bou, 03] : Jérôme Bougeault, Livre « **JAVA La Maîtrise** Guide de formation avec exercices corrigés » , TSoft EDITEUR , Eyrolles 2003. http://www.editions-eyrolles.com

[Bra, 88] : Bratman.M, Israel.D, Pollack.M « Plans and resource bounded practicalresonning » Computatunal intelegence, 4 p.349-355 1998

[Bri, 01] : Jean-Piérre Briot, Yves Demazeau, Livre« Principes et architecture des systèmes multi-agent », Hermés Science PUBLICATION, Lavoisier 2001.

[Bur, 01] : A. BUREAU, C. GARBAY, M. DOJAT « Coopération entre deux populations d'agents pour la segmentation ». ORASIS'2001 – p.281-290 (2001).

[Can, 86] : Canny, J. (1986). A computational approach to edge detection. *IEEE Pattern Analysis and Machine Intelligence*, 8(6) :679–698.

[Cha, 05] : benoît CHARROUX, Aomar OSMANI, Yann THIERRY-MIEG, Livre « Synthèse de cours & exercices corrigés UML2 », collection Synthex PEARSON Education , 2005.

[Che, 98] : M Cheriet, J.N Said, C.Y.Suen, « A Recursive Thresholding Technique for Image Segmentation » , Article IEEE, 1998.

[Duc, 01] : E. DUCHESNAY « Agents situés dans l'image et organisés en pyramide irrégulière. Contribution à la segmentation par une approche d'agrégation coopérative et adaptative ». Thèse de Doctorat de l'Université de Rennes-1 (2001).

[Fer, 92] : Ferguson I.A, Touring machines: an architecture for dynamic, rational, mobile agents, PHD thesis, clare hall, university of combridge, Grande Bretagne 1992.

[Fer, 95] : Ferber. J livre les systèmes multi agent vers une intelligence collective, inter-Edition 1995.

[Jag, 98] : Jaggi, C. «Segmentation par méthode markovienne de l'encéphale humain par résonance magnétique: théorie, mise en oeuvre et évaluation». Thèse de doctorat, Université de Caen. (1998).

[Jen, 98] : Jennings N.R ,Wooldridge M ,Sycara K , A roadmap of agent reserch and development int. Journal of autonomous Agent and Multi Agent systems, vol.1, n°1 p. 7-38, 1998.

[Gal, 01] : DI GALLO Frédéric, Cours magistral du Cnam.doc « Méthodologie des système informatique UML.» Cours dispensé par Annick Lassus. 2001. http://pfdigallo.online.frcoursuml.pdf

[Gar, 01] : C. GARBAY, « Architectures logicielles et contrôle dans les systèmes de vision ». in « Les systèmes de vision » – J-M. Jolion eds. – Traité IC2 – Hermès- chap. 7 (2001).

[Gér, 98] : Géraud, T. « Segmentation des structures internes du cerveau en imagerie par résonance magnétique 3D». Thèse de doctorat, Telecom Paris. (1998).

[Ger, 06] : Pierre Gérard, Cours magistral, université de paris 13, 2006 « introduction à la modélisation orienter objet » http://www-lipn.univ-paris13.fr

[Gon, 92] : R.C. Gonzalez And R.E, Woods. Digital image processing. Addison-Wesley, Reading, MA.

[Ker, 95] : C. Kermad, K. Chehdi, et C. Cariou, segmentation d'images par multiseuillage et fusion de régions labellisées minimisant un critère de similarité. In Quinzième Colloque GRETSI. 2 :641-644, JUAN-LES-PINS, septembre, 1995.

[Liu, 99] : J. LIU, Y. T. TANG « Adaptive Image Segmentation With Distributed Behavior-Based Agents ». IEEE Trans. On PAMI 21(6) – p.544-551 (1999).

[Naz, 84] : A.M. NAZIF, M.D. LEVINE - "Low level image segmentation: an expert system". IEEE PAMI - vol. PAMI-6 n° 5 - p. 555-577 (1984).

[Oua, 06]: S. OUADFEL « Contributions à la Segmentation d'images basées sur la résolution collective par colonies de fourmis artificielles». Thèse de Doctorat de l'Université Hadj Lakhdar de Batna (2006).

[Pav, 92] T. Pavlidis, Why progress in machine vision is so slow, Pattern Recognition Letters, vol. 13, p. 221-225, 1992.

[Ren, 02] : René Mandiau, Emmanuelle Grislin-Le Strugeon, André Péninou, Livre « Organisation et applications des SMA » , Hermés Science PUBLICATION , Lavoisier 2002.

[Ric, 01] : N. RICHARD, M. DOJAT, C. GARBAY « Dynamic adaptation of cooperative agents for MRI brain scans segmentation ». Artificicial Intelligence in Medecine - AIME'01 - p. 349-358 (2001).

[Ric, 02] : N. RICHARD, M. DOJAT, C. GARBAY « Situated cooperative agents: a powerful paradigm for MRI brain scans segmentation ». European Conf. On AI – ECAI-2002.

[Zuc, 76] S. Zucker. Region growing : childhood and adolescence. Computer Graphics and Image Processing, 5 :382–399, 1976.

Références Webographies

[Ref, 01]: http:// uml.free.fr

[Ref, 02]: http://fr.wikipedia.org/wiki/Unified_Process

[Ref, 03]: Objecteering documentation set http://www.softeam.org/produits.htm.

[Ref, 04]: http://laurent-piechocki.developpez.com

[Ref, 05]: http://fr.wikipedia.org/wiki/Syst%C3%A8me_multi-agents

[Ref, 06]: http://bjaton.free.fr/perso/site_signal/deriche.php