UNIVERSITE SAAD DAHLAB DE BLIDA

Faculté des Sciences de l'Ingénieur

Département d'Electronique

MEMOIRE DE MAGISTER

Spécialité : Images et Parole

GESTION ET CONTROLE DES CONNAISSANCES DANS UN SYSTEME MULTI-AGENTS DEDIE A L'ANALYSE D'IMAGES

Par

Salim MALEK

Devant le jury composé de :

H.SALHI	Maître de Conférence, U. de Blida	Président
M.GUERTI	Maître de Conférence, ENP, Alger	Examinateur
M.AIDJA	Chargé de cours, U de Blida	Examinateur
A.GUESSOUM	Professeur, U. de Blida	Rapporteur
Y.KABIR	Maitre assistant, U de Blida	Co-Rapporteur

Blida, février 2007

RESUME

Dans ce mémoire, nous avons présenté une étude bibliographique des techniques d'analyse d'images, en se basant sur celles de segmentation d'images avec les différentes approches (contour et région). D'après cette étude, nous avons remarqué qu'aucune approche (contour et région) ne peut toute seule, garantir la résolution du problème de la segmentation. La solution serait de combiner différentes approches de segmentation d'images et de prétraitements, en exploitant les avantages de certaines méthodes pour pallier les inconvénients des autres. Ces techniques appelées « méthodes coopératives » constituent la tendance actuelle dans la segmentation d'images.

La complexité de la gestion et le contrôle des connaissances dans les algorithmes classiques (séquentiels) nous a amené à utiliser une approche très connue dans le domaine de l'intelligence artificielle distribuée (IAD) c'est les systèmes multi-agents (SMA). En utilisant une plate-forme de parallélisation (PVM : Parallel Virtual Machine), la gestion des informations sera très souple et la coopération sera très fructueuse ; chaque agent sera chargé d'une tâche bien définie et coopéra avec les autres agents pour le compte de l'application globale.

Nous avons appliqué cette approche pour l'analyse d'images et plus particulièrement pour la segmentation coopérative.

Mots clés : analyse d'images, segmentation, détection de contours, extraction de régions, coopération, système multi-agents (SMA), intelligence artificielle distribuée (IAD), contrôle, PVM (Parallel Virtual Machine).

ملخص

في هذه المذكرة، نقدم دراسة حول التقنيات المستعملة في مجال تحليل الصور، مع التركيز على تلك المتعلقة بتجزئة الصور بمختلف مقارباتها (الحواف و المناطق). و من منطلق هذه الدراسة، لاحظنا أنه ليس بإمكان أي مقاربة لوحدها (الحواف و المناطق) أن تحل مشكلة التجزئة. هناك حل يكمن في مشاركة مقاربات و معالجات مختلفة و ذلك باستغلال إيجابيات بعض الطرق لتغطية عجز البعض الآخر منها. يسمى هذا النوع من الطرق ب " الطرق التشاركية " و التي تشكل في الوقت الحالي التوجه المعمول به.

تعقيدات تسيير و مراقبة المعارف في الخوارزميات الكلاسيكية (التعاقبية) جعلتنا نتجه إلى مقاربة معروفة في مجال الذكاء الإصطناعي الموزع (ذ.إ.م) ألا وهي : الأنظمة المتعددة العملاء (أ.م.ع). باستعمال أرضية للبرمجة المتوازية (الآلة المتوازية الإفتراضية : آ.م.إ)، سيصبح تسيير المعلومات أكثر ليونة و التشارك أكثر مردودية ؛ حيث سيكلف كل عميل بمهمة محددة و يشارك مع زملائه (العملاء الآخرون) من أجل المصلحة العامة.

لقد قمنا بتطبيق هذه المقاربة في مجال معالجة الصور و بالخصوص في طرق تجزئة الصور التشاركية.

كلمات مفاتيح : تحليل الصور، تجزئة الصور، التقاط الحواف، استخراج المناطق، المشاركة، الذكاء الإصطناعي الموزع (ذ.إ.م)، الأنظمة المتعددة العملاء (أ.م.ع)، الآلة المتوازية الإفتراضية (آ.م.إ)، المراقبة.

ABSTRACT

In this work we presented a bibliographic study of images analysis techniques, while taking those of images segmentation with the different approaches as a basis (edge and region). According to this survey we noticed that approaches (edge and region) is not able to all alone, to guarantee the resolution of segmentation problem. The solution can be obtained by combining various segmentation methods and some other images processing techniques in order to exploit certain method advantages to solve inconveniences of the other. These techniques called " cooperative methods " constitute the present tendency in the images segmentation.

The complexity of the management and the control of knowledge in the classic algorithms (sequential) brought us to use a quite-known approach in the domain of the Distributed Artificial Intelligence (DAI) it is Multi-Agents Systems (MAS). While using parallel platform (PVM : Parallel Virtual Machine), the management of information will be very flexible and cooperation will be very fruitful ; each agent will be in charge of a very definite task and cooperate with the other agents for the account of the global application.

We applied this approch for the images analysis particularly in the cooperative segmentation.

Keywords : images analysis, images segmentation, edges detection, regions extraction, cooperation, multi-agents systems (MAS), Distributed Artificial Intelligence (DAI), control, PVM (Parallel Virtual Machine).

REMERCIEMENTS

Toute ma gratitude et remerciements au bon Dieu, qui m'a dirigé dans la voie de la science, et m'a donné le courage et la volonté pour arriver à ce niveau.

Je tiens à exprimer toute ma reconnaissance à mon promoteur M^{er} A. GESSOUM qui m'a dirigé et encouragé tout au long de ce travail. Je lui adresse mes vifs remerciements.

Je tiens à remercier mon co-promoteur M^{er} Y. KEBIR pour ses conseils et son aide qui m'ont beaucoup éclairé la voie de la recherche.

Les remerciements iront de même à tous les professeurs qui ont contribué à ma formation, en particulier, ceux du département d'électronique (USTB)

Je remercie aussi M^{er} ABDELKADER de la bibliothèque de l'institut d'électronique Université de Blida qui a toujours su me trouver les ouvrages nécessaires,

Je remercie vivement M^{er} SALHI d'avoir bien voulu me faire l'honneur d'accepter la présidence du jury de ce mémoire.

Je tiens à remercier infiniment M^{me} GUERTI et M^{er} AIDJA pour avoir accepté d'examiner ce travail, et de faire partie du jury.

Je remercie toutes personnes qui, de prés ou de loin, ont participé à la mise en forme de ce travail.

Enfin, tiens à remercier, du profond de mon cœur, mes parents pour leur soutien permanent et leurs encouragements incessants. Je remercie aussi touts les membres de ma famille.

TABLE DES MATIÈRES

LISTE DES ILLUSTRATIONS, GRAPHIQUES ET TABLEAUX

INTRODUCTION

Un système de vision a pour but de reconnaître une scène qui lui est présentée et d'en décrire les différents éléments. Il constitue, de façon générale, un système de traitement de l'information avec en entrée une image comme information et en sortie, une description symbolique de cet image. Entre les deux extrémités se situe tout un ensemble de traitements de l'image qui doivent être effectués afin de réaliser la transformation des informations demandées.

Le traitement d'images est l'ensemble des méthodes et techniques opérant sur des images, dans le but d'améliorer leur aspect visuel et d'en extraire des informations jugées pertinentes, il constitue avec la reconnaissance des formes un système complet de vision par ordinateur (VPO). Cependant, Le traitement d'images est un domaine très complexe, cette complexité est due à plusieurs facteurs qui sont répartis dans les trois (03) axes de recherche suivants :

- La grande quantité d'informations traitée dans une image : par exemple, une image en niveau de gris de taille 256*256 contient 65536 pixels et une image en couleurs de taille 512*512 comporte 786432 pixels, ce qui mène les chercheurs à réduire la complexité des algorithme proposés, donc réduire la qualité des résultats, pour rester dans des temps de calcul raisonnables. Alors que le développement actuel des réseaux informatiques et de calculateurs de programmation parallèle représente un avantage pour progresser plus dans la complexité des algorithmes et améliorer donc les résultats.
- L'hétérogénéité : plusieurs méthodes en traitement d'images sont de nature différentes bien qu'elles sont complémentaires, un exemple bien concret est celui de la segmentation par extraction des contours et de la segmentation par extraction de régions, il est évident que ces deux approches sont duales dans le sens que la frontière d'une région définit un contour et un contour fermé définit une région, mais les résultats obtenus sont différents. Les contours possèdent l'essentiel des caractéristiques de forme des régions, alors que la segmentation en région privilégie les caractéristiques non géométriques, liées

aux critères d'homogénéité, par conséquent au contenu de la région plus que sa forme. Donc faire une coopération entre les différentes méthodes, en particulier dans la phase de la segmentation, en exploitant les avantages de certaines méthodes pour pallier les inconvénients des autres peut explicitement améliorer les résultats obtenus auparavant.

- Le contrôle heuristique : tel que le choix d'un opérateur de filtrage, le seuil de binarisation… ; Ces derniers ne peuvent être définis par aucun formalisme mathématique, ils sont généralement sélectionnés, parmi d'autres, après plusieurs testes, alors que la tendance actuelle est d'automatiser les procédures afin de minimiser l'intervention humaine.

La résolution de ces problèmes complexes d'une manière distribuée est possible grâce à des entités distribuées. Ces entités collaborent entre eux et coordonnent leurs actions pour arriver à accomplir leurs objectifs aussi bien individuels que collectifs. Les systèmes multi-agents (SMA) ont été introduits dans ce contexte comme un outil de l'intelligence artificiel distribué (IAD) pour pallier les problèmes posés.

L'intelligence Artificielle Distribuée consiste à distribuer l'expertise au sein d'une société d'entités appelées agents dont le contrôle et les données sont distribués. Chaque agent a ses propres compétences mais il a besoin d'interagir avec les autres pour résoudre des problèmes qui dépendent de son domaine d'expertise. L'objectif de l'IAD est donc de trouver une solution à des problèmes globaux grâce à un ensemble d'entités distribuées travaillant en collaboration.

L'objectif de cette thèse est d'aborder le problème de la détection des indices visuels simples dans une image, à savoir les contours et les régions, sous l'angle de la gestion des informations et leurs contrôle dans un système multi-agents dédié à l'analyse d'images. Par gestion des informations, nous entendons la manière dont est décomposé le problème, comment s'enchaînent les étapes, comment se fait le flux des informations, quelles sont les relations entre les différents types d'informations et quelle est la structure de contrôle de l'algorithme. Il ne s'agit pas ici uniquement de génie logiciel. Les problèmes de gestion des informations nous amènent à étudier les problèmes de conception, de réalisation et d'évaluation liés à notre application. Nous allons, donc concevoir et implanter un système multi-agents dans le domaine de traitement d'images. La motivation d'une telle implantation est due à la possibilité de

moduler et paralléliser les traitements des différentes étapes de l'analyse d'images et de mieux gérer et contrôler les connaissances échangées dans le système.

Notre travail consiste à améliorer un système qui a été fait par Boukaci.A et Chelbeb.A [46] et dirigé par Mer Kabir.Y, bien que ce système est important, il présente un nombre d'inconvénient que nous essayerons de pallier, parmi lesquels :

- le temps de calcul très élevé.
- L'absence des règles de décision dans plusieurs étapes du système, ce qui laisse toujours le système en dépendance avec l'utilisateur.
- L'absence d'une stratégie de contrôle globale du système.
- Le déroulement du système (des agents) est invisible pour l'utilisateur, ce dernier ne peut voir qu'une fenêtre DOS (l'agent Maitre) qui indique seulement le début et la fin du système.
- Ce système est implanté en mode console (sous DOS), il ne permet d'ouvrir que des images brutes de format « raw » ou « ima ».

Les modifications à faire dans notre travail sont citées dans les points suivants :

- Nous donnerons la priorité aux agents qui nécessitent beaucoup de temps dans leurs traitements, de plus nous allons gérer convenablement l'allocation mémoire et les échanges des connaissances entre les agents.
- Nous implanterons des règles de décision dans notre système, ces règles donneront au système la possibilité de prendre des décisions sans l'intervention de l'utilisateur.
- Nous allons implanter une stratégie globale de contrôle, cette stratégie repose sur le contrôle individuel où chaque agent se charge de contrôler ses actions et sur le contrôle social où nous allons répartir la tâche du contrôle du système sur un ensemble d'agents.
- Nous allons réaliser une interface graphique qui nous permettra de traiter plusieurs formats d'images, cet interface nous permettra aussi de superviser le déroulement du système et de nous donner l'état des agents.

Sur le plan formel, cette thèse est composé de deux parties : l'une théorique et l'autre pratique.

- La première partie est basée sur les quatre premiers chapitres qui sont :

  - Dans le premier chapitre, intitulé « Les problème de gestion des informations », nous proposons une étude théorique ou générale des problèmes de gestion des informations, en illustrant nos propos par des exemples. Nous discutons brièvement du fonctionnement de la vision humaine en essayant de situer l'étape de détection des premiers indices visuels ("indices visuels primaires"), qui nous intéresse plus particulièrement. Nous effectuons ensuite une analyse des problèmes liés à la prise de décision, à l'émergence des informations et au flux de celles-ci. Nous discutons également du problème de conception des algorithmes, où nous opposons les approches qui tentent de modéliser les indices visuels aux approches heuristiques qui privilégient l'aspect expérimental.

  - Le deuxième chapitre, intitulé « Analyse d'images », représente une vue détaillée sur le traitement de l'image où nous définissons les méthodes les plus utilisées, commençons par le prétraitement où nous allons détailler les trois grandes parties, à savoir : le filtrage, les méthodes basées sur la modification de l'histogramme et le rehaussement de contraste, ensuite nous proposant une étude de l'étape de la segmentation qui comporte la segmentation par extraction des contours et la segmentation en région. Enfin, nous nous focalisons sur la segmentation coopérative (région-contour et contour-région. La coopération comporte implicitement des problèmes de gestion des informations, ce qui va nous permettre d'illustrer les grands principes présentés au premier chapitre.

  - Le troisième chapitre, intitulé « Concept de base sur les systèmes multi-agents ». Dans ce chapitre, nous présentons les fondements de l'intelligence artificielle distribuée et son rapport à l'intelligence artificielle « classique ». Nous développerons aussi les caractéristiques spécifiques aux systèmes multi-agents.

  - Le quatrième chapitre, intitulé «Parallélisme et l'environnement PVM». Dans ce chapitre, nous présentons l'environnement PVM (Parallel Virtual

Machine) qui permet à un réseau hétérogène d'ordinateurs d'être utilisé comme une seul machine parallèle virtuelle avec laquelle le système multi-agents proposé est conçu.

- La deuxième partie de notre étude est consacrée à la phase pratique et s'identifie au dernier chapitre qui est intitulé « Implémentation et résultats », dans lequel nous présentons les résultats obtenus dans notre travail. A la fin on termine par une conclusion générale et des perspectives.

CHAPITRE 1
LES PROBLEMES DE GESTION DES INFORMATIONS

1.1. Introduction

Le but des premières étapes de la vision par ordinateur est de décrire l'image à l'aide d'indices visuels simples. Ces indices doivent être pertinents afin de faciliter les étapes suivantes dans une perspective de reconnaissance rapide et robuste des objets de la scène. Cette description consiste souvent en une détection des discontinuités ou une segmentation de l'image en régions.

Dans la première partie de ce chapitre, nous essayons de définir la notion « d'indice visuel primaire » en étudiant le mécanisme global de la vision. D'abord, nous nous penchons brièvement sur la vision humaine, puis nous essayons de dégager les grandes lignes d'un système de vision artificiel. Ceci nous conduit à cerner les difficultés majeures de la détection d'indices visuels.

Dans la deuxième partie de ce chapitre, nous abordons les problèmes liés à la gestion des informations dans les algorithmes de détection d'indices visuels. Ces problèmes sont difficiles à aborder car il n'est pas possible d'en faire une modélisation simple. Nous nous attachons plus particulièrement à identifier et comprendre les difficultés qui précèdent la décision qui détermine la présence d'un indice visuel. Nous présentons ainsi des idées concernant l'abondance et la complémentarité des informations, puis nous abordons le problème délicat de leur qualité objective qui conditionne l'émergence des indices visuels et nous terminons par la notion intuitive de « liberté de l'information ».

Dans la dernière partie de ce chapitre, nous examinons les problèmes liés à la conception d'un détecteur d'indices visuels en présentant deux grandes catégories d'approches, l'une consistant à modéliser l'indice et l'autre cherchant à approcher la solution par des heuristiques.

1.2. La détection d'indices visuels

Nous définissons un indice visuel comme étant une information descriptive de l'image, ne faisant intervenir aucune connaissance sur les objets. Nous essayons ici de situer l'étape de détection des indices visuels dans un système de vision et de définir les « indices visuels primaires ».

1.2.1. La vision humaine

La qualité de la vision humaine est exceptionnelle. Elle s'avère être malheureusement une mécanique extrêmement complexe dont nous n'avons q'une idée générale du fonctionnement. Des expériences ont été réalisées et des théories ont été proposées mais elles ne donnent qu'une vue d'ensemble de la vision naturelle.

Notre but est ici de rappeler quelques éléments connus de la vision naturelle et de situer l'étape de détection d'indices visuels. Cette étude reste d'un niveau très superficiel car d'une part seuls les principes généraux nous intéressent et d'autre part la neurobiologie n'est pas de notre compétence.

Des testes sur la rapidité de la reconnaissance visuelle ont montré que celle-ci est très rapide. En tenant compte de la vitesse relativement lente de transmission du potentiel au niveau de la synapse qui lie deux neurones, les chercheurs ont émis l'hypothèse que le traitement des informations visuelles est direct, des récepteurs situés sur la rétine jusqu'aux neurones exprimant la reconnaissance d'un objet [15]. En d'autres termes, il n'y a pas de temps d'attente ou de cycle de retour pour une remise en cause de l'information ou une vision à deux vitesses. Tout se passe comme si la vision était un enchaînement d'étapes simples sans remise en cause des résultats intermédiaires.

L'étude du cerveau montre qu'il existe des régions où les neurones sont différemment organisés. Les neurones sont souvent organisés en couche avec des spécificités propres, tant au niveau des connexions avec les autres couches qu'au niveau des propriétés internes comme le seuil d'excitabilité. Cette particularité permet de supposer qu'il existe des groupes de neurones dédiés à une tâche particulière. La vision pourrait ainsi être décomposée en plusieurs étapes distinctes. Des0 expériences ont par exemple montré que certaines zones sont sensibles à la couleur alors que d'autres sont sensibles à la forme, ce qui confirme cette hypothèse [15].

Un des phénomènes liés étroitement à la vision est notre capacité d'apprentissage. Par exemple, quelques secondes d'attention suffisent pour que nous mémorisions un nouveau visage et que nous soyons capables de le reconnaître quelques instants plus tard. Nous sommes donc capables d'apprendre dans un temps très court les signes caractéristiques d'un visage. Des expériences ont montré qu'après apprentissage, certains poids synaptiques (la force des liaisons entre les neurones) changent. Lorsque le même événement se produit, des neurones qui n'étaient pas excités avant apprentissage le sont ensuite et caractérisent celui-ci. Cela suppose que le changement des poids synaptiques est un des mécanismes essentiels à l'apprentissage et à la mémorisation [15].

Nous donnons maintenant une idée de ce que pourrait être le fonctionnement de la vision naturelle. La première grande partie de la vision est la détection d'indices visuels, ces indices seraient des informations sur la présence et la forme des contours, des informations sur la couleur, l'homogénéité, la texture et d'autres. Si un indice visuel primaire est présent, nous pouvons supposer que des neurones spécifiques sont excités. Ensuite, à l'étape suivante, d'autres neurones spécifiques sont excités en fonction de la présence simultanée de plusieurs indices visuels primaires. Le schéma de la figure 1.1 est bien sur très simplifié car nous pouvons imaginer qu'il existe ainsi plusieurs niveaux de complexité croissante correspondant à la présence d'indices visuels de plus en plus sophistiqués. Il devrait y avoir de plus des zones traitant un problème spécifique [15].

A ce stade, il n'y a pas encore reconnaissance, mais uniquement génération des hypothèses de ce qui est vu. Les contraintes globales doivent pouvoir inhiber les neurones correspondant à des hypothèses non vérifiées, ce qui permettrait de faire émerger finalement une description de la scène avec des indices visuels de très haut niveau. Ensuite, deux cas se présentent. Si les neurones excités à ce dernier niveau associent leurs réponses pour exciter un groupe de neurones qui identifient un objet, la reconnaissance est validée. Sinon, il est probable que c'est la première fois que les indices visuels présents ont été découvert simultanément [15]. Ils excitent un groupe de neurones et par modification des poids synaptiques, ceux-ci correspondront à l'identification de l'objet inconnu.

Lorsque l'objet est de nouveau présent dans la scène, les indices visuels sont les mêmes. Grâce à l'apprentissage, le groupe de neurones identifiant l'objet va être excité ce qui caractérise alors la reconnaissance.

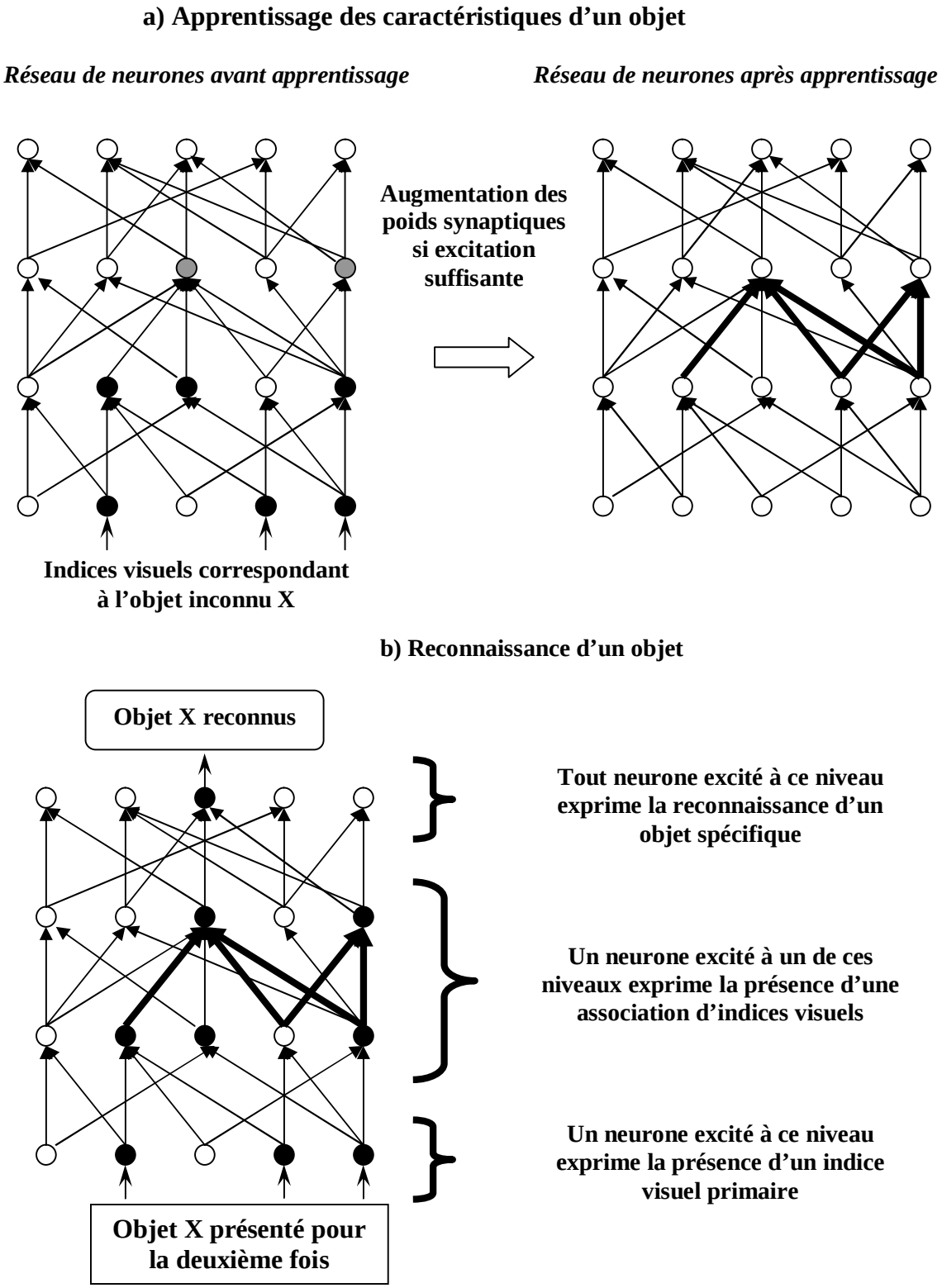


Figure 1-1 : Schéma l'apprentissage et de la reconnaissance d'objets.

1.2.2. La vision artificielle

1.2.2.1. Organisation d'un système de vision

De nombreuses tentatives ont été faites pour réaliser un système complet de vision. Nous pouvons classer les approches en deux catégories, celles qui tentent de copier la vision humaine (approche neuromimétique) et celles qui considèrent la vision indépendamment de l'exemple humain ou animal.

- En ce qui concerne les approches neuromimétiques, nous pouvons citer les travaux de Grossberg et Mingolla dont le but est d'exploiter des réseaux de neurones informatiques pour simuler les différentes étapes de la vision humaine, ainsi que les travaux de Gaussier et Cocquerez pour la reconnaissance de scènes selon des principes similaires avec simulation des "saccades oculaires". Les approches fondées sur les réseaux de neurones ont montré de grandes qualités pour diverses applications et la recherche est actuellement très active dans ce domaine.
- En ce qui concerne les systèmes de vision qui n'ont pas été inspirés de la vision humaine, un des plus connus est celui de Hanson et Riseman, VISIONS. Ce système est capable de reconnaître des routes, des arbres et des maisons dans un contexte relativement limité. Une des caractéristiques de ce système est l'exploitation du "tableau noir" qui permet de relier tous les processus entre eux et d'échanger ainsi des informations de différente nature. Le système ACRONYM est également un projet ambitieux qui a donné lieu à une application en imagerie aérienne, de même que le système MESSIE. Chacun de ces systèmes a été implanté avec des algorithmes *ad hoc* développés pour la circonstance [15].

  Citons également les travaux de Clement et Thonnat avec le système expert OCAPI [6], ainsi que l'approche de Baujard et Garbay [13] qui ont exploité une architecture "multi-experte" (le système MAPS) pour construire un système de vision.

Il existe enfin des approches mixtes combinant par exemple réseaux de neurones et systèmes experts, où le neuromimétisme est centré sur les premières étapes du système de vision.

Que ce soit avec les approches neuromimétiques ou les autres, il existe en fait deux manières différentes de considérer l'étape de détection des indices visuels qui conduisent à deux grandes catégories de système de vision (voir figure 1.2). La première implique une détection d'indices visuels de plus en plus riches sans l'aide d'information sémantique, c'est l'approche guidée par les données, et la deuxième exploite au mieux le contexte pour aider la détection de ces indices, c'est l'approche dirigée par le but.

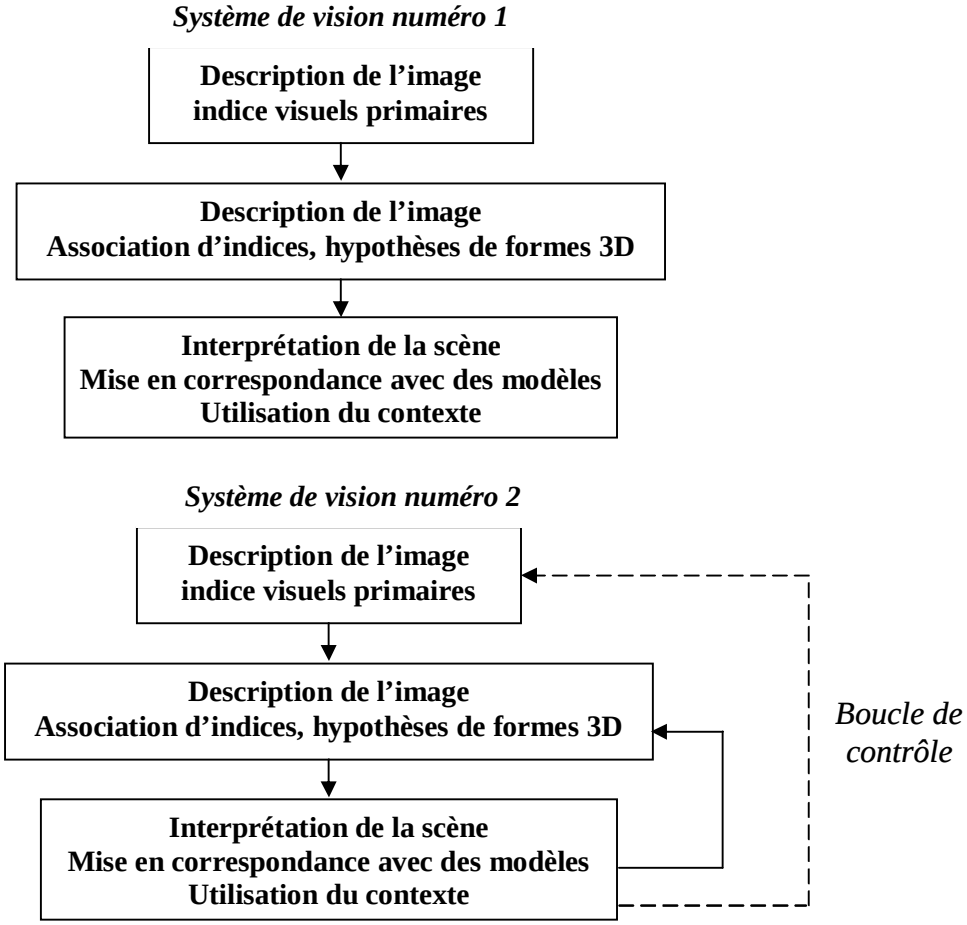


Figure 1-2 : Deux approches pour décomposer le problème de la vision.

1.2.2.2. Les indices visuels

Après avoir étudié rapidement les systèmes de vision, il est nécessaire de savoir ce que nous entendons exactement par indices visuels et indices visuels primaires. De façon intuitive, un indice visuel est une information "visuelle", qui apparaît donc directement à partir de la visualisation de l'image. La visibilité d'un indice est en fait un des problèmes clés qu'il nous faudra résoudre. En ce qui concerne la distinction d'indices visuels primaires ou complexes, nous proposons la décomposition suivante :

Les indices visuels primaires sont des informations sur l'image obtenues directement à partir de calculs sur les pixels et formant les premières structures de données importantes de niveau juste supérieur à celui des pixels. Ce sont par exemple les contours qui correspondent à des ruptures de propriété locales et les régions qui sont au contraire des zones où une propriété locale ou globale est conservée. Les indices visuels étant descriptifs, ces propriétés doivent être relativement simples et faciles à exploiter [15].

Une zone de texture uniforme (texture simple avec variations locales irrégulières) fait donc partie de ces indices visuels primaires, mais une zone de texture complexe

comprenant des éléments répétitifs de structure est du niveau supérieur. Généralement, tout groupe de pixels présentant une propriété intéressante simple constitue un indice visuel primaire qui mérite d'être détecté [15].

Nous proposons intuitivement contours et régions mais notre imagination est limitée et il doit exister un nombre considérable d'indices visuels primaires intéressants.

Tous les autres indices visuels sont le résultat d'une modélisation et d'une association d'indices plus simples, ils ne sont donc plus "primaires". Par exemple, il est possible de modéliser les contours par des courbes paramétriques ou des segments et jonctions, puis de les associer en fonction de leurs propriétés comme la proximité, la forme, la similarité du voisinage, la continuité etc…

De même, les régions peuvent être regroupées en fonction de leurs formes ou de leurs caractéristiques de niveaux de gris. Il peut y avoir également des associations plus complexes entre régions et contours (pour détecter par exemple des textures complexes).

Pour conclure cette partie, nous retenons que les indices visuels primaires s'obtiennent directement à partir de calculs sur les niveaux de gris et la position des pixels. Leur détection pose de nombreux problèmes au niveau de la gestion des informations, problèmes que nous abordons dans la suite de ce chapitre.

1.3. La gestion des informations

La détection des indices visuels primaires s'avère être d'une grande complexité.

Nous essayons dans cette partie de développer les grandes difficultés liées à la gestion des informations. Nous discutons d'abord de l'abondance des informations, tant au niveau de la quantité de données à traiter qu'au niveau du nombre de paramètres intervenant dans la justification de la présence d'un indice visuel. Nous évoquons ensuite les problèmes délicats de fusion des informations et d'exploitation de données d'origines diverses. Nous mettons également l'accent sur les problèmes liés à l'émergence des indices visuels et des informations en général, et nous introduisons la notion de "liberté de l'information" où nous abordons brièvement les problèmes de contrôle.

1.3.1. L'abondance des informations

Les premières étapes de la vision sont confrontées au problème de l'abondance des informations présentes dans une image. Les pixels représentant un ensemble de données de taille non négligeable, outre les problèmes d'espace mémoire, les temps de calculs deviennent

vite prohibitifs. Ainsi, pratiquement tous les algorithmes proposés sont de complexité linéaire [15]. Cela était encore plus vrai il y a vingt ans ou plus, car les ordinateurs étaient beaucoup moins rapides qu'aujourd'hui. En détection de contours, par exemple, des algorithmes très simples ont été proposés avec un calcul de gradient à l'aide de masques 2x2 et un seuillage simple ensuite. Actuellement, grâce au gain de puissance des ordinateurs et le développement des machines parallèles, la tendance est à la sophistication des algorithmes, mais tout en restant essentiellement dans le domaine de la complexité linéaire.

Le problème de l'abondance des informations est toutefois moins bien perçu au niveau de la prise de décision. Par exemple, lorsqu'il faut décider de la fusion de deux régions ou de l'existence d'un contour, il existe un grand nombre d'informations théoriquement indispensables pour que la décision soit correcte. Nous allons détailler ces deux exemples pour mieux comprendre l'importance du problème.

• ***PROBLEME DE FUSION DE DEUX REGIONS***

Lors de la fusion de deux régions, les informations utilisées sont généralement l'écart-type et la moyenne du niveau de gris des deux régions. Ce nombre d'informations est relativement conséquent et les critères amenant la meilleure décision sont difficiles à déterminer. Toutefois, le nombre d'informations prises en compte est-il suffisant ? Ne faudrait-il pas utiliser également des informations sur la forme des deux régions pour décider de leur fusion (par exemple si la région résultant de la fusion est de forme circulaire, les autres critères doivent être relâchés) ? Ne faudrait-il pas aussi prendre en compte la norme du gradient à la frontière et le nombre de pixels qui composent cette frontière ? Il est bien évident qu'il n'est pas possible de tout prendre en compte, mais la sélection de ces informations conditionne sans aucun doute la qualité des résultats.

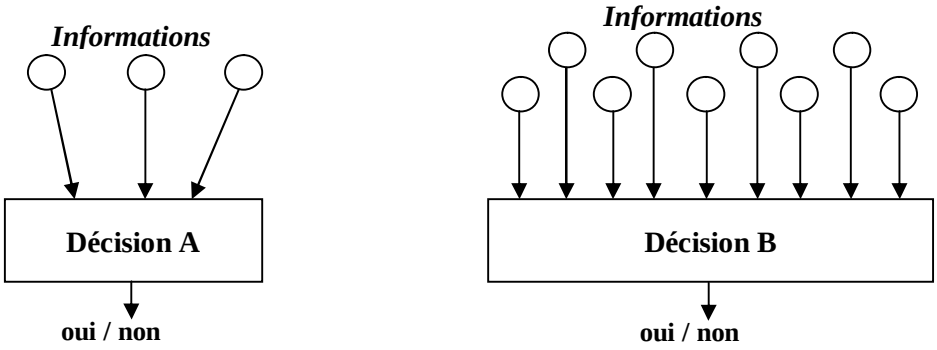


Figure 1-3 : Exemple du problème de l'abondance des informations.

Nous voyons ici que l'abondance des informations qui jouent théoriquement un rôle dans la prise de décision est un des problèmes clés qu'il ne faut surtout pas négliger. En considérant que toutes les informations ont à peu près la même importance, un algorithme B a ainsi un meilleur potentiel qu'un algorithme A s'il prend en compte plus d'informations pour prendre la décision fondamentale inhérente au problème (voir figure 1.3).

• ***VALIDATION D'UN CONTOUR***

Lors de la prise de décision qui détermine la présence d'un contour, les informations prises en compte sont souvent peu nombreuses. Dans la plupart des cas, le problème est ramené au calcul d'une fonction des valeurs de gris dont la réponse est maximale en présence d'un modèle de discontinuité. Cette fonction est souvent un calcul de gradient avec lissage dans une fenêtre d'observation réduite. Il est difficile d'évaluer le nombre d'informations prises en compte pour décider de la présence d'une discontinuité.

Toutefois, il est intéressant de noter que pour éviter la détection de discontinuités dans certaines textures, il parait logique d'avoir des informations provenant des régions voisines. Ces informations ne sont pourtant pas disponibles dans la grande majorité des détecteurs actuels. De manière générale, la détection et la localisation d'un contour dépendent largement de la présence des discontinuités et des propriétés des régions du voisinage. Or, la plupart des détecteurs effectuent une opération de seuillage global sur le gradient sans tenir compte de ces informations. Il est vrai que la détection des contours est souvent considérée comme une détection de toutes les variations présentes dans l'image et que le problème se ramène alors à trouver le meilleur opérateur de différentiation, après régularisation. Cependant, d'un point de vue gestion des informations, cette définition du contour est critiquable car elle ne correspond pas vraiment au besoin des étapes suivantes de la vision. Cela est d'autant plus vrai que des opérations telles qu'un seuillage par hystérésis, un prolongement des contours interrompus ou une suppression des petites chaînes de pixel-contours peuvent être interprétées comme une remise en cause du modèle choisi et donc de la définition même du contour en tant que forte variation.

La recherche de toutes les informations théoriquement nécessaires pour prendre la meilleure décision est donc un problème majeur qui mérite une grande attention lors de la conception d'un algorithme. Les décisions étant complexes, il s'agit d'opérer la meilleure sélection des informations qui vont être prises en compte, en évitant de trop simplifier et

tout en restant bien sûr dans des temps de calcul raisonnables. Ceci nous amène au premier principe pour la conception d'algorithmes de détection d'indices visuels.

PRINCIPE N°1 : La recherche de toutes les informations théoriquement nécessaires pour décider de la présence d'un indice visuel primaire est un problème majeur qui mérite une grande attention lors de la conception d'un algorithme [15].

1.3.2. La complémentarité des informations

1.3.2.1. Les bases du problème

Un des problèmes les plus difficiles lié à la décision est la prise en compte d'informations d'origines diverses. Ainsi, lors de la décision d'agrégation dans un processus de croissance de régions, il faut tenir compte du gradient, de l'écart-type de la région, de la forme qui va être générée etc…. Outre les problèmes inhérents à la différence de nature de ces informations, le problème fondamental qui est sous-jacent est la coopération entre les méthodes qui fournissent ces informations. Nous illustrons nos propos par deux exemples :

- Il n'est pas très intéressant de disposer de l'information du gradient sur un pixel donné ainsi que de l'écart-type d'une région 10x10 centré sur ce même pixel pour valider la présence d'une discontinuité. Ces informations sont de nature différente mais elles ne sont pas vraiment complémentaires. Nous remarquons au passage que la prise en compte d'un grand nombre d'informations ne s'avère efficace que si celles-ci sont pertinentes pour résoudre le problème posé. Il est donc nécessaire de faire coopérer les méthodes pour que soient recherchées des informations spécifiques en fonction des besoins de la situation.

  Par exemple, pour valider un pixel-contour, il est plus intéressant de savoir quel est l'écart-type des niveaux de gris, localement, juste à côté du contour hypothétique (en particulier, si l'écart-type est faible, la présence d'un gradient d'amplitude moyenne valide le contour). Pour avoir des informations sur cette région locale, il est possible d'utiliser une technique de croissance de régions, paramétrée et donc contrôlée en fonction des besoins.

- De même, lors d'un processus de croissance de régions, la fusion entre deux régions dépend souvent des informations sur la moyenne et l'écart-type global des valeurs de gris de celles-ci. Or, lorsque nous effectuons une segmentation "à la main", nous prenons plutôt en compte l'écart-type et la moyenne des valeurs de gris localement, près de la limite entre les

deux régions. Ceci nous parait justifié car la décision de fusion dépend des variations essentiellement locales qui définissent la zone de transition. Pour obtenir ces informations locales, certainement plus complémentaires que les informations globales, il est nécessaire de tenir compte, pour le moins, de la forme et de la localisation de la frontière entre les deux régions, et donc de guider la méthode qui a pour but de fournir ces informations.

Pour accumuler des informations complémentaires, chaque méthode doit préciser ses besoins aux autres méthodes, en déterminant à la fois la manière de les acquérir, le type d'information demandé et la précision requise. Pour cela, une *technique incrémentale* nous semble adaptée. Celle-ci est présentée ci-dessous.

1.3.2.2. La technique incrémentale

Lorsqu'il s'agit de guider une méthode en fonction des informations qui viennent d'être obtenues par d'autres, c'est le principe incrémental qui est implicitement utilisé. Nous proposons en figure 1.4 une détection de contours classique et une détection incrémentale. La différence majeure réside dans l'exploitation immédiate des résultats intermédiaires et la boucle de retour pour revenir traiter un autre endroit de l'image. L'exploitation immédiate des résultats intermédiaires permet en fait un guidage dynamique des méthodes et une focalisation implicite sur le problème courant. Il existe bien sûr de très nombreuses variantes exploitant plus ou moins le principe d'incrémentalité, mais il est tout à fait possible de stocker les résultats intermédiaires, de se focaliser sur un autre endroit de l'image et de ne revenir que bien plus tard exploiter ces résultats pour guider une méthode [15].

On peut remarquer que les algorithmes massivement parallèles permettent une exploitation immédiate des résultats intermédiaires tant que chaque processeur reste focalisé sur un problème local. Même si ces algorithmes ne comportent pas de boucle de retour, ils possèdent donc néanmoins les qualités essentielles des algorithmes incrémentaux. Cette remarque est intéressante si on étudie la vision humaine. En effet, il est actuellement reconnu qu'il n'y a pratiquement pas de boucle de retour vers les neurones correspondant aux premiers traitements visuels. Les traitements étant effectués en parallèle, la boucle de retour qui caractérise l'algorithme incrémental séquentiel n'est effectivement plus nécessaire.

Outre les qualités des algorithmes incrémentaux, retenons donc ici le principe suivant:

PRINCIPE N°2 : Il faut guider les méthodes pour qu'une coopération soit fructueuse et fournisse des informations complémentaires [15].

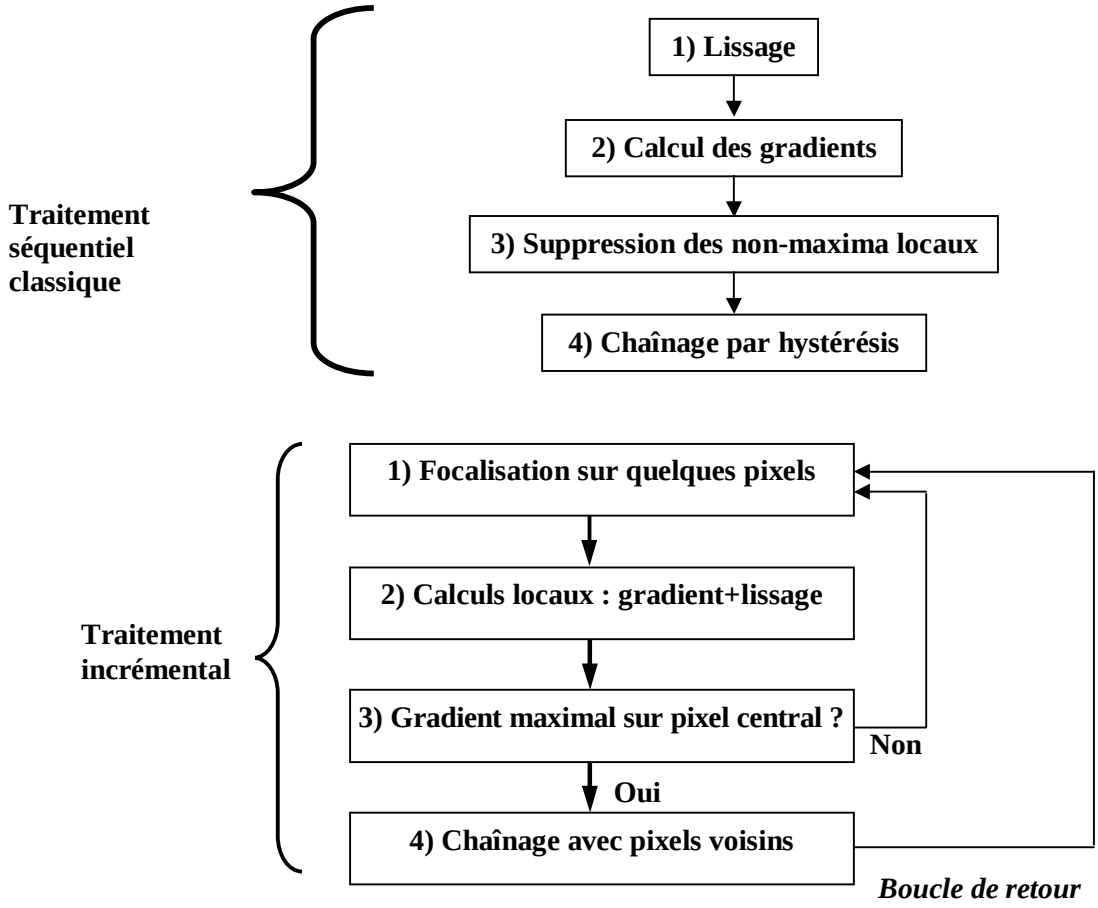


Figure 1-4 : Exemple d'algorithme classique et incrémental

1.3.3. L'émergence de l'information

Le problème de l'émergence des informations est selon nous fondamental, mais rarement apprécié à sa juste valeur, ou mal identifié. Nous commençons donc par définir cette notion, puis nous la discutons et l'illustrons sur un exemple.

1.3.3.1. La notion d'émergence

La notion d'émergence issue des travaux d'intelligence artificielle exprime le fait qu'il est nécessaire d'avoir certaines connaissances locales pour arriver à une autre connaissance plus globale. On parle par exemple de "l'émergence d'un comportement global d'un système grâce au comportement local de ses composantes" [15]. Il existe deux grandes catégories d'émergence de la connaissance.

- La première est l'émergence dans un système réactif. Un comportement global résulte de plusieurs comportements locaux sans qu'il y ait eu ni intention, ni connaissance explicite du problème global. C'est par exemple le cas des neurones qui interagissent entre eux et qui permettent l'émergence de la pensée et de la vision en ce qui concerne

l'être humain, ou plus simplement de l'apprentissage et de la classification de données en intelligence artificielle.

- La deuxième notion exprime l'émergence de l'information attendue par le système, dans notre cas l'indice visuel ou une donnée nécessaire à son identification. Cette deuxième définition diffère de la première au niveau de la connaissance globale du système et de l'intention.

Le problème relatif à l'émergence est essentiellement de collecter des informations "objectives". Nous reprenons ici la définition de "l'objectivité" de Lux [3]. << Une information est objective dans la mesure où sa connaissance permet de résoudre des problèmes >>

Si les informations ne sont pas objectives, alors les connaissances ne vont pas émerger correctement et le problème ne sera pas résolu. Cette notion est en fait très subjective et ne peut être étudiée que de façon qualitative. Néanmoins, elle est très importante car c'est la facilité d'exploitation des indices visuels qui conditionne avant tout les étapes suivantes de la vision. Par exemple, les informations de formes peuvent être codées à l'aide de chaînes de points-contours, de segments, de courbes B-splines, de combinaison de différents types de courbes, de squelettes, etc… . Or, si nous voulons tenter une mise en correspondance entre la forme d'un objet connu et une forme présente dans l'image, le choix que nous ferons pour modéliser cette forme va être déterminant pour la qualité du résultat. Le problème est en fait de savoir si telle ou telle représentation est objective et facilite l'émergence des informations de forme ou au contraire si elle les dissimule. A l'extrême, la forme peut également être représentée par la valeur de gris des pixels. Cependant, autant il est aisé de se rendre compte que cette représentation ne fait pas émerger correctement l'information, autant il est très difficile de comparer l'émergence d'une représentation par B-splines et l'émergence d'une représentation par un ensemble de courbes modèles.

Les problèmes d'émergence des informations se retrouvent à tous les niveaux d'un système de vision. En effet, s'il n'existe encore aucun système relativement général de vision, c'est sans doute en grande partie parce qu'il manque des informations indispensables en sortie des processus de description d'images. Autrement dit, il existe des informations qu'il est difficile de faire émerger et qui sont comme dissimulées dans les indices visuels fournis. Nous étudions ce problème sur un exemple relatif à la reconnaissance de visages.

1.3.3.2. La reconnaissance de visages

La vision humaine est très performante pour la reconnaissance de visages. Ainsi, en apercevant une personne pour la deuxième fois, nous sommes capables d'affirmer dans la plupart des cas que nous l'avons déjà vue. Cette reconnaissance implique l'existence d'indices visuels invariants. Cependant, ces indices semblent être très complexes, puisque les systèmes actuels de reconnaissance de visages n'obtiennent pas encore des résultats du même ordre. Le problème est donc de déterminer quelles sont les informations qui correspondent à ces indices visuels invariants et comment parvenir à l'émergence de celles-ci. Sans résoudre le problème de façon explicite, nous pouvons tout de même aller à la recherche de ces indices pour comprendre ce qui est sans doute la nature de ces informations. En effet, il est remarquable que la vision humaine puisse reconnaître des visages uniquement à partir d'une ébauche des traits les plus marquants du visage. Par exemple, les caricatures de personnages célèbres ne sont que des traits noirs sur du papier blanc, mais cela suffit néanmoins pour que l'identification soit correcte. L'émergence des indices visuels invariants peut donc être ramenée en simplifiant à une association d'informations caractérisant ces traits. Le mot association nous semble ici très important.

Il faut noter que de nombreuses approches ont été tentées en décomposant les traits en courbes ou segments, puis en les associant selon les symétries détectées, mais les résultats ne sont pas toujours probants et les applications sont simplifiées. Pourtant, on peut affirmer que certaines associations d'informations relatives à ces courbes ou segments devraient être suffisantes pour résoudre le problème. Ce sont ces informations associatives qui sont dissimulées derrière une description simple des contours et qui n'émergent pas dans nos algorithmes. La solution devrait donc passer par l'étude de relations moins strictes que la symétrie prenant par exemple en compte des informations sur l'éloignement relatif des traits, et conduisant sans doute à des indices visuels d'un niveau plus complexe pouvant à leur tour être associés selon d'autres principes relationnels [15]. Il est d'ailleurs intéressant de constater que nous n'étudions en général que des relations binaires, alors que les neurones de notre cerveau sont souvent connectés à des centaines d'autres neurones. Cette solution théorique montre clairement l'ampleur du problème lié à ce que nous appelions l'émergence de l'information. Tant que les informations indispensables n'émergeront pas correctement de nos processus descriptifs, il est probable que nous n'obtiendrons que des résultats sous-optimaux.

1.3.3.3. Conclusion sur l'émergence des informations

A travers le problème de l'émergence des informations, c'est donc l'approche générale des problèmes de vision qui est en question. En ce qui concerne les indices visuels primaires, des problèmes similaires existent. En particulier, les régions fournies par la plupart des algorithmes de segmentation ne sont pas toujours très pertinentes. La difficulté est en fait de combiner à la fois plusieurs informations pour faire émerger des régions aux caractéristiques intéressantes. La solution passe sans doute par une explicitation de la connaissance et par un choix "objectif" de structure de données. Bien que la notion d'émergence ait été abordée de façon informelle, nous retiendrons donc qu'il s'agit d'un problème épineux qui conditionne fortement le potentiel de nos algorithmes. Nous en tirons notre troisième principe :

***PRINCIPE N°3* :** Un modèle de description des indices visuels est suffisamment "objectif", s'il permet une exploitation efficace et une émergence d'indices visuels de plus haut niveau [15].

1.3.4. La liberté de l'information

Nous définissons la "liberté de l'information" de façon assez intuitive comme la facilité de création, d'accessibilité et de manipulation de l'information. Nous illustrons nos propos par un exemple.

Si nous procédons à un lissage de l'image selon la technique de Nagao avant d'appliquer un détecteur de contours, il existe de nombreuses informations locales qui sont exploitées dans le premier traitement et qui ne servent plus ensuite. Nous pouvons dire en quelques sortes que ces informations sont "prisonnières" d'une procédure (absence de liberté) et que seule l'image lissée est disponible. Le problème est que ces informations sont du même ordre que celles exploitées par le détecteur et qu'elles auraient donc pu servir à nouveau. La critique que nous venons de formuler est en fait très importante car c'est une des raisons pour laquelle la coopération entre plusieurs méthodes n'est pas toujours fructueuse. Si la coopération présente de nombreux avantages qui résident dans le cumul et la complémentarité des informations, le risque est de garder également les défauts de chaque méthode si celles-ci sont appliquées indépendamment les unes des autres.

Ceci nous amène directement aux problèmes de la structure de contrôle. En effet, pour faciliter la création et la manipulation d'informations d'origines diverses, il est

nécessaire d'avoir une gestion souple qui autorise une focalisation adaptative, un traitement partiel et un retour sur un problème non résolu.

Nous rejoignons ici l'idée d'un traitement incrémental qui permet une exploitation adaptative des informations. Si le principe d'incrémentalité est intéressant, il s'agit en fait de l'utiliser à bon escient selon les besoins. Ainsi, si la gestion des informations est suffisamment souple, il doit être possible de traiter les informations d'une façon séquentielle classique tout en préservant l'opportunité d'un traitement local plus riche.

Tous ces principes se rattachent plus ou moins à la liberté de mouvement de l'information et à l'émergence de celle-ci.

Nous concluons en proposant un dernier principe lié à la liberté de l'information :

***PRINCIPE N°4* :** Pour une gestion efficace des informations, il faut préserver la liberté de mouvement de celles-ci [15].

1.4. La conception des algorithmes

Il existe de multiples façons d'aborder un problème et de proposer une solution algorithmique. En ce qui concerne la construction d'un détecteur d'indices visuels, nous distinguons deux grandes catégories d'approches [16].

- La première est l'étude théorique. Elle consiste à effectuer une modélisation du problème et à découvrir la meilleure solution en fonction de critères définis à l'avance.
- La deuxième est expérimentale. Elle consiste à élaborer une stratégie heuristique qui tente d'approcher la meilleure détection possible, en fonction de critères expérimentaux.

Ces deux approches sont fondamentalement différentes. Alors que dans le premier cas, il s'agit de trouver la solution optimale pour un problème et un but donné, dans le deuxième cas, la solution est approximative mais le problème et le but restent imprécis et dépendent de l'évaluation des résultats expérimentaux.

Il existe, bien sûr, tous les stades intermédiaires, mais c'est en général l'une ou l'autre des deux méthodologies qui est utilisée.

1.4.1. Etude théorique de l'indice visuel

Pour concevoir un algorithme de détection d'indices visuels, l'approche la plus répandue est de proposer un modèle de l'indice ainsi que des critères à respecter, puis de trouver la meilleure solution théorique au problème posé. La solution algorithmique est alors une traduction simple de la solution mathématique. Par exemple, pour la détection de

contours, le modèle de Canny est une "marche d'escalier" bruitée, tandis que ses critères sont :

1) Une bonne détection.
2) Une bonne localisation.
3) Une réponse unique pour une discontinuité unique.

L'intérêt de la modélisation d'un problème est évident. Il s'agit de décrire exactement le cadre de travail, de fixer les limites du problème et d'avancer des hypothèses pour finalement effectuer une analyse mathématique rigoureuse et essayer de dégager une solution. C'est en fait la démarche de base de tout chercheur qui essaie de résoudre un problème. Si aucune modélisation n'est faite, le problème reste mal défini et aucune étude théorique sérieuse ne peut être envisagée. La modélisation permet en fait de donner la spécification complète d'un programme, et d'aborder le problème méthodiquement, au contraire des approches empiriques où le chemin menant à la solution est souvent chaotique [15].

De manière générale, toute étude théorique des indices visuels implique des choix au niveau du modèle de l'indice et au niveau de la définition formelle du but, ce qui place donc les problèmes d'objectivité de l'information au premier plan. Si cette façon d'aborder les problèmes est couramment utilisée et a fait ses preuves dans de nombreux domaines, elle comporte toutefois certains risques que nous rappelons ici :

Le risque majeur de toute étude théorique est de travailler sur un modèle trop simple ou insuffisamment objectif de la réalité. Il existe de très nombreuses configurations de pixels et il est souvent difficile de choisir un modèle sans simplifier à la fois les problèmes et les buts. Par exemple, le modèle de contour de Canny est local (il ne dépend pas des régions voisines ni de l'alignement avec des pixel-contours voisins) et l'analyse du modèle est effectuée dans un espace unidimensionnel. Dans toute tentative de modélisation théorique de la détection d'indices visuels, il est donc très important de donner toutes les limites explicites ou implicites que celle-ci implique par rapport au problème général. Si ces limites ne sont pas claires, alors le risque est de dénaturer le problème, en assimilant le but de la détection des indices visuels à la détection d'un modèle de l'indice.

Une autre difficulté liée à la modélisation de l'indice visuel concerne l'évaluation des résultats et l'amélioration de l'algorithme. En effet, lorsque la solution algorithmique est proche de la détection optimale du modèle, l'évaluation des résultats doit permettre de se rendre compte de la pertinence de celui-ci. Pour une application donnée, il est intéressant

d'exploiter un certain type d'indices et pour une autre application, d'autres indices sont préférables. De plus, pour de nombreuses images, nous ne sommes pas capables de déterminer quelle serait une détection optimale, tout en respectant la prise en compte d'informations strictement locales, donc sans aide contextuelle. Nous pouvons éventuellement comparer deux résultats et préférer celui qui facilite le plus la reconnaissance, mais nous n'avons qu'une vague idée du résultat théorique optimal. La présence de ce flou au niveau de l'évaluation rend difficile le jugement concernant la pertinence de la modélisation. Le modèle du "step edge" en détection de contours a ainsi été adopté par de nombreux chercheurs, tandis que d'autres pensent qu'il n'est pas approprié [15], sans que personne ne soit parvenu à montrer clairement les avantages ou les défauts de ce modèle. Enfin, si la modification d'une théorie est tout à fait possible dans le cas général, en ce qui concerne les indices visuels, la complexité est telle que nos outils mathématiques sont peut-être insuffisants pour envisager des améliorations notables dans la formulation théorique des problèmes. Comment envisager par exemple une amélioration de la modélisation de Canny ? Bien que cela soit possible, la marge de manœuvre est peut-être limitée.

Pour conclure, nous tenons à souligner que nous ne critiquons pas les études théoriques et les modélisations qui ont été faites. Notre propos est de rappeler les limites et les faiblesses de telles approches, dans le cadre de la détection d'indices visuels.

1.4.2. Conception expérimentale d'une stratégie heuristique

Lorsqu'un problème est trop complexe pour que toutes les solutions soient examinées, ou que l'étude théorique s'avère trop difficile, une approche classique en intelligence artificielle consiste à approcher la solution idéale à l'aide d'une stratégie heuristique. La solution trouvée n'est pas toujours la meilleure, mais elle garantit toutefois une qualité minimale. En ce qui concerne la détection d'indices visuels, que signifie une approche heuristique ? Nous l'expliquons de la façon suivante. Le problème est de déterminer la meilleure fonction de décision qui réponde correctement si l'indice est présent ou pas à une position donnée.

Une stratégie heuristique peut très bien s'intégrer dans la modélisation d'un problème. Il s'agit alors d'une fonction de décision qui permet de résoudre une étape précise dont la solution algorithmique exacte est trop complexe ou demande trop de temps de calcul.

La création ou l'ajustement d'une stratégie heuristique se fait généralement en fonction de l'expertise que nous avons du problème. La difficulté majeure de cette approche est la délimitation et la transcription de l'expertise. Pour cela, la démarche classique est la construction d'un système expert qui intègre les différentes règles logiques de notre expertise. Toutefois, en ce qui concerne la détection d'indices visuels, le problème est plus complexe, car d'une part nous n'avons qu'une vague idée de ce qui caractérise un indice visuel, et d'autre part, les relations entre les informations manipulées, telles que niveaux de gris et positions des pixels, sont difficiles à exprimer avec une base de règles.

Par conséquent, pour construire un détecteur d'indices visuels à l'aide d'une stratégie heuristique, il nous semble important de tenir compte de deux principes :

1) Il faut pouvoir évaluer les heuristiques, les comparer et choisir la meilleure stratégie.

   Mais si notre expertise de la détection des indices visuels est approximative, comment parvenir à une évaluation satisfaisante ? Il y a en fait deux aspects à considérer :

   - Le premier concerne l'évaluation quantitative des résultats. Il est nécessaire d'adopter une démarche expérimentale rigoureuse pour évaluer et comparer les différentes approches. Le problème est de définir les critères qui vont caractériser la qualité d'une détection. Ces critères dépendent de la qualité de notre expertise, qui est le deuxième aspect à considérer.
   - Puisque notre expertise est approximative, le problème majeur est en fait l'enrichissement de celle-ci. Pour cela, nous proposons de privilégier la démarche expérimentale afin d'améliorer l'évaluation qualitative des résultats. En effet, n'ayant qu'une vague idée de ce qui caractérise un indice visuel pertinent, il est important de "prendre la place du détecteur" afin de comprendre quelles sont les relations qui existent entre les différentes informations locales et comment les combiner pour discriminer un indice visuel donné.

   En privilégiant l'aspect expérimental, notre expertise grandit ce qui permet de proposer d'autres heuristiques ou même d'envisager une autre manière de traiter le problème.

2) Si nous adoptons une démarche expérimentale et que notre expertise grandit, il reste encore à expliciter et à transcrire cette expertise dans un algorithme. Or, le problème se prêtant difficilement à une décomposition en règles simples, il est nécessaire d'une part de ramener notre expertise à un traitement méthodique, et d'autre part de

concevoir un algorithme performant, capable de traiter des cas complexes avec une structure de contrôle efficace. Les principes que nous avons dégagés dans ce chapitre concernant la gestion des informations méritent ainsi une attention particulière.

La recherche du meilleur compromis entre notre expertise et sa transcription algorithmique est une démarche empirique à plusieurs niveaux. Après évaluation des résultats d'une stratégie de détection, toutes les étapes de l'algorithme peuvent en effet être remises en cause pour adopter une autre gestion des informations. Cette démarche est à opposer à l'approche classique de modélisation du problème avec étude théorique de la meilleure solution. En effet, il n'est pas possible de donner clairement la spécification de l'algorithme car il faudrait expliciter notre expertise, ce qui est justement le problème à résoudre. Si l'approche empirique a le défaut de ne pas proposer de modélisation du problème, l'avantage se situe au niveau expérimental. Alors qu'une étude théorique peut être biaisée par une évaluation insuffisante, due à une expertise incomplète ou incertaine, l'approche que nous venons de décrire tend à réduire le fossé existant entre ce que nous croyons être la détection des indices visuels, et ce qu'il est réellement possible de faire avec un algorithme. Notons que cet avantage est spécifique de la détection des indices visuels car le problème fondamental est certainement celui de la recherche d'une référence solide pour évaluer les résultats. Cette référence, c'est l'expertise humaine, que nous proposons donc d'approcher par une démarche empirique.

1.5. Conclusion

Nous avons présenté une analyse de la gestion des informations pour la détection d'indices visuels primaires, ce qui nous a permis de dégager quelques principes importants : Nous proposons le cumul des informations avant la prise de décision, la nécessité de guider les méthodes afin d'obtenir des informations complémentaires, l'exploitation d'informations "objectives" pour résoudre les problèmes d'émergence des indices visuels et enfin la "liberté" de l'information pour que la structure de contrôle soit efficace et permette par exemple le changement de contexte ou le report des décisions difficiles.

Dans le chapitre suivant, nous allons donner un aperçu sur le domaine de l'analyse d'images en se basant sur l'étape de la segmentation dans la détection des indices visuels primaires qui sont : contours et régions.

CHAPITRE 2
ANALYSE D'IMAGES

2.1. Introduction

Avec la parole, l'image constitue l'un des moyens les plus importants qu'utilise l'Homme pour communiquer avec autrui. C'est un moyen de communication universel dont la richesse du contenu permet aux êtres humains de touts âges et de toutes cultures de véhiculer des informations relatives à leur environnement. Ces informations sont transmises aux individus de la nature, grâce à leurs organes visuels, et interprétées selon des critères de connaissance *a priori*.

L'automatisation de ce mode de communication s'est traduite par la création d'un nouveau domaine englobant des connaissances en physique, en mathématiques et en informatique. C'est ce qu'on appelle le traitement d'images.

Ainsi, le traitement d'images est l'ensemble des méthodes et techniques opérant sur des images, dans le but d'améliorer leur aspect visuel et d'en extraire des informations jugées pertinentes. C'est une aide importante pour l'Homme qui y trouve, en plus, une motivation dans la satisfaction de ses curiosités et les besoins qui en découlent.

Le traitement d'images possède un très large champ d'application touchant plusieurs disciplines. On trouve ses applications dans des domaines très variés tels que l'imagerie aérienne et spatiale, dans laquelle, les traitements concernent l'exploitation des images satellitaires (pour effectuer l'analyse des ressources terrestres, la cartographie automatique et les analyses météorologiques). On peut également citer, les télécommunications (T.V, vidéo, publicité,...), la médecine (radiographie, échographie,...), la biologie, l'astronomie, la géologie, l'industrie (robotique, sécurité), l'architecture, l'imprimerie et l'armement (application militaire). De nouvelles applications sont possibles aujourd'hui et touchent tous les domaines d'activités, tels que : les métiers du spectacle, de la radio, les créations artistiques, etc.

Le domaine de traitement d'images comporte cinq (05) disciplines : l'amélioration, la restauration, la compression, la détection et l'analyse. Notre intérêt se situ dans la dernière, c'est-à-dire : l'analyse d'images.

2.2. Le processus d'analyse d'images

Depuis les années 1950, l'association de l'image et de l'ordinateur connaît un essor considérable, tant en ce qui concerne le domaine de l'analyse que celui de la synthèse d'images. L'analyse d'images et la vision par ordinateur s'attachent à construire une description explicite du contenu de l'image, tandis que la synthèse d'images part de la modélisation d'une scène pour construire une image cohérente.

Le processus d'analyse d'images, qui a pour but de fournir une description ou une interprétation d'une scène à partir de l'information extraite de l'image, peut être décomposé en plusieurs étapes, comme le montre la figure 2.1.

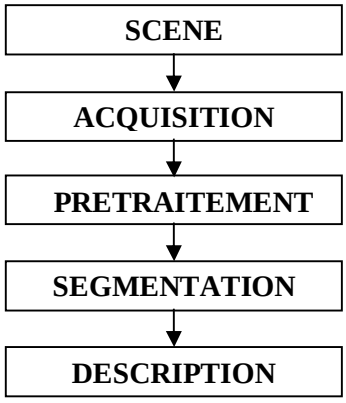


Figure 2-1 : Etapes du processus d'analyse d'images

Au début, on doit faire l'acquisition d'une scène, en discrétisant l'image réelle continue. Normalement la quantité d'information brute initiale, après la discrétisation, est très volumineuse et difficile à manipuler. De plus, cette discrétisation entraîne une perte d'information, de même qu'elle nous pose des problèmes d'ordre technique, comme le bruit et la texture, entre autres.

Etant bruitées ou floues, ces images doivent passer par un prétraitement avant toute opération de détection, où l'information dégradée est restaurée. Dans cette étape, des détails de l'image peuvent être rehaussés par plusieurs techniques.

Ensuite vient la phase la plus importante et la plus difficile du processus d'analyse d'images : la segmentation. Segmenter une image correspond à trouver les régions qui ont un sens ou détecter leurs frontières. Cette étape doit permettre d'interpréter la scène aussi bien que le ferait l'observateur.

A partir de là, viennent les traitements de haut niveau, tels que la description de l'image, la reconnaissance des formes et les décisions qui pourront être prises à partir des résultats fournis par la segmentation.

Comme nous l'avons remarqué, pour arriver à la reconnaissance des formes dans une image, plusieurs étapes sont nécessaires. Nous allons les détailler dans ce qui suit.

2.3. Prétraitement

L'étape de prétraitement est une étape essentielle avant tout traitement. Son but est l'amélioration d'images ayant été dégradées pour des raisons diverses. En effet, cette étape facilite largement celle de segmentation, en renforçant la ressemblance entre les pixels appartenant à la même région ou, en accentuant la différence entre pixels appartenant à des régions différentes.

Les méthodes de prétraitement se résument en trois grandes parties: le filtrage des bruits, modification de l'histogramme et le rehaussement de contraste dont les définitions sont décrites ci-dessous.

2.3.1. Réduction de bruit par filtrage

Une image est généralement affectée par des perturbations, désignées sous le terme de bruit de l'image.

L'objectif du filtrage du bruit (lissage) est de réduire l'amplitude des variations d'intensité dans chaque région, tout en conservant les transitions entre régions adjacentes. Cette transformation est souvent nécessaire avant la segmentation, en particulier la détection de contours [38].

Il existe une grande variété de filtres plus ou moins utilisés et il serait fastidieux de vouloir tous les décrire, nous avons choisi de citer les plus connus.

2.3.1.1. Filtre linéaire

Où la transformation d'un pixel est le résultat d'une combinaison linéaire des pixels voisins. Et parmi les filtres utilisés on peut citer :

a) Le filtre moyen

On désire filtrer une image bruitée, c'est-à-dire, si ce n'est pas éliminer complètement le bruit, du moins en diminuer sensiblement les effets. Une méthode simple consiste à considérer chaque point de l'image et d'en faire la moyenne avec les huit pixels qui lui sont voisins. Ceci va avoir pour effet d'adoucir l'image en réduisant les fluctuations des niveaux de gris [9].

Ce type de filtre utilisant la moyenne non pondérée des voisins peut être mis sous la forme d'un masque tel que celui-ci :

$$H_1 = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad \text{.....(2.1)}$$

On va alors déplacer ce masque sur toute l'image, le pixel affecté par la transformation étant le pixel central du masque. Le facteur 1/9 est égale à la somme des coefficients et sert à normaliser le filtre de manière que celui-ci n'influe pas sur l'intensité globale de l'image [9].

Si nous posons I_1 comme étant l'image de départ, I_f l'image résultat et H_1 le masque, nous avons alors pour chaque pixel de coordonnées x,y [9]:

$$I_f(x,y) = \frac{1}{9}\sum_{i=-1}^{1}\sum_{j=-1}^{1} H_1(i+1, j+1) \times I_I(x+i, y+j) \quad \text{.....(2.2)}$$

De manière plus générale, pour un filtre H de taille (n+1)x(n+1) avec n paire et dont la somme des coefficients vaut k [9]:

$$I_f(x,y) = \frac{1}{k}\sum_{i=-n/2}^{n/2}\sum_{j=-n/2}^{n/2} H_1(i+n/2, j+n/2) \times I_I(x+i, y+j) \quad \text{.....(2.3)}$$

La formule précédente n'est autre que l'expression en chaque point de l'image d'un produit de convolution discrète et on peut écrire pour l'image entière [9]:

$$I_f = H \otimes I_I \quad \text{.....(2.4)}$$

Où $\otimes$ symbolise l'opérateur de convolution.

D'autres filtres ont été réalisés en utilisant des coefficients de pondération différents, par exemples [9] :

$$H_2 = \frac{1}{10}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad H_3 = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Figure 2-2 : Exemple des filtres moyens avec des coefficients de pondérations

b) filtre gaussien

Le filtre gaussien est un filtre linéaire parmi les plus courant, tant par sa facilité de mise en œuvre que par la bonne qualité de ses résultats. Ce filtre tire son nom de la valeur de ses coefficients qui sont ceux d'une courbe de Gauss à deux dimensions.

Rappel :

**** courbe de Gauss à une dimension***

Rappelons l'équation de la courbe de Gauss :

$$G(x,s) = \frac{1}{\sqrt{2p}\,s}\exp\left(-\frac{x^2}{2s^2}\right) \quad \text{.....(2.5)}$$

Cette courbe est positive, symétrique (voir figure 2.3) et possède les propriétés suivantes [9] :

$$\int_{-\infty}^{+\infty} G(x,s)dx = 1 \quad \text{.....(2.6)}$$

$$\int_{-2s}^{+2s} G(x,s)dx = 0.95 \quad \text{.....(2.7)}$$

$$\int_{-3s}^{+3s} G(x,s)dx = 0.999 \quad \int_{-3s}^{+3s} G(x,s)dx = 0.999 \quad \text{.....(2.8)}$$

**** Cas bidimensionnel – calcul du filtre***

Un filtrage Gaussien consiste en la convolution d'une image I_1 avec une gaussienne G(x,y,σ) :

$$I_f = I_I \otimes G \quad \text{.....(2.9)}$$

Avec : $G(x,y,s) = \frac{1}{\sqrt{2p}\,s} \exp\left(-\frac{x^2+y^2}{2s^2}\right)$ .....(2.10)

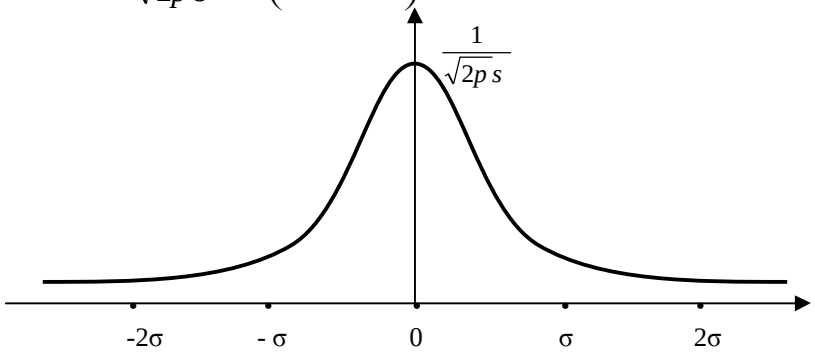


Figure 2-3 : Courbe Gaussienne

Où G est un masque carré dont les coefficients sont les éléments discrétisés de la gaussienne.

Comme le masque est de taille (n+1)x(n+1) finie et que la courbe de gausse est définie de moins l'infinie à plus l'infini, il est nécessaire d'en prendre une portion finie. Or nous avons vu que pour une largeur de 4σ, nous avions 95% de la courbe, et 99.9% pour 6σ, il faut donc pour obtenir une bonne approximation, choisir un masque G de taille N égale à au moins 4σ [9].

Le calcul des coefficients du filtre s'effectue de la manière suivante [9]:

- Choix de N = 6σ + 1, taille du filtre en pixel.
- Calcul de σ = (N-1)/6
- ***Pour*** i = 0 à N-1 ***Faire***
- $G[i] = \frac{1}{\sqrt{2p}\,s} \exp\left(-\frac{\left(i-\frac{N-1}{2}\right)^2}{2s^2}\right)$ .....(2.11)
- ***Fin pour***

2.3.1.2. Filtre non linéaire

Un filtrage non linéaire est un filtrage où les pixels voisins interviennent suivant une loi non linéaire. Ce filtrage permet la mise en évidence des transitions entre les plages de niveaux de gris différents d'où un filtrage d'accentuation des contours. Parmi ces filtres :

a) Filtrage médian

Un filtre médian est un filtre non-linéaire qui produit comme résultat le médian des valeurs traitées.

Son principe est d'associer à chaque pixel le médian de ses voisins et les étapes à suivre sont les suivantes [46]:

- Lire les valeurs des pixels dans un voisinage d'un pixel donné.
- Tirer ses valeurs par ordre croissant.
- Remplacer la valeur du pixel par la valeur située au milieu de la liste triée.
- Répéter cette opération pour tous les pixels de l'image.

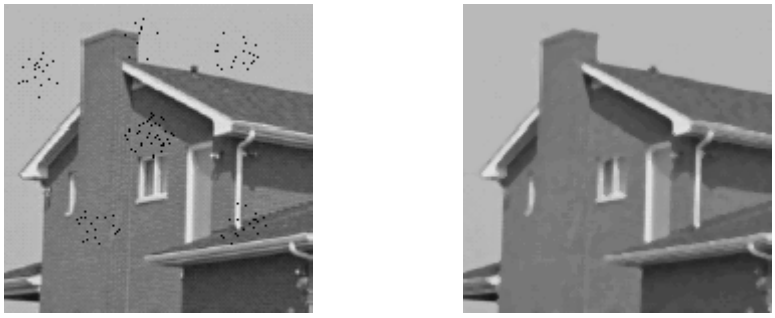


Image Maison bruité **Image Maison bruité après le filtrage médian**

Figure 2-4 : Exemple d'un filtrage médian

b) Filtre de Nagao

Le Filtre de Nagao permet d'extraire des régions homogènes d'une image 'complexe'. Le principe du filtre est de faire tourner un motif (masque de Nagao) autour du point à traiter et de chercher le masque pour lequel la variance est minimale.

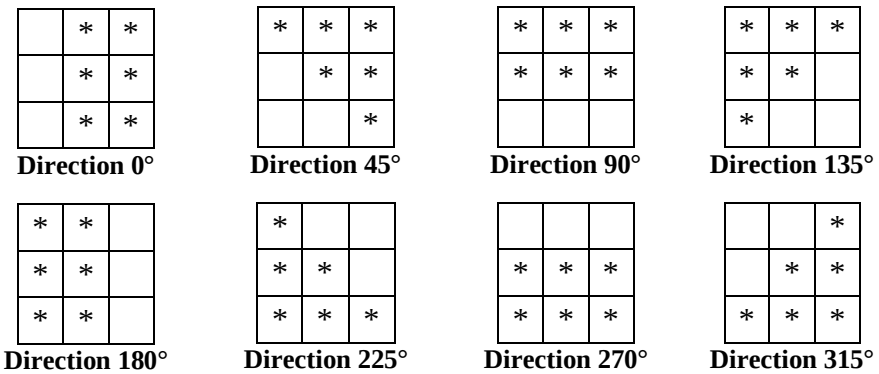


Figure 2-5 : Les différentes directions du masque de Nagao 3*3

2.3.2. Modification d'histogramme

Appelée aussi «transformations ponctuelles de luminance ». Quand l'image est mal contrastée, on peut par transformation d'histogramme redistribuer les niveaux de gris pour occuper toute l'échelle des niveaux de gris possible. Cette transformation permet de bien exploiter l'échelle des niveaux de gris disponible, et n'affecte pas la forme des régions, mais modifie seulement l'apparence visuelle [18]**.**

Parmi les techniques les plus utilisées, on peut citer :

- L'expansion de la dynamique.
- L'égalisation d'histogramme.

2.3.2.1. Tables de conversion ou LUT (Look Up Table)

Une LUT est une fonction qui transforme un niveau de gris *i* en un niveau de gris *j*, et ce sans modification de la structure spatiale de l'image, c'est-à-dire que la transformation du niveau de gris d'un pixel est indépendante des pixels voisins. Plus mathématiquement, une LUT *f* est définie comme suit [9] :

Soit G_1 l'ensemble des niveaux de gris de départ : $G_1 = \{0, 1, \ldots, N_i\}$

Soit G_F l'ensemble des niveaux de gris de l'image résultat : $G_F = \{0, 1, \ldots, N_F\}$

f est une application de G_1 dans G_F telle que :

$$\forall g_1 \in G_1, \exists\ g_F \in G_F : g_F = f(g_1)$$

La figure 2.6 illustre ceci : à chaque valeur initiale de gris correspond un niveau final. On voit que la conversion d'un niveau initiale à un niveau final peut être réalisée par une table, d'où le nom de LUT. Une fois cette table construite, le principe de la LUT est simple : on examine le niveau de gris de chaque point de l'image de départ et l'on déduit à l'aide de la table le niveau de gris à lui donner dans l'image résultat. La table se présente sous la forme d'un vecteur L à N_i+1 éléments dont chaque élément L(i) contient la valeur du niveau de gris résultat, i étant le niveau de gris en entrée [9].

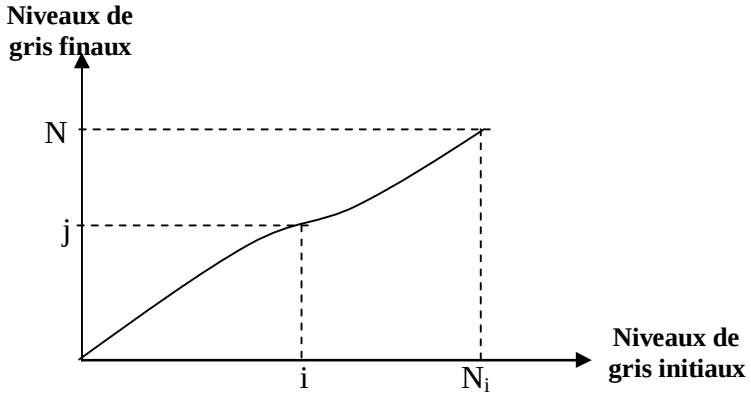


Figure 2-6 : Définition d'une LUT.

2.3.2.2. Recadrage de dynamique

L'une des applications les plus courantes des LUT est le recadrage de dynamique [9]. Certaines images sont initialement trop claires, trop foncées, ou bien peut contrastées. Cela est du respectivement au fait que les niveaux de gris de l'image sont tassés vers le haut de l'échelle, vers le bas, ou bien sont regroupés dans un intervalle étroit. Ce défaut est très visible sur l'histogramme, comme la montre la figure 2.7. Le but de la LUT est alors de redistribuer les niveaux de gris de l'image afin de leur faire occuper toute la bande de nuances possibles. C'est ce que réalise le recadrage de dynamique comme il est indiqué dans la Figure 2.8.

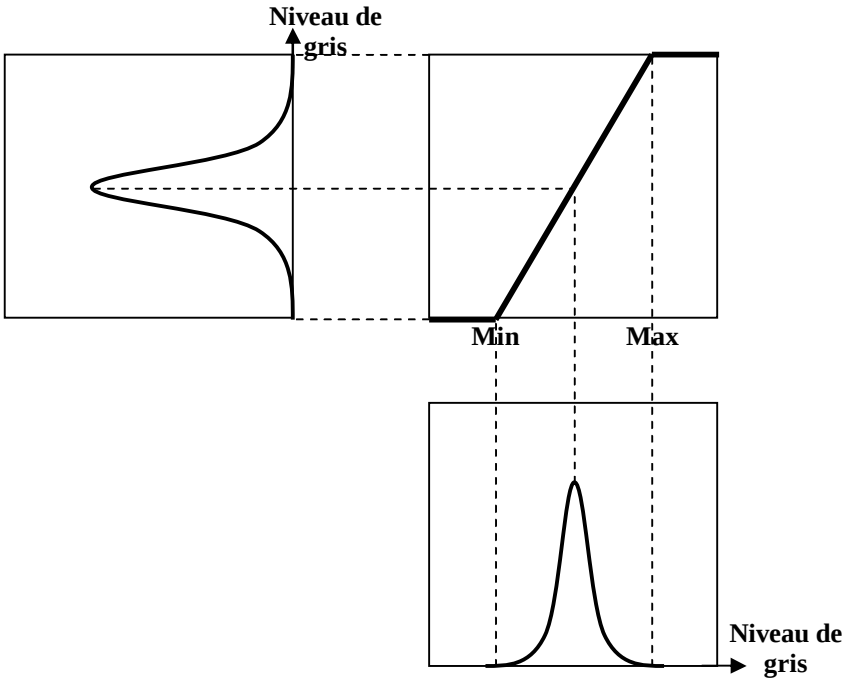


Figure 2-7 : Principe de recadrage de dynamique

Le choix des valeurs min et max du recadrage peut être effectué soit manuellement, soit automatiquement en décidant d'un certain pourcentage de pixels recadrés. Par exemple, 5% des pixels entre 0 et min, 5% entre max et 255. On recadre ainsi 90% de l'histogramme, les 10% restants étant forcés pour moitié à 0 et pour moitié à 255. Il est à noter que ces plages de valeurs extrêmes seront perdues après recadrage. Donc un détail de l'image composé de niveaux de gris faibles compris entre 0 et min disparaîtra. Bien que ce cas de figure soit rare (on s'intéresse rarement aux détails de ce type dans une image), il est bon de prendre conscience qu'un recadrage de dynamique peut entraîner une perte d'information, même minime, sur l'image [9].

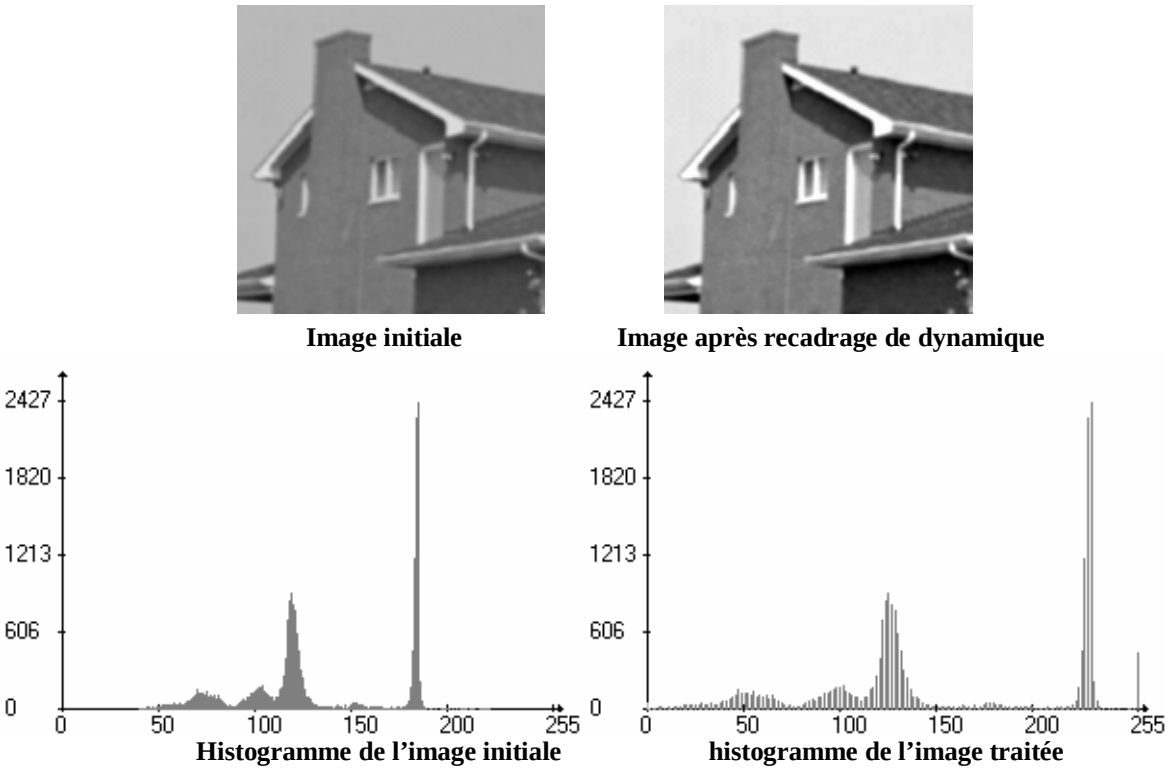


Figure 2-8 : Exemple de recadrage de dynamique

2.3.2.3. Méthodes d'égalisation

L'égalisation procède de la manière suivante [9]:

- on part du niveau zéro et on calcul la valeur cumulée des niveaux suivants dans l'histogramme original jusqu'au moment où la somme est le plus prêt possible de la valeur moyenne idéale $(L.H)/N_F$. (L : largeur, H : hauteur de l'image)
- tous les niveaux de l'histogramme initial qui ont participés à cette somme sont alors recadrés sur un niveau final unique qui se trouve au centre de la première bande de l'histogramme final. Dans l'exemple de la figue 2.9, la somme des valeurs des niveaux 0, 1, 2 de l'histogramme original est proche de la moyenne de l'histogramme final idéal. Or, celui-ci comprend cinq bandes de niveau. La première bande étant celle des niveaux 0 à 2, le centre de cette bande et donc 1. cela signifie que lorsque aura lieu la transformation de l'image proprement dite, tous les niveaux de l'image de départ dont les valeurs seront 0,1, ou 2 seront remplacés par le niveau unique 1 dans l'image résultat [9].
- On recommence ensuite l'opération pour les bandes suivantes. On peut remarquer que plus le nombre de niveaux finaux N_F est restreint, plus l'intégration est importante du fait de la valeur plus élevée de la valeur $(L.H)/N_F$. en général on utilise $NF = N_i/2$ qui est l'intégration minimale. Il suffit alors de répéter le processus d'égalisation plusieurs fois sur l'image afin d'obtenir une intégration

plus importante. Prenons un exemples numérique : pour une image 128 x 128 codée sur 256 niveaux, si l'on veut un histogramme égalisé sur 64 bande la valeur moyenne idéale dont il faut se rapprocher est $128^2/64=256$.

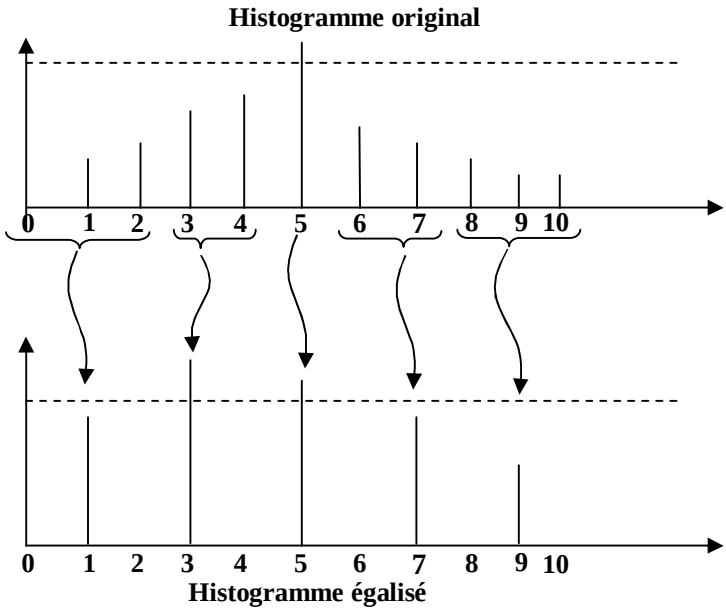
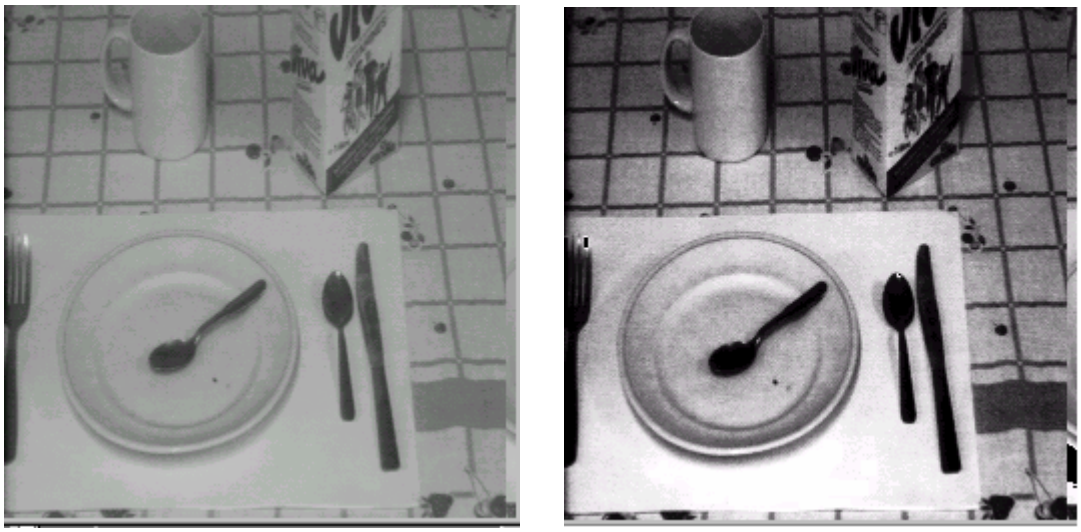


Figure 2-9 : Principe de l'égalisation de l'histogramme



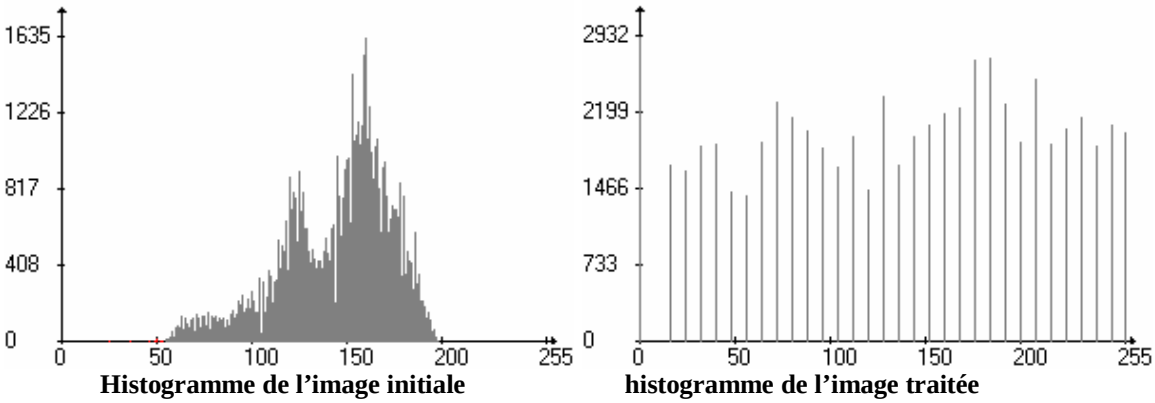


Figure 2-10 : Exemple de recadrage de dynamique

2.3.3. Rehaussement de contraste

Quand les changements d'intensité dus aux frontières de régions sont larges (flous), c'est-à-dire s'étalent sur plusieurs pixels, il est possible de faire rehaussement de contraste de l'image, en diminuant l'étendue de la zone de transition, tout en préservant l'homogénéité des régions de part et d'autre de la frontière. Ainsi, on ne risque pas de fusionner intempestivement deux régions distinctes lors d'une opération de segmentation**.**

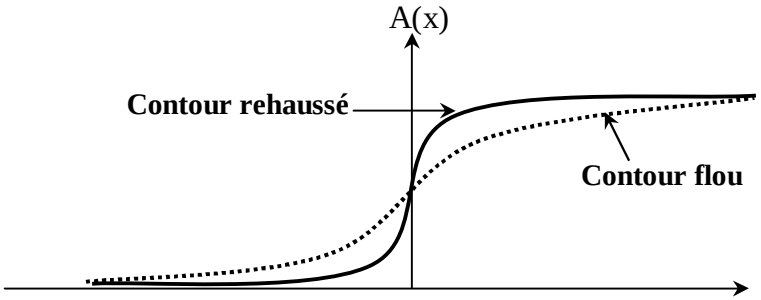


Figure 2-11 : Rehaussement de contraste

Pour rehausser une image, on utilise une méthode basée sur le Laplacien selon l'expression suivante : Image rehaussée = Image initiale – λ.Laplacien de l'image initiale
Avec $0 < \lambda < 1$

Le principe du Laplacien est décrit dans le paragraphe 2.4.3.2.

2.4. Segmentation

2.4.1. Définition

La segmentation est sans doute la tâche qui, en analyse d'images, mobilise le plus d'efforts. Certes, cette étape importante du traitement n'apparaît pas toujours de façon explicite, mais on peut affirmer qu'elle est toujours présente, même lorsque les images à analyser sont simples.

La segmentation a en effet plusieurs cheminements, selon le type d'images sur lesquelles on travaille, selon la nature des outils de segmentation utilisés et surtout selon ce que l'on attend de cette procédure. De manière intuitive, on peut définir la segmentation d'images comme son partitionnement en zones d'intérêts selon des critères de ressemblance ou de dissemblance. Son rôle est l'extraction d'objets présents dans l'image, de façon aussi exacte que possible. Ceci se fait par transformation de l'image en régions d'intérêts selon des propriétés locales telles que le groupement de pixels ayant des propriétés communes. [8]

Les deux approches les plus sollicitées de segmentation sont celles en **détection de contours** qui répondent au principe de discontinuité entre deux pixels et en **analyse en**

régions qui elles, se basent sur le principe de similarité. Une troisième approche consiste à utiliser les deux premières méthodes de segmentation ; c'est segmentation coopérative contour-région et région-contour [37].

2.4.2. Choix d'une méthode de segmentation

Il n'y a pas de méthodes uniques de segmentation d'une image, le choix d'une technique peut être lié [46]:

1) à la nature de l'image (éclairage non homogène, présence de bruit, contours flous, …)
2) Aux opérations situées en aval de la segmentation (localisation, mesure, reconnaissance de formes, interprétation, …etc.)
3) Aux primitives à extraire (contours, formes, régions, textures, …etc.)
4) Aux contraintes d'exploitation (complexité algorithmique, taille de la mémoire disponible, …etc.)

Du fait de cette diversité, il est difficile de définir de manière absolue, une bonne méthode de segmentation.

2.4.3. La segmentation par extraction de contours

La détection de contour est une étape préliminaire à de nombreuses applications de l'analyse d'images. Les contours constituent en effet des indices riches pour toute interprétation ultérieure de l'image.

Les contours sont des variations d'intensité qui représentent des changements de propriétés physiques ou géométriques de la scène ou de l'objet observé correspondant par exemple à : [18]

- Des variations d'illumination, des ombres,
- Des changements d'orientation ou de distance à l'observateur,
- Des changements de réflectance de surface,
- Des variations d'absorption es rayons (lumineux, X, etc.)

Il existe cependant plusieurs méthodes de détection de contours selon le type de variation des intensités des niveaux de gris, nous citons par exemple [34]:

- Approche dérivative.
- Approche surfacique.
- Approche morphologique.
- Approche markovienne.
- Approche par la logique floue.
- Approche connexionniste, par réseaux de neurones.

Dans la suite de ce travail, on s'intéressera à la méthode dérivative seulement.

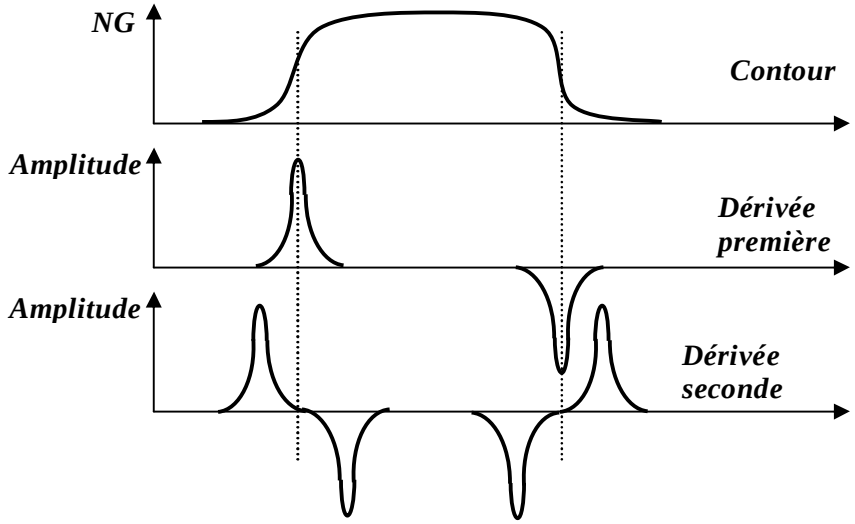


Figure 2-12 : Dérivation en présence d'un contour.

2.4.3.1. Le gradient

Le gradient des niveaux de gris en un point d'une image 2D est un vecteur constitué des deux dérivées partielles de la fonction de luminance :

$$G = \begin{vmatrix} \frac{d\ f(x,y)}{d\ x} \\ \frac{d\ f(x,y)}{d\ y} \end{vmatrix} \quad \text{.....(2.12)}$$

D'un point de vue géométrique, le gradient évalué sur un point indique la direction où la variation des niveaux de gris est la plus grande. Ce vecteur a une longueur très faible sur les points se trouvant à l'intérieur d'une région homogène. Par contre, sur les points se trouvant à la limite entre deux régions, le gradient a une longueur importante, il est orienté vers la région la plus claire et se trouve perpendiculaire au contour. D'un point de vue pratique, les deux dérivées partielles suivant la largeur et la hauteur d'un point de l'image I(x,y) peuvent être exprimé comme suite [9]:

Ax = I(x+1,y) – I(x,y) .....(2.13)

Ay = I(x,y+1) – I(x,y) .....(2.14)

Ce qui revient à convoluer l'image avec les masques :

Hx = [-1 1] .....(2.15)

$$Hy = \begin{pmatrix} -1 \\ 1 \end{pmatrix} \text{.....(2.16)}$$

L'amplitude est alors :

$$A(x,y) = \sqrt{Ax^2 + Ay^2} \text{ .....(2.17)}$$

La direction du gradient est donnée par :

$D(x,y) = \mathrm{ArcTan}(Ay/Ax)$ .....(2.18)

Ce gradient est appelé « Opérateur de Roberts ».

A cet opérateur ont succédé les gradients de Premitt et Sobel. Contrairement à l'opérateur de Roberts, ils présentent l'avantage d'être moins sensible au bruit. En effet, tout processus de dérivation d'un signal tend à accentuer le bruit présent dans ce signal. Cette sensibilité a été diminuée en effectuant une moyenne locale sur le domaine couvert par le masque [9].

$$Hx = \begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix} \qquad Hy = \begin{matrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{matrix}$$

Figure 2-13 : Masques de Sobel

$$Hx = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \qquad Hy = \begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}$$

Figure 2-14 : Masques de Prewitt

Dans ce but, ont été développés des masques dits optimaux tels que l'opérateur de Kirsh. Cet opérateur associe un masque à chaque direction détectable dans le voisinage carré de taille 3*3, il existe huit directions possibles du gradient qui sont 0, 45, 90, 135, 180, 225, 270 et 315 degrés et qui correspondent chacune à la direction entre le point central du voisinage et les huit voisins. Dans ce cas, on définit huit masques $H_1,\ldots H_8$ comme ci-dessous [9]:

$$H_1 = \begin{matrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{matrix} \qquad H_2 = \begin{matrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{matrix} \qquad H_3 = \begin{matrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{matrix} \qquad H_4 = \begin{matrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{matrix}$$

$$H_5 = \begin{matrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{matrix} \qquad H_6 = \begin{matrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{matrix} \qquad H_7 = \begin{matrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{matrix} \qquad H_8 = \begin{matrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{matrix}$$

Figure 2-15 : Masques de Kirsh

Ce qui définit des gradients sur huit directions. Si A_1, A_2,…, A_8 sont les résultats de chacun des masques, l'amplitude du gradient est alors donnée pour chaque pixel par [9]:

$A(x,y) = \mathrm{Max}\{A_i(x,y)\}$ $i = 1,2,\ldots,8$ .....(2.19)

Et l'indice « i » correspondant au A_i maximum donne directement la direction qui est dans le cas 3*3 : Dir = 45° * i.

2.4.3.2. Le Laplacien

Au lieu de rechercher les maxima locaux du gradient de l'image, on peut identifier les contours comme les passages par zéro de la deuxième dérivée dans la direction du gradient [33]. Il existe cependant plusieurs masques de Laplacien tous très similaires tels que ceux présentés ci-dessous :

									0	0	-1	0	0
									0	-1	-2	-1	0
0	-1	0		-1	-1	-1		1	-2	1	-1	-2	16	-2	-1
-1	4	-1		-1	8	-1		-2	4	-2	0	-1	-2	-1	0
0	-1	0		-1	-1	-1		1	-2	1	0	0	-1	0	0

Figure 2-16 : Quelques masques de Laplacien

Le Laplacien d'un point dans une région homogène vaut zéro. Par contre, on observe une forte oscillation avec un passage par zéro lorsque l'on traverse un contour suivant sa normale. L'image affichée après normalisation (entre [0, 255]) sera gris moyen avec les contours d'objets marqués par un double trait noir/blanc. Si le calcul du Laplacien est d'une complexité plus faible que celui du gradient (une seule passe sur l'image au lieu de deux), il ne donne aucune information sur la direction du contour [9].

2.4.3.3. Seuillage et affinage des contours

Lorsque le gradient a été calculé, un simple seuillage de ces valeurs fournit déjà de bonnes informations sur la présence des discontinuités. Cependant, les contours sont généralement épais, c'est à dire que pour une même discontinuité, plusieurs pixels ont une valeur de gradient supérieure au seuil. Une étape d'affinage des contours est nécessaire. Une des techniques les plus simples consiste à remettre à zéro toutes les valeurs de gradient qui ne sont pas maximales par rapport à celles des pixels voisins situés dans la direction du gradient. En effet, la valeur du gradient doit être maximale localement à la position exacte du contour et décroître de part et d'autre de celui-ci dans la direction du gradient.

D'autres techniques d'affinage existent. Par exemple, il est possible d'appliquer des filtres morphologiques, érosion et dilatation, ou de calculer le squelette. Ces techniques sont intéressantes mais elles sont parfois moins précises pour la localisation car elles ne tiennent pas compte de la valeur du gradient.

Un seuillage simple s'avère en général insuffisant. Des faux pixel-contours sont détectés dans les zones bruitées ou les zones de texture (même légèrement) et des pixel-

contours importants dont le gradient est faible sont oubliés. D'autres techniques de seuillage existent, nous en détaillons quelques unes.

- Le seuillage par hystérésis est la technique la plus répandue. Proposée à l'origine par Canny [4] en même temps que son détecteur, cette technique consiste à choisir deux seuils s1 et s2 (avec s1 > s2) et à effectuer un chaînage en opérant une sélection des gradients de la façon suivante : l'image est parcourue de haut en bas et de gauche à droite (par exemple). Dés qu'une valeur de gradient dépasse s1, elle constitue le départ de la chaîne. On essaie alors de continuer la chaîne dans les deux sens, en fonction de l'orientation du gradient. Tant que le gradient sur le pixel suivant est supérieur à s2, on continue la chaîne, sinon une extrémité de la chaîne est atteinte. Lorsqu'une chaîne est terminée, le parcours de l'image reprend là où il s'était arrêté. Tous les pixels chaînés sont bien sûr marqués pour qu'ils ne soient pas traités plusieurs fois.
- Lorsque le problème est l'estimation du bruit pour adapter automatiquement le seuillage, Haddon propose d'évaluer le bruit de façon statistique à partir de deux images prises exactement dans les mêmes conditions [34]. Toutes les variations d'une image à l'autre sont essentiellement dues à la digitalisation et au bruit électronique, il suffit donc de calculer la variance entre les deux images pour estimer le bruit et proposer un seuillage adaptatif.
- Zuniga et Haralick propose d'utiliser le "modèle des facettes" pour estimer le bruit local et adapter le seuillage des gradients [18]**.** L'image est assimilée à une surface échantillonnée dont la troisième dimension est la valeur de gris. A partir d'un voisinage local, il est possible d'estimer les paramètres de la surface locale (la facette) ainsi que le bruit par des méthodes statistiques. Le seuil adaptatif du gradient est alors donné à l'aide d'une fonction Bayésienne. Les résultats qu'ils présentent sont intéressants. Toutefois, Zuniga et Haralick se sont restreints à des images synthétiques bruitées avec peu de contours dans l'image.
- Quiguer, Mich. et Debrie proposent un seuillage adaptatif à partir de l'analyse de l'histogramme des déclivités [15]. Une déclivité est définie comme l'écart entre deux extremums locaux voisins dans une direction donnée, horizontale ou verticale. Ils supposent que l'histogramme doit avoir deux modes, le premier correspondant au bruit, le second aux contours. Le seuil choisi est situé dans la vallée entre les deux modes. Cette technique a été utilisée pour obtenir les contours horizontaux et verticaux de l'image dans un contexte temps réel.

Notons finalement qu'il existe d'autres techniques simples comme un seuillage en fonction de la moyenne des gradients sur l'image ou sur des zones de l'image, un seuillage en fonction de l'écart-type global du gradient ou un seuillage en fonction du pourcentage attendu de pixel-contours par rapport au nombre total de pixels.

2.4.3.4. Le prolongement, la correction et le chaînage

Un prolongement des contours peut s'avérer nécessaire pour deux raisons.

Premièrement, les résultats du détecteur ne sont pas toujours de grande qualité et de nombreux contours tronqués doivent être rallongés et raccordés. Deuxièmement, pour certaines applications, il est intéressant d'avoir des contours fermés pour travailler ensuite sur les formes obtenues. Il existe de très nombreuses techniques pour prolonger les contours. La plupart d'entre elles sont fondées sur la recherche d'un parcours optimal dans un graphe. De façon générale, il s'agit d'exploiter les informations sur le gradient, associant intensité et direction, ainsi que l'orientation à l'extrémité pour générer des prolongements jusqu'à ce qu'un critère d'arrêt soit vérifié, comme par exemple la proximité d'un autre contour, un gradient trop faible ou un nombre limite de pixel-contours ajoutés. Des techniques de programmation dynamique, de recherche heuristique, ou de construction d'automates d'états finis ont été développées dans ce but. Deriche et Coquerez [18] présentent ainsi un processus de fermeture de contours fondé sur l'étude de contours d'épaisseur un pixel ; il est alors aisé de détecter les extrémités des contours. Le procédé élabore à partir de chaque extrémité de contour, tous les chemins possibles d'une longueur donnée, et ne retient que le meilleur (la fonction de coût peut être la somme des normes du gradient des points qui composent les chemins).

Il est également possible de raisonner uniquement sur les chaînes de pixel-contours : si deux contours sont suffisamment proches l'un de l'autre et à peu prés alignés, ils sont prolongés et raccordés.

Le prolongement des contours est souvent considéré comme une étape de correction, puisque les informations utilisées pour rallonger les contours restent essentiellement les mêmes que pour la détection des contours, c'est à dire l'intensité et la direction du gradient. Dans certains cas cependant, les valeurs de gradient ne permettent pas de déterminer la présence ou la fin d'un contour. Un éventuel prolongement ne relève alors pas de la détection des contours, mais plutôt des techniques d'association inspirées de la théorie Gestaltiste, avec notamment la continuité de la forme.

Parmi les autres moyens de correction, citons également la suppression des petites chaînes de contour, qui bien que très arbitraire, permet d'améliorer les résultats de façon non négligeable.

En ce qui concerne le chaînage, il est en fait souvent réalisé en même temps que le seuillage par hystérésis, et parfois en même temps que le prolongement. Les seuls problèmes sont d'ordre algorithmique avec le traitement des jonctions et un parcours en une seule passe de la carte des contours.

2.4.3.5. Méthode de CANNY

Un simple calcul du gradient ou du Laplacien s'avère trop simple. De nombreuses recherches ont été faites pour modéliser une transition plus sophistiquée et concevoir un filtre dont la réponse est maximale en présence du modèle. Nous développons ici l'approche de Canny [4]. Cette approche est intéressante car leur méthodologie est une caractéristique des techniques de conception d'un détecteur d'indices visuels à partir d'une étude théorique, techniques que nous avons abordées au premier chapitre.

Organigramme du détecteur de CANNY :

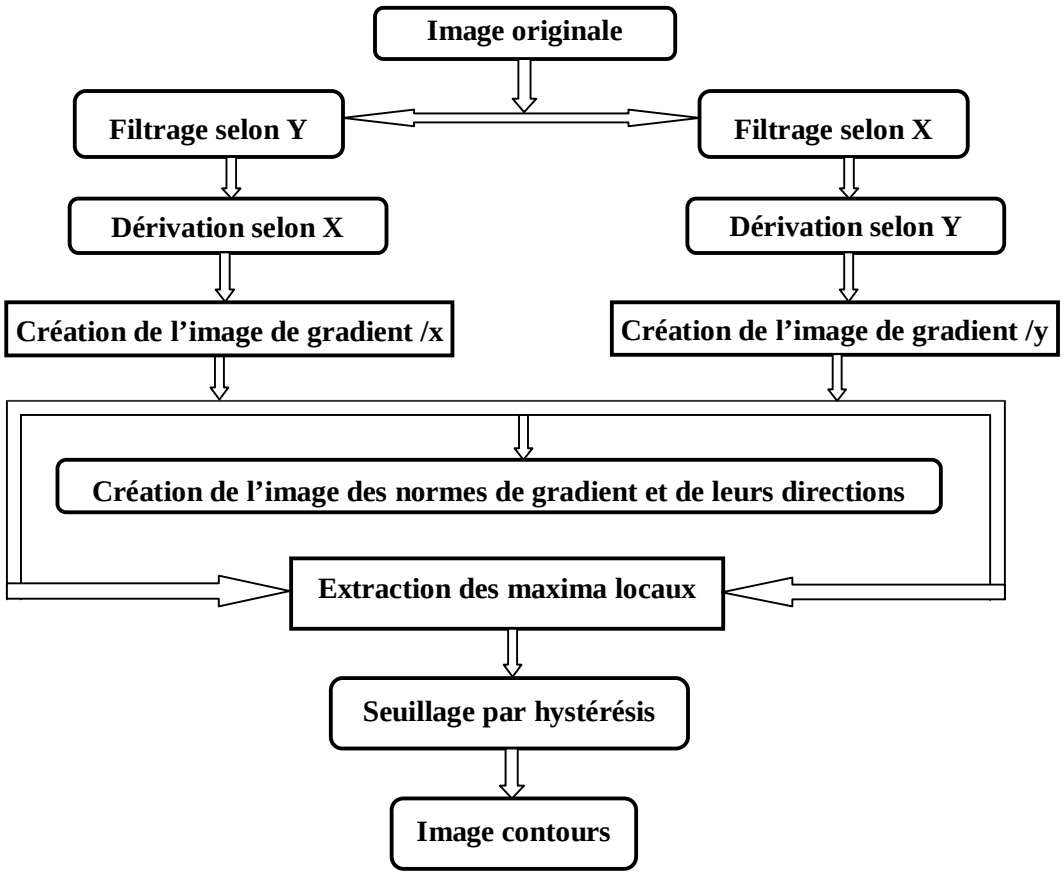


Figure 2-17 : Organigramme du détecteur de CANNY.

. Atténuation des effets du bruit

La partie concernant le filtrage nous a montré que les effets du bruit pouvaient être atténués par convolution de l'image avec un filtre passe bas. Il est possible d'effectuer ce filtrage, puis de dériver l'image pour diminuer l'influence du bruit. Mais il est aussi possible de dériver directement le produit de convolution de l'image et du filtre. Nous prendrons pour exemples l'algorithme développé par Canny qui dérive la convolution d'une image avec une gaussienne. Pour alléger les écritures, nous ne tiendrons pas compte des termes en $\frac{1}{\sqrt{2p}\,s}$ devant les courbes de gauss. La méthode est la suivante [9].

Soit la gaussienne :

$$G(x,y) = \exp\left(-\frac{x^2+y^2}{2s^2}\right) \quad \text{.....(2.20)}$$

La dérivée de l'image I filtrée est :

$$\Delta f = \Delta(G \otimes I) = f_x + f_y \quad \text{.....(2.21)}$$

Où f_x et f_y représentent les dérivées suivant x et y. il vient :

$$\begin{aligned}\Delta f &= (G \otimes I)_x + (G \otimes I)_y \\ &= (G_x \otimes I) + (G_y \otimes I)\end{aligned} \quad \text{.....(2.22)}$$

Les dérivées partielles de la courbe de Gauss sont :

$$G_x(x,y) = -\frac{x}{s^2}\exp\left(-\frac{x^2+y^2}{2s^2}\right) \quad \text{.....(2.23)}$$

$$G_y(x,y) = -\frac{y}{s^2}\exp\left(-\frac{x^2+y^2}{2s^2}\right) \quad \text{.....(2.24)}$$

Les filtres étant séparables, nous pouvons réaliser séparément les convolutions suivant *x* et *y* :

$$G_x(x,y) = G_x(x) \otimes G(y) \quad \text{.....(2.25)}$$

$$G_y(x,y) = G_y(x) \otimes G(y) \quad \text{.....(2.26)}$$

D'où :

$$f_x = G_x(x) \otimes G(y) \otimes I \quad \text{.....(2.27)}$$

$$f_y = G_y(x) \otimes G(y) \otimes I \quad \text{.....(2.28)}$$

La direction et l'amplitude sont alors données par les relations :

$A = (f_x + f_y)^{1/2}$ .....(2.29)

$Dir = \text{ArcTan}\,(f_y / f_x)$ .....(2.30)

On joue sur la sensibilité de l'opérateur au bruit en faisant varier la variable σ de la courbe de Gauss. Notons que plus se paramètre est grand, plus les contours s'adoucissent et donc perdent en finesse dans les détails.

. Amincissement des lignes de contour

Lorsqu'on a effectué le gradient d'une image, il est nécessaire d'isoler les maximums locaux de l'image dérivée pour déterminer les points exacts de contour afin de réduire ceux-ci à une courbe d'un seul pixel d'épaisseur. Une méthode est appelée « méthode de suppression des points non maximum ». Son principe est le suivant :

- Soit A(x,y) l'image composée des amplitudes des gradients (on dit aussi la carte des gradients), et Dir(x,y) la carte des directions.
- Pour chaque point A(x,y), on détermine les points adjacents qui se trouvent dans la direction du gradient, comme illustré figure 2.18. Soit $A(x_1,y_1)$ et $A(x_2,y_2)$ ces points.
- L'algorithme est alors très simple : si A(x,y) est supérieur à la fois à $A(x_1,y_1)$ et à $A(x_2,y_2)$, alors A(x,y) est conservé, sinon A(x,y) est mis à zéro [9].

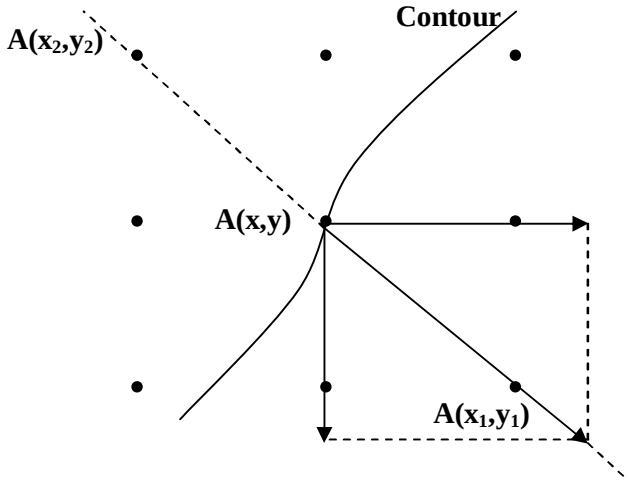


Figure 2-18 : Direction du contour et de Gradient

. Seuillage par Hystérésis

Cette technique de seuillage utilise deux seuils : un seuil haut (noté Sh) et un seuil bas (noté Sb). En dessous du seuil bas, on considère qu'il ne s'agit pas d'un contour. Au-dessus du seuil haut, on estime qu'il y a un contour. Entre les deux seuils, on ne garde comme points de contour que ceux qui sont connexes à au moins un point du contour sûr (au-dessus de Sh) [46].

Deux méthodes existent pour réaliser le seuillage par Hystérésis [46]:

- Ø On fait un seuillage avec le seuil haut. Certains points du contour vont donc disparaître. Ensuite, on ajoute au contour tous les points supérieurs au seuil bas qui sont reliés à un contour déjà retenu.

Ø On fait un seuillage avec le seuil bas. On ne garde ensuite que les contours dont un point au moins est au-dessus du seuil haut.

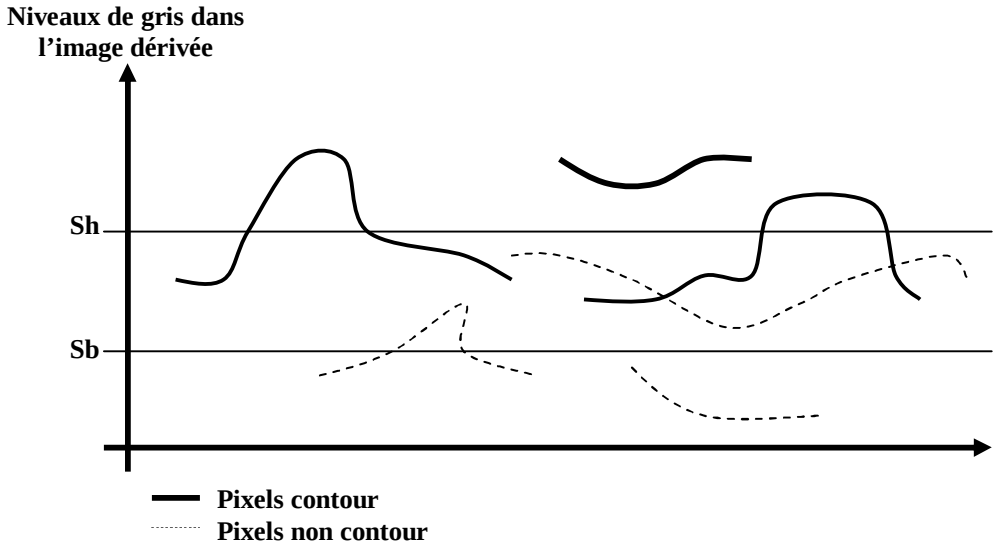


Figure 2-19 : Exemple de seuillage par hystérésis

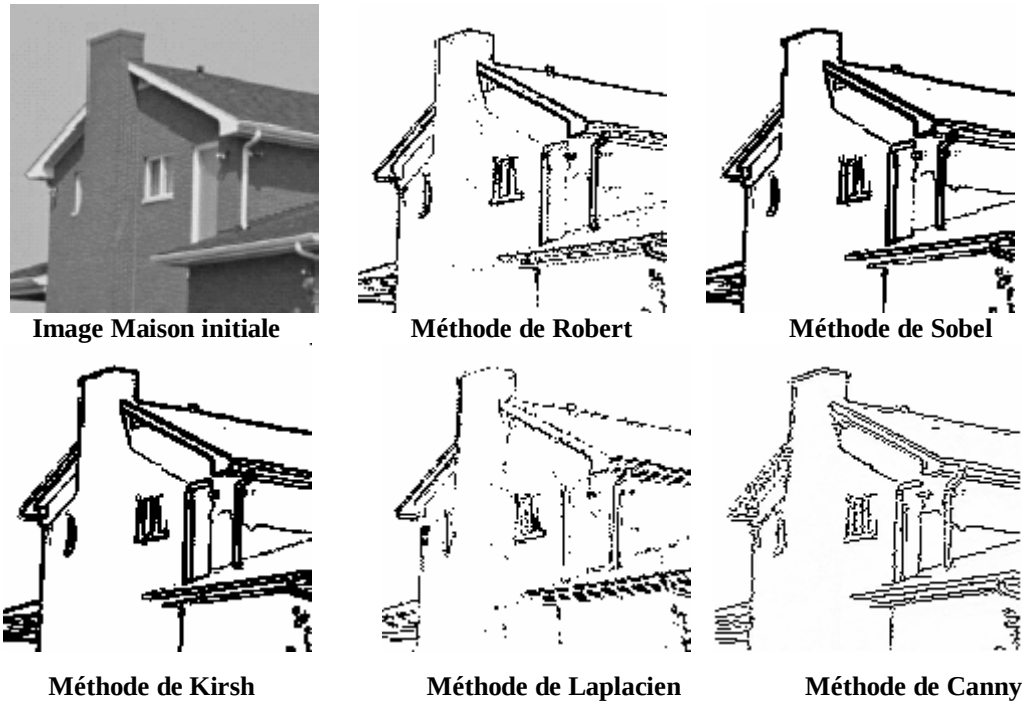


Figure 2-20 : Détection de contours par différentes méthodes

2.4.4. Approche région

La segmentation en réions est un des problèmes les plus difficiles de la vision par ordinateur. Elle consiste à décrire l'image à l'aide de régions pertinentes regroupant tous les pixels respectant une même propriété. De nombreux travaux sont fondés sur une segmentation idéale de l'image, où chaque région correspond à une partie bien précise d'un

objet. Le problème est alors ramené à une mise en correspondance directe entre les régions et les objets de la scène. La segmentation est donc une étape très importante, pratiquement incontournable pour de nombreuses applications.

Les premiers travaux sur la segmentation datent d'une trentaine d'années. Les espoirs étaient grands de réaliser un système de vision performant en s'aidant des techniques de segmentation.

Nous pouvons classer les méthodes de segmentation régions en deux grandes catégories :

- Celles qui exploitent des connaissances sur les objets pour aider la segmentation.
- Celles qui ne les exploitent pas.

Dans le cadre de cette thèse, nous considérons uniquement la segmentation descriptive, c'est à dire que les régions fournies par l'algorithme ne correspondent pas obligatoirement aux objets de la scène, mais plutôt à des ensembles cohérents de pixels fournissant les premiers indices visuels.

Le formalisme mathématique de la segmentation en région est le suivant [18] :

La segmentation consiste à créer une partition de l'image A en sous région R_i, tels que :

1) $\forall$ i ; $R_i \neq \emptyset$ (une région ne doit pas être vide).
2) $\forall$i, j ; i≠j ;$R_i \cap R_j = \emptyset$ (Séparation entre les deux régions).
3) $A = \cup R_i$ (Segmentation complète, chaque pixel appartient a une région et une seule).

Ainsi la région sera un ensemble connexe de points image ayant des propriétés communes (intensité, texture,…) qui les différencient des pixels des régions voisines.

Cette définition conduit à deux remarques importantes :

– Tout d'abord, une segmentation dépend du critère employé. Le choix du critère est donc primordial.
– Ensuite, la décomposition obtenue n'est pas unique. Pour un critère donné, il existe donc plusieurs solutions.

Il existe plusieurs méthodes pour cette approche nous allons citer quelques-unes :

2.4.4.1. Méthode de division/fusion

Cette méthode fait partie des méthodes fondamentale de la segmentation en régions, elle se base sur deux phases successives, qui sont la division de l'image en blocs, et la fusion de ces blocs suivant certains critères d'homogénéité.

a) La division

L'image à traiter doit être de dimension N * N ($N = 2^m$). C'est le bloc initial B0. Le principe consiste à diviser récursivement tout bloc non homogène selon un prédicat défini. La division d'un bloc Bi de dimension $2^{N-k} * 2^{N-k}$ donne naissance à quatre sous-blocs B0[k + 1], B1[k + 1], B2[k + 1], B3[k + 1], de dimension ($2^{N-k-1} * 2^{N-k-1}$). Un exemple montre ce découpage sur une image binaire (figure2.21).

Après chaque nouvelle division, les attributs de chaque bloc nouvellement créé sont recalculés pour être à nouveau soumis au prédicat d'homogénéité. Présentée de cette façon, la technique du *quadtree* suggère la structure d'une pyramide construite à l'envers (l'apex en bas et la base en haut). En effet, le partitionnement initial est représenté par une unique région (l'image entière) et le nombre de régions (ou blocs) augmente lors de la construction de la structure.

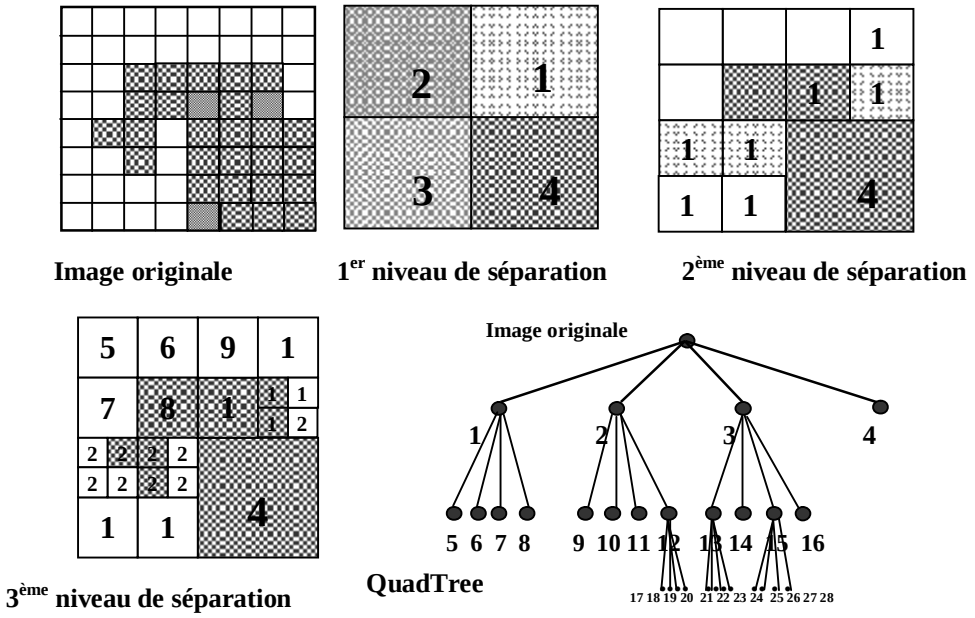


Figure 2-21 : Utilisation de l'arbre Quartenaire pour la division

La résolution finale de la pyramide *quadtree* est fortement liée au seuil fixant la limite du critère d'homogénéité. Elle est également liée à la taille minimale des blocs en dessous de laquelle on ne veut plus provoquer de division quel que soit le résultat de l'évaluation du prédicat d'homogénéité. Celui-ci porte généralement sur la variance des niveaux de gris des pixels qui composent le sommet : un bloc n'est plus décomposé lorsque la variance de ses pixels devient inférieure au seuil défini.

A cause des divisions en quatre des régions, cette méthode est plutôt adaptée à des images carrées ayant un nombre de lignes et de colonnes égal à une puissance de deux, et

dans lesquelles les régions sont de forme rectangulaire. D'autre part, cette méthode a tendance à faire apparaître des effets de blocs [43].

b) La fusion

L'idée consiste à exploiter une partition initiale de l'image constituée de petites régions, puis ces régions sont fusionnées successivement jusqu'à ce que le critère de fusion ne soit plus vérifié. Plusieurs critères de regroupement ont été proposés mettant en jeux certaines propriétés, nous citons [37] :

- **Propriétés statistiques :**

  Telle que la moyenne ou la variance des niveaux de gris des régions, le gradient moyen des frontières de régions, le contraste maximum des régions,…

- **Propriétés géométriques ou morphologiques :**

  Telle que l'élongation ou la compacité des régions. Deux régions sont regroupées si par exemple un facteur de forme est conservé ou amélioré après leur fusion.

L'inconvénient majeur de la méthode de division-fusion provient du fait qu'elle ne tienne pas compte des relations spatiales des pixels. La méthode qui va suivre utilise aussi bien les relations spatiales entre les points que leurs propriétés.

2.4.4.2. Méthode de croissance de régions

L'approche par croissance de régions est l'agrégation de points, en utilisant des critères qui tiennent compte à la fois du contexte local et des informations sur la région formée ; pour ces raisons, elle a donc été retenue. Cette approche permet une meilleure efficacité quant à la rapidité et la taille mémoire nécessaire à la structure des données relatives aux régions et à leurs attributs [46].

La méthode de croissance de régions est une méthode ascendante qui, partant de la représentation de l'image comme un ensemble de pixels, les regroupe selon un double critère d'homogénéité et d'adjacence. Cette méthode est conduite par l'utilisation d'un mode de contrôle appelé prédicat, qui permet d'identifier une contrainte que doivent satisfaire les régions ; par exemple le prédicat pour une région peut être [46]:

- La variance des niveaux de gris de l'image associés aux points de la région qui doit être inférieur à un seuil préfixé.
- La différence entre le niveau de gris de la région en cours de formation et le niveau de gris du pixel candidat.

L’étape initiale consiste à sélectionner les germes des régions qui correspondent généralement à un pixel. Puis, les régions sont construites en y ajoutant successivement les pixels qui leur sont connexes et qui vérifient un critère de similarité colorimétrique [43].

La plupart des algorithmes de croissance de régions sont régis par un parcours de l’image selon le balayage de gauche à droite et de haut en bas.

L’avantage de la croissance de régions est de préserver la forme de chaque région de l’image.

Cependant, une mauvaise sélection des germes ou un choix du critère de similarité mal adapté peuvent entraîner des phénomènes de sous-segmentation ou de sur-segmentation.

Figure 2-22 : Croissance des régions sur l’image Maison

2.4.5. La segmentation coopérative

2.4.5.1. Les approches classiques

L’intérêt de faire coopérer plusieurs approches est de réussir à définir automatiquement des ensembles de contraintes pour guider les processus de segmentation. La stratégie de résolution peut donc être séquentielle, c’est-à-dire exploiter successivement le résultat d’une approche pour en guider une autre, ou bien itérative, c’est-à-dire fondée sur un principe réciproque de définition de contraintes [35].

La coopération entre un processus de segmentation en régions et un détecteur de contours a suscité un regain d’intérêt ces dernières années. Les premiers travaux importants dans ce domaine sont pourtant relativement vieux, avec par exemple l’approche de Brice et Fennema en 1970 [1]. Bien qu’ils ne présentent pas leur méthode de segmentation comme une coopération, ils exploitent l’information du gradient à la frontière ainsi que la longueur de celle-ci pour déterminer la fusion des régions à partir de deux heuristiques qu’ils appellent "phagocyte" et "weakness".

Melkemi et Chassery proposent également une coopération avec détection préliminaire des contours, suivie d'une technique de croissance de régions fondée sur les polygones de Voronoi. La "segmentation coopérative" de Bonnin est fondée sur une comparaison des résultats d'un processus de fusion et d'un détecteur de contours, avec plusieurs étapes de raffinements [15]. Même si les contours sont remis en question, ceux-ci sont calculés également au début de la coopération, sans l'aide des informations sur les régions. De manière générale, la coopération se fait essentiellement des contours vers les régions (coopération contour-région). Ceci est probablement dû à l'incapacité des détecteurs de contours à exploiter les informations globales véhiculées par les régions.

Une autre coopération a été proposée par Pavlidis et Liow [7]. Après une première étape de pré-segmentation par l'algorithme de "split and merge" de Pavlidis, une procédure de fusion est appliquée en tenant compte des gradients à la frontière. La segmentation régions n'est ensuite plus remise en cause, mais les frontières entre celles-ci sont approximatives. Une technique de contours déformables permet alors de repositionner correctement les contours, en exploitant les informations de direction du gradient et de courbure. Cette approche est intéressante car il est très rare que les algorithmes de segmentation utilisent des informations sur la forme pour localiser la frontière des régions.

Chu et Aggarwal [14] suggèrent l'intégration a posteriori de cartes de segmentation calculées en parallèle. Ces cartes de contours sont issues de l'application de détecteurs de contours et de l'extraction des contours après une segmentation par approche région. Ils utilisent, de plus, plusieurs sources de la même scène. L'intégration finale cherche à générer un consensus entre les différentes segmentations sur la base d'une pondération, réglable par l'utilisateur, apportée à chacune des cartes [44].

Cho et Meer [27] proposent une approche basée sur le consensus entre plusieurs segmentations. Les résultats des diverses segmentations permettent la construction d'un graphe reflétant pour chaque paire de pixels « la probabilité de cooccurrence » autrement dit, la probabilité d'appartenance à la même région. Le consensus est obtenu en traitant ce graphe de façon classique afin de regrouper les pixels ayant une forte probabilité d'appartenir à la même région.

La coopération entre plusieurs techniques de segmentation a donc souvent consisté en une succession d'étapes de segmentation régions ou contours. La coopération se situe surtout au niveau de la confrontation des résultats, mais pas au niveau de la prise de

décision de chaque processus, notamment en ce qui concerne la validation d'un contour [29][30].

Les problèmes majeurs sont, selon nous, l'intégration des informations complémentaires et le contrôle de la coopération qui a pour charge la gestion et l'enchaînement des étapes.

2.4.5.2. La système expert de Nazif et Levine

Nazif et Levine ont été les premiers à présenter un système expert pour effectuer la segmentation d'images [2]. Cette approche s'inscrit un peu à contre courant de la tendance de l'époque où l'explicitation des connaissances était réservée à l'étape suivante d'interprétation. Néanmoins, nous allons nous intéresser de près à leurs travaux car le système qu'ils proposent constitue une coopération région-contour et contour-région très originale.

En premier lieu, ils obtiennent la carte des contours par un seuillage simple des gradients, ainsi qu'une carte des premières régions homogènes avec de nombreux attributs dans la liste des régions adjacentes (peu de précisions sont apportées sur ces étapes préliminaires). A ces deux cartes s'ajoutent des informations sur les zones d'intérêt de l'image, correspondant à des ensembles de régions de même type, et à des ensembles de contours. Ensuite, une zone d'intérêt est choisie, où chaque élément va être étudié.

En ce qui concerne les régions, il existe des règles dédiées dont la conclusion est la fusion ou la division de deux régions et dont la prémisse est constituée d'attributs sur les régions et sur les contours. La fusion de deux régions dépend, selon les règles, de la grosseur des deux régions, de la différence de moyenne dans chaque couleur, de la variance des deux régions dans chaque couleur, du nombre de pixels à la frontière, de la présence de pixel-contours à la frontière ainsi que de la forme de l'histogramme.

Les règles dédiées aux contours consistent à prolonger, à joindre ou à supprimer ceux-ci. Les prémisses prennent en compte la présence d'autres contours dans le voisinage, leur alignement, le gradient moyen sur le contour, la longueur de celui-ci ainsi que l'unicité des régions adjacentes.

La coopération entre les régions et les contours apparaît dans les deux sens, pour diviser correctement les régions traversées par un contour, mais aussi pour aider le prolongement des contours.

De façon générale, le système expert qu'ils présentent est difficile à juger, car nous ne sommes pas capables d'évaluer l'expertise contenue dans chaque règle, ce qui constitue un paradoxe pour un système expert. Il aurait sans doute fallu qu'ils expliquent la méthodologie de création des règles, ainsi que les moyens mis en œuvre pour les évaluer et les modifier. Nous retiendrons que ce système expert est une tentative d'explicitation des connaissances en segmentation d'images, avec émergence des informations dans les règles de production.

2.4.5.3. Le système de Bellet

Bellet a développé un système coopératif pour la segmentation de bas niveau des images [29]. Ce système est multi-processus, puisqu'une image est segmentée par une multitude de processus indépendants, qui sont basés sur des modèles de croissance de région ou de suivi de contour. Chaque processus est initialisé à un endroit précis de l'image, avec un germe, et il fait croître sa primitive à partir de cet endroit. La croissance (région ou contour) s'effectue par une évaluation multi-critéres de pixels avoisinants et sélection des meilleurs candidats.

Ce système est basé sur une nouvelle forme de coopération entre différents processus de segmentation de natures différentes. Lorsqu'un processus manque d'informations pour mener à bien les décisions qu'il doit prendre, il crée de nouveaux processus, qualifiés de *fils* pour segmenter de nouvelles primitives et ainsi, récolter plus d'informations sur l'environnement local. Un processus peut également s'appuyer sur les informations récoltées par ses voisins pour mener à bien sa tâche. C'est sur ce modèle de coopération que s'effectue toute la segmentation de l'image.

Le système est généraliste et ne bénéficie d'aucune connaissance sur les applications et les images qu'il segmente. Il a été utilisé pour différents types d'images.

2.4.5.4. L'approche multi-agents

L'approche multi-agents, pour effectuer une segmentation coopérative, semble naturelle (chaque agent est un petit système expert autonome lié aux autres agents).

Baujard et Garbay proposent un système de vision complet pour l'identification de cellules biologiques, fondé sur la coopération entre le détecteur de contours de Deriche et la technique de classification appliquée aux régions de Fisher, précédée d'un filtrage de l'image [13]. Un agent est dédié à chacun des processus.

Après détection des régions et des contours, les résultats sont analysés et comparés dans un agent spécifique. Lorsqu'il y a compatibilité entre la frontière des régions et les contours, le résultat est validé et envoyé à un agent qui tente de classer la région à partir de la forme de celle-ci. Si la compatibilité n'est pas vérifiée, les régions sont découpées ou fusionnées.

Cette approche est particulièrement intéressante car elle pose clairement le problème de la gestion de la coopération. Il ne s'agit pas ici de juger cette méthode sur la qualité des résultats mais sur la manière d'aborder le problème. Il semble en effet qu'il y ait de grandes difficultés à faire coopérer deux méthodes qui ont peu de points communs et qui ne sont pas spécialement préparées pour cette coopération. Si le principe du cumul des informations est important, le principe de complémentarité des informations doit également être pris en compte pour que l'efficacité soit réelle. Ainsi, à partir d'un détecteur tel que celui de Canny (en fait à partir de la plupart des détecteurs de contours), il est difficile de trouver un processus de détection des régions qui soit complémentaire.

2.5. Conclusion

L'analyse d'images constitue un outil puissant pour l'extraction des données pertinentes d'une image. Une approche méthodique sous forme de trois étapes (acquisition, prétraitement, segmentation) est appliquée. Pour chaque étape de l'analyse, il existe une multitude de méthode qu'on peut utilisé séparément ou en coopération.

Les différents traitement dans l'analyse d'images peuvent être implémenté soit en utilisant l'approche classique (séquentielle), soit en utilisant une approche intelligente : système multi-agents.

De manière générale, l'approche multi-agents et les algorithmes parallèles méritent une plus grande attention car la façon d'aborder les problèmes incite à une réflexion profonde sur la gestion des informations et l'enchaînement des étapes.

Cette dernière approche fera l'objet du chapitre suivant.

CHAPITRE 3
CONCEPT DE BASE SUR LES SYSTEMES MULTI-AGENTS

3.1. Introduction

Afin de résoudre certains problèmes complexes, l'intelligence artificielle (IA) a été introduite pour essayer de reproduire des comportements intelligents réservés jusqu'alors aux êtres humains.

L'architecture des systèmes d'IA classiques est constituée d'un seul composant pouvant résoudre un certain nombre de problème en s'appuyant sur une seule base de connaissances.

Dans une approche basée sur l'Intelligence Artificielle Distribuée (IAD), l'architecture est constituée de plusieurs composants d'IA indépendants coopérant pour résoudre un problème.

Manipulant des concepts de haut niveau tel que la rationalité et l'intentionnalité, les systèmes multi-agents (SMA) sont apparus comme résultat de l'évolution de l'IAD.

Ce chapitre commence par une brève définition sur l'IAD suivi des concepts de base des SMA, leur organisation en société, leur contrôle et la communication entre eux. En fin, différentes approches ainsi que quelques exemples existants de ces systèmes sont présentés.

3.2. L'intelligence artificielle distribuée

L'intelligence Artificielle Distribuée (IAD) consiste à distribuer l'expertise au sein d'une société d'entités appelée agents dont le contrôle et les données sont distribuées. Chaque agent a ses propres compétences mais il a besoin d'interagir avec les autres pour résoudre des problèmes qui dépendent de son domaine d'expertise. L'objectif de l'IAD est donc de trouver une solution à des problèmes globaux grâce à un ensemble d'entités distribuées travaillant en collaboration (21].

L'IAD diffère du domaine des traitements distribués**.** En effet, ce dernier concerne la coordination d'un ensemble de nœuds, n'ayant aucune autonomie et qui accomplissent un ensemble souvent indépendant. Au contraire, l'IAD signifie que les entités sont autonomes et qu'elles doivent coopérer pour arriver à la résolution du problème. Les tâches accomplies par les différentes entités peuvent être dépendantes. De même, une autre différence réside dans le fait que l'IAD concerne la distribution aussi des données que du contrôle parmi les différentes entités [19][21].

3.3. Système Multi-Agents

Le concept des « Systèmes Multi-Agents (SMA) » combine la notion de « système » avec les notions de « multi » et d'« agent ». La sémantique du préfix « multi » est simple : elle indique que le système est typiquement composé de deux ou plusieurs agents. Pour le mot « agent », nous le définissons comme un processus, i.e. quelque chose qui agit, doté d'une certaine autonomie, qui n'est pas complètement contrôlé par un autre processus [32].

En générale, un système multi-agents est un système qui se compose d'un ensemble d'entités spécialisées coopérantes pour le compte d'une application globale (fonction globale où but commun). [25].

3.4. Apport des systèmes Multi-Agents

Il existe de nombreuses raisons amenant à choisir une solution basée sur l'IAD pour résoudre des problèmes nécessitant l'apport d'expertise. L'IAD intervient ainsi dans les mêmes domaines que l'IA classique en lui apportant de nombreux avantages.

L'approche multi-agents est justifiée par les points suivants : [21] [31] [46]

- Elle s'adapte bien à la réalité dans la mesure ou de nombreux problèmes sont de nature distribuée. Des cas typiques de problèmes qui entrent dans cette catégorie sont :
  - Distribution géographique ou spatiale (gestion du trafic aérien ou tâches perceptuelles en vision par exemples).
  - Distribution fonctionnelle en plusieurs sous tâches (environnement industriel multi-robots).
  - Distribution des connaissances et des méthodes de traitement requises par l'application.
- La coopération : entre plusieurs agents, dont les expertises se chevauchent mais dont les points de vue sont différents.
- Elle permet de résoudre des problèmes de taille et de complexité importante, telles qu'il n'est pas réaliste d'essayer de les résoudre à l'aide d'un seul agent.
- L'intégration de plusieurs agents : dont les expertises incomplètes ou peu fiables peut amener à une expertise plus sure et plus robuste.
- La modularité : Le fait de décomposer un système en sous système permet un développement beaucoup plus facile. Pour les problèmes de grande taille, cela permet d'avoir une vue simplifiée. Différentes équipes peuvent participer au développement d'un sous système et accélérer le temps de réalisation. Une fois le système crée, il est

ensuite aise de le maintenir, le tester et l'enrichir. Chaque expertise faisant partie d'un sous système peut être traitée séparément pour arriver à une construction modulaire. Cette caractéristique se trouve en conception ou tout objet conçu de façon modulaire est reconnu pour être un objet bien conçu.

- L'efficacité : La répartition des tâches sur différentes ressources implique un accroissement de la capacité de calcul. Les architectures multiprocesseurs et les machines parallèles peuvent être le support matériel de système distribué dans le cadre d'application nécessitant des temps de réponses rapides. Chaque processeur est associé à un traitement particulier ce qui permet d'augmenter sensiblement la capacité de calcul.
- La fiabilité : Les systèmes distribués, construits à partir de plusieurs modules, sont plus tolérants aux pannes puisque l'arrêt de l'activité d'un module n'entraîne pas l'arrêt complet du système. Un autre module peut par exemple être délégué pour remplacer provisoirement une défaillance.
- La Réutilisation : un agent peut être réutilisé pour implanter une partie d'un autre système. L'expertise qu'il englobe ne demande pas à être de nouveau acquise et réimplantée.
- Le Coût : L'intérêt d'une solution par l'IAD relève aussi de ce que 10 ordinateurs de puissance P coûtent moins cher qu'un ordinateur de puissance 10*P. Cette constatation, purement financière constitue l'une des principales raisons du choix de systèmes distribués par les industriels.

3.5. Agents

Selon Ferber [17] On appelle agent une entité physique ou abstraite :

- qui est capable d'agir sur elle-même et sur son environnement,
- qui peut communiquer directement avec d'autres agents,
- qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voir de survie),
- qui possède des ressources propres,
- qui est capable de percevoir son environnement,
- qui ne dispose que d'une représentation partielle de cet environnement,
- qui possède des compétences et offre des services,
- Dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont il dispose, et en fonction de sa perception, de ses interactions avec les autres agents.

3.5.1. Point de vue interne : ce qu'est l'agent

Il est particulièrement habituel de distinguer dans un agent des étapes de *perception*, de *raisonnement* et *d'action*. Le point de vue interne correspond à ces trois étapes, avec une emphase particulière sur celle de raisonnement, qui est l'étape en laquelle se passe la décision d'action **(voir figure 3.1).** Ce point de vue englobe l'ensemble des mécanismes produisant le comportement de l'agent. A ce niveau, une distinction classique dans le domaine des systèmes multi-agents (SMA) se fait entre les agents [45].

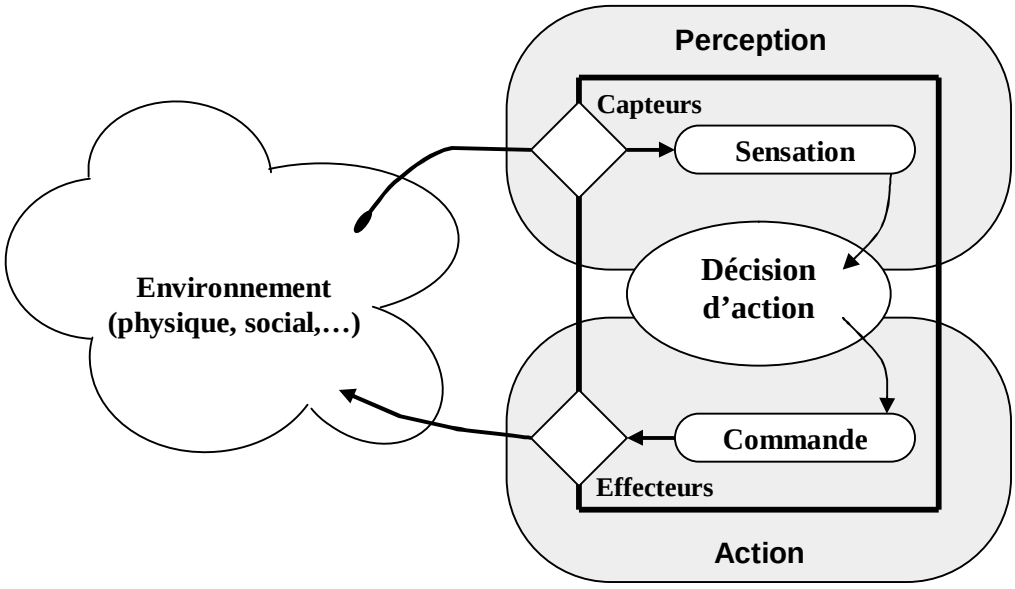


Figure 3-1 : Structure classique d'un agent logiciel

Cette distinction classique se fait souvent en termes de complexité des mécanismes produisant le comportement de l'agent, des agents par exemple peuvent faire appel à des représentations de l'environnement, des mécanismes d'apprentissage ou de planification, de communication directe avec d'autres agents, etc., alors que d'autres se contentent de réagir de façon réflexe à son environnement.

Selon le principe de récursion [23], un SMA peut être considéré comme un seul agent à un niveau supérieur d'abstraction. C'est-à-dire que, d'un point de vue interne, un agent peut être un SMA. N'hésitant pas par exemple à attribuer des buts à un SMA. Une telle attribution est beaucoup plus compréhensible si l'on considère que les buts sont rattachés à l'agent représentant le système plutôt qu'au système lui-même.

3.5.2. Point de vue externe : ce que fait l'agent

Il est possible de dire que ce qui est important dans un agent, c'est ce qu'il fait, et non les mécanismes qui l'amènent à agir de telle ou telle façon. Une position proche dirait que seul le comportement peut être un objet d'étude, car seul le comportement est observable. Ce dernier point est à la base du mouvement behaviouriste, pour lequel la psychologie doit étudier les comportements observables plutôt que les processus mentaux. Ces points de vue sont aussi partagés, dans une certaine mesure, par des chercheurs en IA et robotique comme Rodney Brooks. Une des différences évidente entre la psychologie et l'IA est le support de leurs études : humains pour la psychologie, systèmes artificiels pour l'IA. Toutefois, on peut considérer une problématique commune dans la mesure où ces deux sciences s'intéressent au comportement de leur objet d'étude [45].

D'un point de vue fonctionnel, on pourrait effectivement considérer comme seul point pertinent le comportement des agents, et, dans une certaine mesure, cette attitude est effectivement suffisante. Même dans les situations où un agent chercherait à anticiper, simuler ou tout simplement modéliser le comportement d'un autre agent, les modèles qu'il emploierait n'ont pas à être congruents avec ceux effectivement employés par l'agent à modéliser. Après tout, certains systèmes cherchent bien à reproduire un comportement humain sans que les mécanismes de génération de comportement de l'humain ne soient connus. Ainsi, dans le cadre des modèles mis en place dans le domaine de la psychologie, tout se passe comme si ‘le système logique’ existait. Il est une construction intellectuelle du psychologue qui l'aide à comprendre le sujet et à lui parler en ‘entrant dans sa logique’. [45].

Cependant, toute analyse fine du comportement ne peut que difficilement se passer de répondre à la question de l'ontologie de l'agent. En effet, l'aspect comportemental est la conséquence des mécanismes internes de l'agent. La prise en compte des aspects internes de l'agent permet d'affiner la connaissance (et l'exploitation de cette connaissance) issue de l'analyse du comportement perçu. Ainsi, par exemple, pour garantir la conformité des interprétations construites par la machine aux attentes des utilisateurs, le fonctionnement du système mis en œuvre doit présenter une certaine analogie avec celui de la cognition humaine. [24].

Le risque majeur de limiter un agent à son comportement est l'assimilation : « mêmes comportement → mêmes phénomènes en soi ». Comme le dit Searle (cité par Christian Brassac et Sylvie Pesty [20]) : « *si ce principe était correct, il nous faudrait tous conclure*

que les postes de radio sont conscients parce qu'ils manifestent un comportement verbal intelligent ». Il est donc important de ne pas se limiter à l'étude du comportement lorsqu'il est possible d'avoir accès aux mécanismes qui produisent ce comportement.

3.5.3. Agents réactifs et agents cognitifs

La granularité des agents impliqués dans une application varie selon deux écoles coexistantes aujourd'hui : l'école cognitive et l'école réactive. Suivant le type d'agent utilisé, on parlera de systèmes cognitifs ou de systèmes réactifs. [11]

3.5.3.1. Agents réactifs

Les défenseurs de cette approche partent du principe suivant : dans un système multi-agents, il n'est pas nécessaire que chaque agent soit individuellement intelligent pour parvenir à un comportement global intelligent. En effet, des mécanismes simples de réactions aux événements peuvent faire émerger des comportements correspondant aux objectifs poursuivis. Cette approche propose la coopération d'agents de faible granularité (fine grain) mais beaucoup plus nombreux [11].

Les agents réactifs sont de plus bas niveau, ils ne disposent que d'un protocole et d'un langage de communication réduits, leurs capacités répondent uniquement à la loi stimulus/action [11].

L'exemple le plus manifeste d'organisation émergente est la fourmilière [31] [46], alors que toutes les fourmis se situent sur un plan d'égalité et qu'aucune d'entre elles ne possède de pouvoir d'autorité stricte sur les autres, les actions des fourmis se coordonnent de manière que la colonie survive et fasse donc face à des problèmes tels que ceux poses par la recherche de la nourriture, les soins aux œufs, etc.

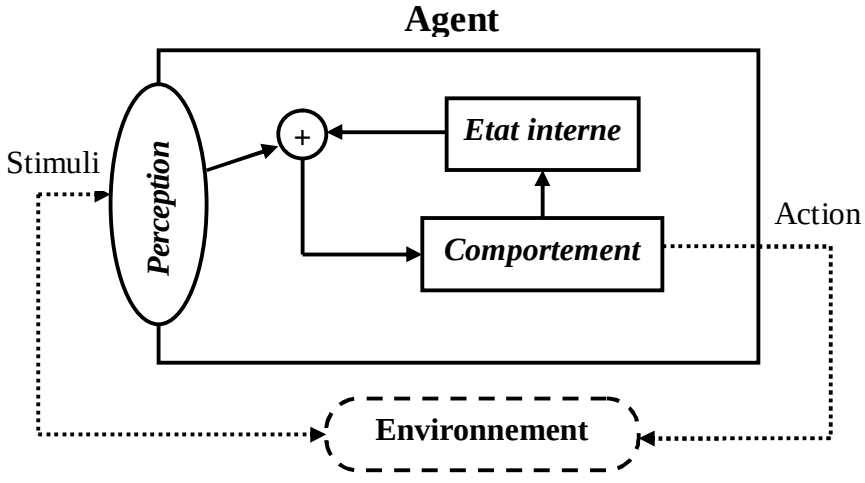


Figure 3-2 : Structure de fonctionnement d'un agent réactif.

3.5.3.2. Agents cognitifs

Les systèmes d'agents cognitifs sont fondés sur la coopération d'agents capables _a eux seuls d'effectuer des opérations complexes. Un système cognitif comprend un petit nombre d'agents qui disposent d'une capacité de raisonnement sur une base de connaissances, d'une aptitude à traiter des informations diverses liées au domaine d'application, et d'informations relatives à la gestion des interactions avec les autres agents et l'environnement. Chaque agent est assimilable, suivant le niveau de ses capacités, à un système expert plus ou moins sophistiqué, on parle d'agent de forte granularité (coarse grain) [11].

3.5.3.3. Comparaison entre agent cognitif et agent réactif

Tableau 3-1 : Différence entre agent cognitif et agent réactif

Agent cognitif	**Agent réactif**
Représentation explicite de l'environnement	Pas de représentation explicite
Peut tenir compte de son passé	Pas de mémoire sur son historique
Agent complexe	Fonctionnement stimulus/réponse
Petit nombre d'agents de forte granularité	Grand nombre d'agents de faible granularité

3.5.4. Caractéristique d'un agent

L'avancement des travaux en IAD et en SMA a conduit les chercheurs à définir non seulement la notion d'agent mais aussi quelques-unes de ses caractéristiques [11] :

- **Intentionnalité :** un agent intentionnel est un agent guidé par ses buts. Une intention est la déclaration explicite des buts et des moyens d'y parvenir. Elle exprime donc la volonté d'un agent d'atteindre un but ou d'effectuer une action [11].
- **Rationalité** : Un agent rationnel est un agent qui sui le principe suivant (dit principe de rationalité) :

  « Si un agent sait qu'une de ses actions lui permet d'atteindre un de ses buts, il la sélectionne ».

  Les agents rationnels disposent de critères d'évaluation de leurs actions, et sélectionnent selon ces critères les meilleures actions qui leur permettent d'atteindre

le but. De tels agents sont capables de justifier leurs décisions. La notion de rationalité se rapporte au comportement cognitif de l'agent. Ce terme qualifie l'utilisation efficace des ressources par l'agent. [11]

- **Engagement :** la notion d'engagement est l'une des qualités essentielles des agents coopératifs. Un agent coopératif planifie ses actions par coordination et négociation avec les autres agents. En construisant un plan pour atteindre un but, l'agent se donne les moyens d'y parvenir et donc s'engage à accomplir les actions qui satisfont ce but ; l'agent croit qu'il est en mesure d'exécuter tout le plan qu'il a élaboré, ce qui le conduit (ainsi que les autres agents) à agir en conséquence. [11]
- **Adaptative :** un agent adaptatif est un agent capable de contrôler ses aptitudes (communicationnelles, comportementales, etc.) celon l'agent qu'il interagit. Un agent adaptatif est un agent de haut niveau de flexibilité. [11]
- **Intelligence :** c'est l'ensemble des fonctions mentales ayant pour objet la connaissance conceptuelle et rationnelle. Un agent intelligent est un agent cognitif, rationnel, intentionnel et adaptatif. Cette classe d'agent tend à stimuler les aptitudes (comportements) humaines. [46] [11]
- **Autonomie :** Ce point est le plus communément invoqué dans les définitions d'agents, mais aussi le plus discuté. Disons que, globalement, l'agent doit pouvoir prendre des initiatives et agir sans l'intervention de l'utilisateur final [45].

3.5.5. Architecture d'agent

L'architecture interne d'un agent est constituée comme suite :

a. Le savoir – faire

Le savoir-faire est une interface permettant la déclaration de connaissances et de compétences de l'agent, il permet la sélection des agents à solliciter pour une tâche donnée. Il n'est pas nécessaire mais il est très utile pour améliorer les performances du système, quel que soit le mode de coopération utilisé. [11]

b. Croyance (connaissance de soi et des autres)

Dans un univers multi-agents, chaque agent possède des connaissances sur lui-même et sur les autres. Ces connaissances ne sont pas nécessairement objectives, on parle alors de croyances d'un agent. Les logiques des connaissances et des croyances s'intéressent à la formalisation de telles connaissances considérées comme

incertaines. Cette formalisation est à la base de la conception de tout système multi-agents puisqu'elle détermine en grande partie le comportement intelligent des agents.

Les croyences de l'agent lui permettent [31] :

- De prédire le comportement de ses accointances et de connaître leurs activités. Ceci aide l'agent à planifier et à coordonner ses actions avec les autres.
- De raisonner sur les intentions des autres et par conséquent savoir quoi communiquer aux autres.
- D'évaluer les informations qu'il reçoit de ses accointances.
- De réduire la communication, car seules les informations utiles sont communiquées.
- De savoir qui fait quoi, et donc à qui s'adresser pour avoir l'information.

c. Contrôle

On peut définir le contrôle dans un agent comme l'exécution d'une action à partir de son état de résolution et de l'état de l'environnement [12].

d. Expertise du domaine (compétences)

Cette partie concerne le domaine de spécialisation de l'agent. L'expertise de l'agent lui confère sa compétence pour un domaine d'application donné. C'est la connaissance sur la résolution de problème. Pour un système expert utilisant le formalisme de règle, par exemple, cette connaissance correspond à sa base de règles [11] [31].

e. Communication

La communication permet aux agents d'échanger des informations. L'agent doit posséder un protocole de communication lui permettant d'interagir avec les autres agents pour une bonne coopération et une bonne coordination d'actions [31].

3.5.6. Fonctionnement

Dans ce paragraphe, nous présentons notre vision de l'architecture fonctionnelle d'un agent cognitif et nous détaillons son fonctionnement. Cette architecture est illustrée par la figure 3.3. Dans cette figure, les ellipses représentent les processus mis en œuvre lors du fonctionnement de l'agent [11].

Les agents sont immergées dans un environnement dans lequel et avec lequel ils interagissent. D'où leur structure autour de trois fonctions principales : percevoir, décider et

agir. Parmi les sous-fonctions importantes d'un agent, on peut citer : la détection de conflits, la révision des croyances, la coopération (négociation, coordination), l'apprentissage, etc. Toutes les fonctionnalités ne sont pas représentées dans la figure [11].

Un agent a la possibilité d'acquérir des connaissances sur l'environnement externe (perception). Il a aussi des capacités d'interaction avec les autres agents (communication, négociation). En fonction des connaissances et croyances dont il dispose et des buts qu'il se fixe suite à une perception ou à une interaction avec le monde extérieur, l'agent doit élaborer un plan d'action. Pour cela, il doit décider quel serait le but à retenir et à satisfaire en premier, ensuite planifier en fonction de ce but et passer à l'exécution. Ces deux derniers processus doivent être alternés du fait du caractère dynamique des environnements multi-agents. [11]

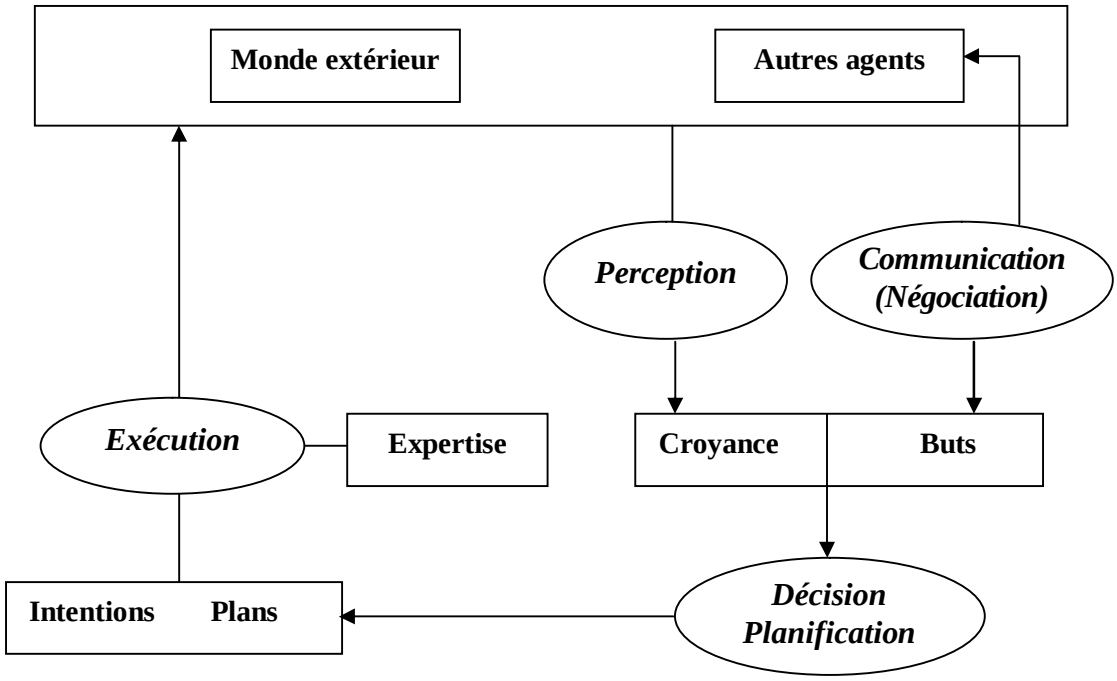


Figure 3-3 : Structure de fonctionnement d'un agent cognitif.

3.5.6.1. Perception

Les connaissances d'un agent ont plusieurs origines :

- le savoir initial de l'agent ;
- la perception de soi (perception proprioceptive) et du monde (perception extéroceptive) ;
- la communication avec les autres agents.

Généralement, les informations issues de la perception et du savoir initial de l'agent sont considérées comme des connaissances certaines puisqu'elles n'ont subi aucune mise à jour, alors que les connaissances provenant des autres agents sont considérées incertaines puisqu'elles évoluent sans que l'agent en soit informé. [11]

3.5.6.2. Prise de décision

Durant son exécution, un agent se fixe un certain nombre de buts, suite à ses observations et à ses interactions avec le monde (perception, communication, négociation). Il se trouve donc confronté au problème de la sélection du but à satisfaire en premier et pour chaque but, de l'action qui permet de l'atteindre. [11]

Face à de telles situations, l'agent analyse les différentes alternatives en termes d'utilité (quel avantage l'agent pourrait en retirer) et d'incertitude (quelle chance a l'action d'être effectuée en fournissant le résultat attendu). [11]

3.5.6.3. Planification

La planification dans les systèmes multi-agents est une planification distribuée : il n'existe pas de plan global et chaque agent construit son propre plan en coordonnant avec les autres (cas d'agents coopératifs). Certains systèmes d'IAD présentent une planification centralisée, un organe central se chargera de la gestion des conflits et de l'élaboration d'un plan global. [11]

3.6. Société d'agents

Un système comportant un agent unique est un cas pour le moins trivial dans le domaine des systèmes multi-agents [45]. Comme le fait remarquer Christof Baeijs dans sa thèse [28], cette organisation à un membre peut se justifier d'un point de vue théorique, grâce au principe de récursion, comme étant la représentation d'un système multi-agents. Ce cas limite mis à part, les systèmes multi-agents se composent de plusieurs agents en interaction.

Nous présentons dans cette section des composantes d'un SMA intimement liées à la création d'un collectif d'agents.

3.6.1. Organisation sociale

La partie Organisation d'un SMA correspond à la façon dont s'articulent les relations d'un agent à l'autre. A l'intérieur d'un groupe d'humains, on emploierait le terme de ***relations*** au sens large [19]. L'aspect Organisation d'un SMA rassemble les rapports entre agents, des plus formalisés (hiérarchie, statut, rôle) aux plus subjectifs (respect, confiance, préférences). Cet aspect du système est intimement lié à la partie Interaction, sans laquelle les relations à l'autre sont impossibles [45].

Nous présenterons cette section en reprenant la distribution sur trois dimensions : les dimensions fonctionnelle, spatiale et temporelle [41].

a) Distribution fonctionnelle

La distribution fonctionnelle répartit aux différents agents leurs rôles (et statuts) en tenant compte de leur aptitude à tenir ces rôles. Cette distribution spécifie aussi les liens entre agents (relations de pouvoir, interactions, etc.). Dans la littérature, l'organisation d'un système est souvent réduite à ce point tant ce choix est prépondérant au niveau de la fonctionnalité du système.

Cristof Baeijs énumère dans sa thèse [28] cinq grandes catégories d'organisations (l'organisation à membre unique, le groupe, la hiérarchie (à un ou plusieurs niveaux), l'organisation décentralisée et le marché) qui correspondent à des relations de contrôle et de communication entre agents et entre agents et ressources.

Un même SMA peut être composé de différents sous-groupes relevant chacun d'un type d'organisation différent (ne serait-ce qu'en vertu du principe de récursion).

D'une manière assez générale, l'augmentation de complexité d'une structure organisationnelle va de pair avec l'augmentation de son adaptabilité et avec l'augmentation de la quantité de messages échangés au sein du système. L'analyse de Cristof Baeijs montre que, dans le cadre d'agents réactifs, il existe une complémentarité entre la coordination (« *coût de maintenance des liens de communication et coût des échanges de messages* ») et la vulnérabilité (« *coût nécessaire pour s'adapter à un changement de l'environnement ou de l'organisation* »).

b) Distribution spatiale

La distribution spatiale se rapporte à la place de l'agent dans l'environnement.

Cet aspect de l'organisation est intimement lié aux environnements que l'agent partage avec d'autres agents. Ainsi, dans le cadre de l'application Microb-2 [41], application à la ROBOCUP, le terrain (l'environnement) est séparé en trois zones qui conditionnent trois sous-équipes aux propriétés distinctes :

- l'équipe d'attaque, composée d'agents réactifs auto-organisés ;
- l'équipe de milieu de terrain, dotée d'une organisation dynamique ;
- l'équipe de défense, disposant de capacités d'apprentissage distribué.

Le terme « spatial » peut induire en erreur, car les environnements dans lesquels les agents ont une place ne sont pas forcément de nature spatiale au sens commun du terme.

c) Distribution temporelle

La distribution temporelle concerne les aspects dynamiques des organisations.

Dans le cadre d'organisations statiques, la dimension temporelle n'existe pas ; dans le cadre d'organisations dynamiques, elle conduit les variations le long des deux autres axes, fonctionnel et spatial.

L'intérêt d'une organisation statique est essentiellement en terme de coût de communication et de résolution de conflits. En effet, en fixant une organisation précise, le concepteur élimine a priori plusieurs conflits susceptibles d'apparaître dans les interactions entre les agents [41]. De même, le concepteur peut spécifier de manière particulièrement précise les méthodes d'interaction à mettre en œuvre. Ces avantages apparaissent aux dépens de l'autonomie interactionnelle de l'agent.

Les organisations dynamiques permettent de leur côté de s'adapter, au sens large, c'est-à-dire réagir à des changements d'objectifs, à l'arrivée ou à la sortie d'agents, à la modification de l'environnement, etc. Cette capacité d'adaptation s'acquiert souvent au prix d'une plus grande activité de communication entre les agents et/ou une plus grande complexité interne de l'agent. Face à ce problème, Kelly Fernandes [42] propose un système multi-agents basé sur une hiérarchie, capable d'ajouter ou de retirer des niveaux à cette hiérarchie (en ajoutant ou retirant des agents) selon la difficulté du problème à traiter. Si le problème est trop complexe, le système s'adapte en faisant appel à d'autres agents, si le problème est trop simple, les agents superflus sont supprimés. Ainsi, la complexité de la structure est précisément adaptée à la complexité du problème (les pénalisations dues aux interactions sont minimales), mais, bien sûr, au prix de mécanismes (parfois coûteux) d'adaptation de la structure.

d) Liens entre ces trois distributions

Les trois axes que nous venons de présenter ne sont pas indépendants. Nous pouvons le voir, en prenant l'exemple de la ROBOCUP (simulation). Dans une équipe où les agents ont tous les mêmes capacités, leur comportement sera essentiellement conditionné par leur position sur le terrain, la position de la balle et celle des adversaires. L'organisation spatiale des agents peut entraîner l'attribution d'un statut d'attaquant (organisation fonctionnelle) à un agent positionné de manière avantageuse [28]. Inversement, si l'organisation fonctionnelle est fixe (pas de variation sur la dimension temporelle), un agent ayant un statut de défenseur restera dans une zone de terrain donnée. De manière triviale, notons aussi que le déplacement d'un agent entraîne un changement de distribution spatiale le long de l'axe temporel.

3.6.2. Contrôle

Le contrôle est un mécanisme de résolution de certains problèmes et conflit, il est régi soit par un agent unique (contrôle centralisé) soit par plusieurs agents (contrôle distribué). On va plus détailler le contrôle dans la section 3.7.

3.6.3. Coopération

3.6.3.1. Principe de la coopération

La coopération est une caractéristique très importante dans un SMA. En effet, une résolution distribuée d'un problème est le résultat de l'interaction coopérative entre les différents agents. Par définition la coopération consiste à faire en sorte que des agents travaillent à la satisfaction d'un but commun ou du moins à la satisfaction des buts individuels [11].

Dans le cadre de la coopération un agent doit :

- Mettre à jour le modèle du monde environnant.
- Intégrer des informations venant d'autres agents.
- Interrompre un plan pour aider d'autres agents.
- Déléguer la tâche qu'il ne sait pas résoudre à un autre agent dont il connaît les compétences.

3.6.3.2. Modèle de coopération

Quelle que soit l'organisation d'une société d'agents, un agent peut coopérer suivant les modèles suivants :

a. coopération par partage de tâches

Le partage des tâches est une forme de coopération entre les agents du système, adaptée surtout aux problèmes décomposables facilement en sous problèmes. Le problème est distribué entre les différents agents travaillant indépendamment les uns des autres ; chaque agent dispose des ressources et des compétences nécessaires pour accomplir la tâche qui lui a été désignée. Le contrôle est dirigé par les buts, et les agents sont représentés par les tâches qu'ils se sont engagés à exécuter. [46]

b. coopération par partage de résultats

Les agents ne peuvent accomplir leurs tâches de manière indépendantes. Ils sont appelés à se transmettre mutuellement des résultats partiels. [46]

c. Coopération par commande (maître/ esclave)

Un agent supérieur A décompose le problème en sous-problèmes qu'il repartit entre les agents Xi. Ceux-ci résolvent les sous-problèmes et renvoient les solutions partielles à l'agent A. [11]

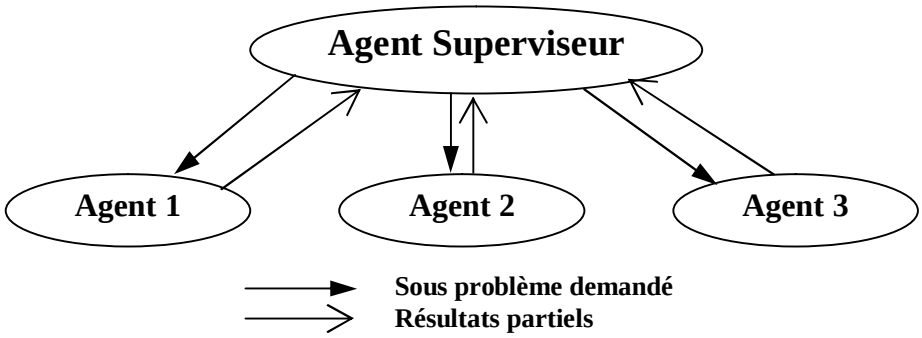


Figure 3-4 : Modèle de coopération : Mode de commande

d. Coopération par appel d'offres

Un agent « A » décompose le problème en des sous-problèmes dont il diffuse la liste. Chaque agent « Xi » qui le souhaite envoie une offre ; « A » choisit parmi celles-ci et distribue les sous-problèmes. Le système travaille ensuite en mode commande [11].

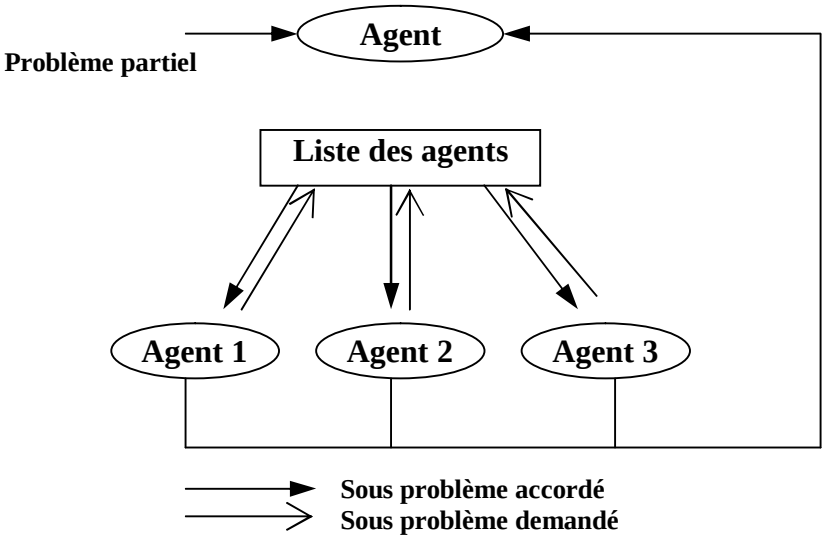


Figure 3-5 : Modèle de coopération : Mode appel d'offre

e. Coopération par compétition

Dans le mode compétition, « A » décompose et diffuse la liste des sous-problèmes comme dans le mode appel d'offre, chaque agent « Xi » résout un ou plusieurs sous-problèmes et envoie les résultats correspondants à « A » qui à son tour fait le tri [11].

3.7. Le Contrôle dans un SMA

3.7.1. Principe du contrôle

La distribution des données et des connaissances sur un ensemble d'agents induit à la nécessité d'un comportement globalement cohérent. Pour cela, il faut définir un mécanisme de contrôle qui assure la convergence du système vers une solution.

Le contrôle est un mécanisme de résolution de certains problèmes et conflit, il est régi soit par un agent unique (contrôle centralisé) soit par plusieurs agents (contrôle distribué). Le mécanisme de contrôle s'occupe de divers problèmes dont :

- Antagonisme entre les buts du système et les buts des agents.
- Disponibilité et partage des ressources.
- Solutions respectives d'agents exclusives.
- Conflit entre agents.

Pour mener à bien sa tâche, le système de contrôle doit être efficace (choisir l'action appropriée à exécuter le plus rapidement, avec un nombre d'hypothèses créées, de ressources utilisées et communications effectuées réduits et en satisfaisant le plus de contraintes possibles) souple (insertion et modification facile de nouvelles composantes), général, clair et uniforme (mécanisme de résolution identique).

3.7.2. Contrôle en Intelligence Artificielle Distribuée

La problématique du contrôle a été poussée plus loin dans le domaine de l'informatique avec les problèmes rencontrés au sein des systèmes répartis et des systèmes d'Intelligence Artificielle Distribuée (IAD) [10].

L'approche IAD étudie et résout les problèmes rencontrés lorsque plusieurs systèmes ou agents interagissent pour la résolution d'un même problème (Résolution Distribuée de Problèmes) ou parce qu'ils sont dans le même environnement (Multi-Agents).

3.7.2.1. Contrôle individuel

Un agent est un système concret ou abstrait capable de percevoir son environnement sur lequel il peut entreprendre des actions. Pour chaque agent, le problème du contrôle relatif à sa propre activité est le même que celui évoqué précédemment pour un système de résolution de problèmes excepté que l'évolution de l'environnement doit être prise en compte. L'agent est inséré dans un environnement qu'il peut percevoir mais dont il ne maîtrise pas a priori l'évolution. [12]

Le problème du contrôle au sein d'un agent ou contrôle individuel consiste à déterminer quelle action exécuter à partir de son état de résolution et de l'état de l'environnement.

3.7.2.2. Contrôle social

Un agent inséré dans une société doit inscrire son activité dans la résolution de celle-ci. Les agents étant répartis, une vision globale de l'ensemble de l'activité leur est impossible. Des systèmes de surveillance du trafic routier, par exemple, répartis sur un carrefour ne peuvent pas, sans diminution de leur performance, avoir une vision globale de la circulation : en effet, ne pouvant capter qu'une région du carrefour, ils ne peuvent se construire une vision globale que par communication avec les autres agents. Ceci se déroule au dépend de leur activité de surveillance. D'autre part, les informations qu'ils peuvent avoir ainsi reçues sont rapidement traitées impliquant une continuelle remise à jour. [12]

Lors de la surveillance du trafic par ces agents, chacun d'eux a reçu une région limitée à surveiller. Le problème de contrôle au niveau de l'agent peut donc être plus facilement résolu. Cependant de nouvelles contraintes apparaissent. En effet, quelle doit être la distribution des différents problèmes à résoudre (sélection des agents et des régions)? On peut résoudre ce problème lors de la conception du système comme on peut réaliser dynamiquement la distribution du contrôle. Un autre problème est celui des échanges entre les agents : la solution est-elle construite indépendamment par chacun des agents et intégrée à la fin? Ou des échanges réguliers ont-ils lieu entre les agents afin de réduire les incohérences possibles? [12]

Dans l'exemple précédent, les agents sont conçus pour résoudre ce problème de surveillance du trafic routier. Plaçons-nous maintenant dans un cadre général où des agents possédant des capacités plus importantes sont dans un même environnement : des robots se déplaçant dans un couloir reliant différentes pièces. Ce couloir ne permet pas le passage de deux robots de front. Dans cet exemple aussi, une vision globale de l'état de l'environnement est exclue. Nous voyons apparaître un problème de partage de ressources relatif au passage dans le couloir. Les agents voulant emprunter ce couloir vont donc devoir se coordonner pour que la circulation puisse se réaliser sans blocage. [12]

Nous mettons ainsi clairement en évidence une dimension qui était absente dans notre exemple précédent : les agents prennent dynamiquement en compte le but commun qui consiste à résoudre le problème de l'accès à la ressource, et mettent en place un mode de coordination définissant les règles de résolution de ce problème. Les agents doivent

établir un équilibre permanent entre leurs intérêts propres et ceux des autres agents. En effet une règle de conduite doit s'établir pour la circulation dans le couloir qui fait prévaloir l'évitement du blocage par rapport, par exemple, à la volonté d'un agent de traverser immédiatement le couloir.

Le problème du contrôle social ou contrôle de la société dans un système d'IAD consiste à déterminer pour tout agent les contraintes à imposer sur ses actions et ses interactions. Ces contraintes établissent un équilibre entre des critères locaux (objectifs de l'agent) et des critères sociaux (objectifs partagés par tous les agents). [12]

Les stratégies proposées pour le résoudre définissent des protocoles d'interaction et l'organisation (centralisation, décentralisation) qui établit les relations de pouvoir entre les agents.

a) Contrôle centralisé

Le contrôle centralisé est caractérisé par le fait qu'un seul agent se charge du contrôle ce qui permet [46]:

- Une mise au point facile du système.
- Une meilleure efficacité du système.

Mais il présente certains inconvénients dont nous citons les suivants [46]:

- La possibilité de communication dépend de son emplacement.
- La capacité de traitement dépend de son emplacement.
- La capacité de traitement dépend de son processeur.
- La fiabilité est compromise par la défaillance de cet agent.
- S'il existe un délai de communication les informations peuvent être obsolètes ce qui entraîne des prises de décision inadéquates.

b) Contrôle décentralisé (distribué)

Dans ce type de contrôle, on peut avoir un groupe d'agents dédié au contrôle ou la répartition de cette tâche sur l'ensemble des agents du système.

Parmi les structures d'organisation des agents on cite [46]:

- L'organisation horizontale : tous les agents ont le même niveau du fait qu'il n'y a pas de maître et esclave.
- L'organisation verticale : caractérisée par une structure à plusieurs niveaux, chaque niveau est structuré de manière horizontale.

- Appel d’offres : un agent diffuse une liste de sous problèmes et attend les propositions des autres agents intéressés par la résolution d’un ou plusieurs sous problème afin de choisir l’un d’eux.

3.8. Communication dans un SMA

Pour pouvoir considérer les agents comme un collectif, il doit exister entre eux des possibilités de communication. En effet, la communication est un élément très important dans les systèmes multi-agents, sans la communication les agents ne peuvent pas coopérer, coordonner leurs actions et réaliser les tâches en commun. Pour ces raisons, la communication est à la base des interactions et de l’organisation sociale. Elle permet d’assurer la cohérence de résolutions entreprises localement par chacun des agents.

3.8.1. Protocole de communication

Dans un système distribué, il faut que les entités disposent d'un langage commun pour pouvoir échanger des informations et coopérer pour la résolution d'un problème. Ce langage, appelé mécanisme de communication interprocessus, est un ensemble de primitives connues par chaque entité et dont l'ensemble constitue un protocole de communication.

Les primitives de base d’un protocole sont [46] [11]:

- Etablissement d’une connexion entre deux entités.
- Identification du nœud destinataire dans un réseau de communication.
- Envoie de données.
- Définition d’un type de communication : synchrone ou asynchrone.

3.8.2. Type de communication

Il existe quatre types de communication [46]

- Sélective ou distribuée.
- Sollicitée ou non sollicitée.
- Avec ou sans accusé de réception.
- Transmission simple ou répétée.

3.8.3. Modèles de communication

Parmi les méthodes utilisées, deux grandes catégories peuvent se distinguer, sur la base de la destination du message. Dans la première catégorie, communication par envoi

de messages, le destinataire est un ou des agents. Dans la seconde, la communication par partage d'information, le message est déposé dans un espace commun. [45]

3.8.3.1. Communication par envoi de messages

Certains des paramètres spécifiant le message envoyé portent l'identifiant du ou des destinataires. Ce modèle de comunication est marqué par les techniques classiques de la communication en informatique, typiquement le modèle de shanon. Dans cette situation, l'agent émetteur a un rôle actif, tandis que l'agent récepteur est passif dans sa réception de message. Le message est déplacé jusqu'à son destinataire. [45]

Ce modèle est utile quand l'expertise d'un domaine peut être totalement distribuée à un ensemble d'agents. Chaque agent a des connaissances et des mécanismes de résolutions distincts. Il possède une base de connaissance locale et échange des messages avec les autres agents qu'il connaît (ses accointances) [31].

On distingue deux (02) modes dans la communication par envoi de message :

- La communication directe (point à point)
- La communication par diffusion (one to all)

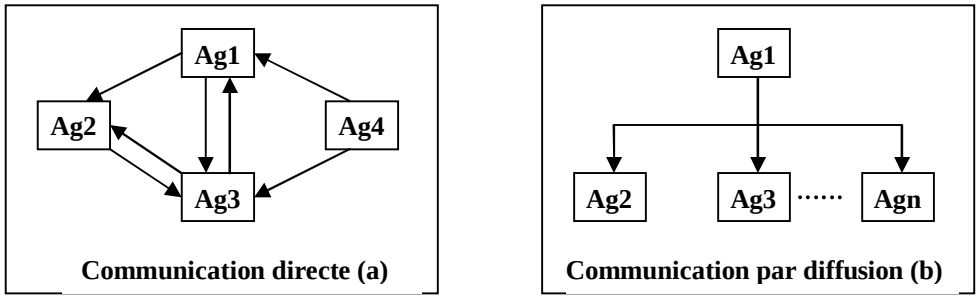


Figure 3-6 : Communication par envoi de message

3.8.3.2. Communication par partage d'information (tableau noir)

En intelligence artificielle la technique du tableau noir (« blackboard ») est très utilisée pour spécifier une mémoire partagée par divers systèmes. Dans un SMA utilisant un tableau noir, les agents peuvent écrire des messages, insérer des résultats partiels de leurs calculs et obtenir de l'information. Le tableau noir est en général partitionné en plusieurs niveaux qui sont spécifiques à l'application. Les agents qui travaillent sur un niveau particulier peuvent accéder aux informations contenues dans le niveau correspondant du tableau noir ainsi que dans des niveaux adjacents. Ainsi, les données peuvent être synthétisées à n'importe quel niveau et transférées aux niveaux supérieurs

alors que les buts de haut niveau peuvent être filtrés et passés aux niveaux inférieurs pour diriger les agents qui oeuvrent à ces niveaux.

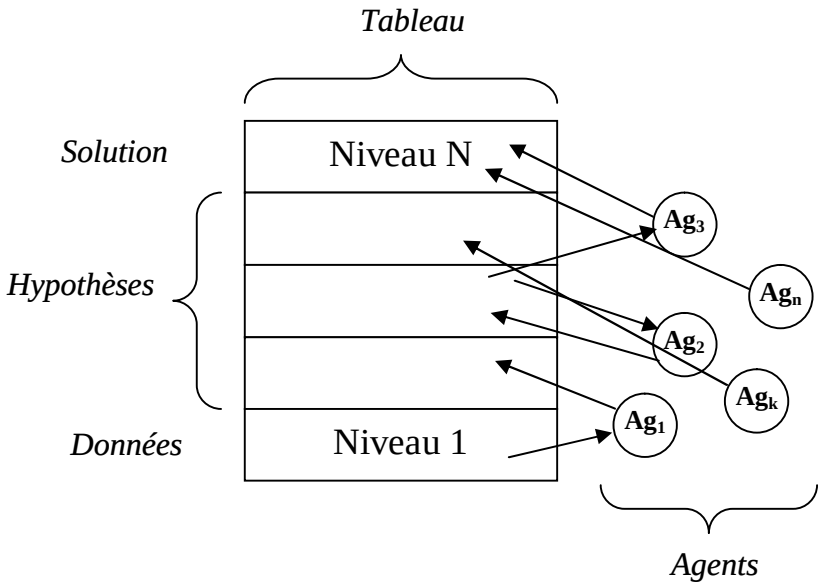


Figure 3-7 : Communications par partage d'informations

3.9. Modèles et langages de développement des systèmes multi-agents

3.9.1. Le modèle des Acteurs

Un des premiers modèles proposé de l'IAD fut le modèle des Acteurs [40]. Les acteurs sont des composantes autonomes d'un système qui communiquent par messages asynchrones. Ils sont composés d'un ensemble de primitives, parmi lesquelles on retrouve:

- *Create* : créer un acteur à partir d'un ensemble de paramètres décrivant son comportement;
- *Send* : envoyer un message à un autre acteur;
- *Become* : changer l'état local d'un acteur.

Les acteurs s'avèrent être un modèle assez naturel pour le calcul parallèle. Cependant, les divers modèles d'acteurs, comme bien d'autres modèles de l'IAD font face à un problème de cohérence. Leur granularité fine pose des problèmes de comportement dans des systèmes qui renferment plusieurs acteurs. Ils éprouvent également des difficultés à atteindre des buts globaux avec seulement des connaissances locales. [40]

3.9.2. Contract Net

Le protocole du Contract Net fut une des premières solutions au problème d'allocation de tâches auquel fait face généralement un ensemble de résolveurs de problèmes. Dans ce protocole, les agents peuvent prendre deux rôles: *gestionnaire* ou

contracteur. L'agent qui doit exécuter une tâche donnée (le gestionnaire) commence tout d'abord par décomposer cette tâche en plusieurs sous-tâches. Il doit ensuite annoncer les différentes sous-tâches au reste des agents de l'environnement. Les agents qui reçoivent une annonce de tâche à accomplir peuvent ensuite faire une proposition devant refléter leur capacité à remplir cette tâche. Le gestionnaire rassemble ensuite toutes les propositions qu'il a reçues et alloue la tâche à l'agent ayant fait la meilleure proposition. [40]

3.9.3. Le modèle du Tableau Noir

Un tableau noir doit être compris comme un modèle de résolution de problème pour organiser les connaissances et organiser l'utilisation des connaissances, afin de construire une (des) solution (s). En effet, la connaissance nécessaire pour résoudre un problème peut provenir de différentes sources. Chaque source :

- Apporte une part d'information qui distribue à la construction d'une solution.
- Lit un ensemble d'informations sur le tableau et le met à jour en appliquant ses connaissances
- Est responsable des conditions sous lesquelles elle peut contribuer à une solution

3.9.4. Le langage KQLM

Le langage KQML (Knowledge Query Manipulation Langage) a été proposé pour supporter la communication inter-agents. Ce langage définit un ensemble de types de messages (appelés abusivement « performatifs ») et des règles qui définissent les comportements suggérés pour les agents qui reçoivent ces messages. Les types de messages de KQML sont de natures diverses : simples requêtes et assertions (ex : *ask*, *tell*) ; instructions de routage de l'information (*forward* et *broadcast*) ; commandes persistantes (*subscribe*, *monitor*); commandes qui permettent aux agents consommateurs de demander à des agents intermédiaires de trouver les agents fournisseurs pertinents (*advertise*, *recommend*, *recruit* et *broker*). Ce langage a été développé pour les besoins des développeurs d'agents logiciels. Le terme « performatif » a été utilisé pour nommer diverses commandes qui ont une certaine ressemblance avec des verbes utilisés de façon performative dans le langage naturel ; l'interprétation sémantique actuelle de ces commandes n'est pas satisfaisante. [40]

3.9.5.Le langage ACL

Le langage ACL (Agent Communication Langage) est basé sur la théorie du langage et a bénéficié grandement des résultats de recherche de KQLM. Si toutefois, les deux langages se rapprochent au niveau des actes du langage, il n'en ait rien au niveau de la sémantique et il

semble qu'un grand soin a été apporté au niveau de ACL tant au niveau de certains protocoles qui sont plus explicites qu'au niveau de la sémantique des actes eux-mêmes [40].

3.9.6.Le langage Objet

Un O*bjet* est un ensemble de données sur lesquelles des procédures peuvent être appliquées. Ces procédures ou fonctions applicables aux données sont appelées *méthodes*. La programmation d'un objet se fait donc en indiquant les données de l'objet et en définissant les procédures qui peuvent lui être appliquées.

Il se peut qu'il y ait plusieurs objets identiques, dont les données ont bien entendu des valeurs différentes, mais qui utilisent le même jeu de méthodes. On dit que ces différents objets appartiennent à la même *classe* d'objets. Une classe constitue donc une sorte de type, et les objets de cette classe en sont des *instances*. La classe définit donc la structure des données (appelées *champs* ou *variables d'instances*) que les objets correspondants auront, ainsi que les méthodes de l'objet. À chaque instanciation, une allocation de mémoire est faite pour les données du nouvel objet créé. L'initialisation de l'objet nouvellement créé est faite par une méthode spéciale, le *constructeur*, lorsque l'objet est détruit, une autre méthode est appelée : le *destructeur*. L'utilisateur peut définir ses propres constructeurs et destructeurs d'objets si nécessaire.

3.10. Quelques exemples de systèmes multi-agents

3.10.1. Le système ADEPT

Les gestionnaires de grandes compagnies effectuent des prises de décisions en se basant sur une combinaison de jugement et d'informations provenant de plusieurs départements. Idéalement, toutes les informations pertinentes devraient être rassemblées avant qu'une décision ne soit prise. Cependant, le processus d'obtenir des informations, qui sont à jour et pertinentes, est très complexe et prend énormément de temps. Pour cette raison, plusieurs compagnies ont cherché à développer des systèmes informatiques afin de les assister dans leur processus d'affaires. [40]

Le système ADEPT attaque ce problème en voyant le processus d'affaires comme un ensemble d'agents qui négocient et qui offrent des services. Chaque agent représente un rôle distinct ou un département de l'entreprise et est en mesure de fournir un ou plusieurs services. Les agents qui requièrent les services d'autres agents le font par une négociation qui permet d'obtenir un coût, un délai temporel et un degré de qualité qui sont acceptables aux deux parties. Le résultat d'une négociation terminée avec succès constitue un engagement entre les deux parties. [40]

3.10.2. Le système GUARDIAN

Le système GUARDIAN a pour but de gérer les soins aux patients d'une unité de soins intensifs chirurgicale. Les principales motivations de ce système sont: premièrement, le modèle des soins d'un patient dans une unité de soins intensifs est essentiellement celui d'une équipe, où un ensemble d'experts dans des domaines distincts coopèrent pour organiser les soins des patients ; deuxièmement, le facteur le plus important pour donner de bons soins au patients est le partage d'informations entre les membres de l'équipe de soins critiques. Particulièrement, les médecins spécialistes n'ont pas l'opportunité de superviser l'état d'un patient minute par minute ; cette tâche revient aux infirmières qui, quant à elles, ne possèdent pas les connaissances nécessaires à l'interprétation des données qu'elles rassemblent. [40]

Le système GUARDIAN répartit donc le suivi des patients à un certain nombre d'agents de trois types différents. Les agents perception/action sont responsables de l'interface entre GUARDIAN et le monde environnant, établissant la relation entre les données des senseurs et une représentation symbolique que le système pourra utiliser, et traduisant les requêtes d'action du système en commandes pour les effecteurs (effectors). Les agents en charge du raisonnement sont responsables d'organiser le processus de prise de décision du système. Finalement, les agents en charge du contrôle (il n'y en a habituellement qu'un seul) assurent le contrôle de haut niveau du système. [40]

3.10.3. Le système KISS

Le système KISS (Knowledge-based Image Segmentation System) est né d'une convergence entre les besoins d'un domaine, la Vision, et les outils d'un autre champ de recherche, l'Intelligence Artificielle Distribuée. Cette convergence s'exprime autour de trois maîtres mots: distribuer, coopérer et intégrer. La distribution des connaissances permet son introduction sous une forme adaptée à son mode d'utilisation, la distribution des tâches permet le développement de stratégies coopératives de résolution, l'existence d'un schéma de conception « intégrateur », enfin, permet un développement parfaitement modulaire (introduction de nouvelles tâches, complexification des connaissances, par exemple). [5]

L'Environnement de Développement

Le système KISS a été conçu à partir d'un environnement de génération de systèmes multi-agents. Cet environnement, appelé MAPS (Multi-Agent Problem Solving Environment) met en jeu deux classes d'agents, appelés Serveurs (KS) et Processeurs (KP) de

Connaissances, qui communiquent par envoi de messages. Ces agents découlent d'une catégorisation des connaissances attachées à la résolution d'un problème en termes de connaissances descriptives et opératoires et permettent la distribution des savoir-faire et des comportements attachés à leur exploitation. [5]

- ***Serveurs de Connaissances***

  Un Serveur de Connaissances (noté KS, pour Knowledge Server) est un agent dont le rôle est de représenter, maintenir et transmettre des connaissances à caractère descriptif. Il doit être capable non seulement de recevoir des informations, mais également d'en requérir, de les propager, d'en vérifier la consistance, d'en proposer en retour à l'extérieur. [5]

- ***Processeurs de Connaissances***

  Un Processeur de Connaissances (noté KP, pour Knowledge Processor) est un agent dont le rôle est de représenter et mettre en oeuvre les connaissances opératoires attachées à la résolution d'un sous-problème. Parmi les tâches qui lui sont imparties, il doit être capable de résoudre les problèmes qui lui sont soumis et d'exploiter les informations qui lui sont transmises à des fins d'analyse ou de déduction. [5]

3.10.4. Le système NETSA

Le système NETSA est un système multiagent coopératif, développé à l'université Laval (Canada) est destiné aux environnements riches en informations. Ce système comporte plusieurs types d'agents [40]:

- Un *agent* utilisateur en charge de la cueillette et du filtrage des informations provenant et allant vers l'usager ;
- Un agent *courtier* servant de répertoire pour les agents qui évoluent au sein de NETSA ;
- Des agents *ressources* reliés chacun à une ressource d'informations et pouvant rapatrier et mettre à jour les données ;
- Un agent d'*exécution* en charge de la décomposition des tâches et du suivi du déroulement d'exécution des différentes sous-tâches ;
- Un agent *ontologie* en charge du maintien de la cohérence des concepts utilisés par les agents.

3.11. Conclusion

Les systèmes multi-agents sont apparus comme résultats de l'évolution des techniques d'intelligence artificielle distribuée (IAD), un SMA se compose d'un ensemble d'entités (agents) spécialisées coopérantes pour le compte d'une application globale.

Pour fonctionner, un SMA nécessite un calculateur ou une machine de programmation parallèle. PVM (Parallel Virtuel Machine) a été réalisé dans ce contexte pour permettre au processus de différentes natures de coordonner leurs actions en parallèle.

Toutes ces idées seront développées dans le chapitre suivant.

CHAPITRE 4
PARALLELISME ET ENVIRONNEMENT PVM

4.1. Introduction

Une grande majorité des ordinateurs fonctionnent encore de nos jours selon le modèle séquentiel proposé par « Von Neuman » dans les années 40. Pendant longtemps, l'accroissement des performances a été rendu possible par l'évolution de la microélectronique, en conservant toujours cette architecture. Mais les progrès de l'électronique se trouvent ralentis par des contraintes physiques, qui rendent difficile une augmentation de puissance. Face à cette contrainte les efforts d'amélioration ont été penchés sur le parallélisme des traitements [39].

Comme première étape, des systèmes multi-tâches sont apparus en donnant l'illusion à l'utilisateur de travailler simultanément sur plusieurs tâches (programmes), mais cela n'a pas résolu le problème des besoins en calculateurs de plus en plus performants. D'ou la nécessité, à défaut de pouvoir augmenter de manière significative les performances des processeurs classiques, de dupliquer les éléments déjà présents dans les architectures séquentielles (unité de calcul, unité de contrôle, mémoire), modifiant par la même l'architecture et le modèle de fonctionnement de la machine, et introduisant le problème nouveau des communications inter-processeurs [39].

Dans ce chapitre, nous avons commencé par un petit rappel sur les systèmes multi-tâches suivi par une introduction aux machines parallèles en décrivant leurs architectures, ainsi que leurs classes. En fin, nous avons présenté l'outil de programmation parallèle PVM (Parallèle Virtuel Machine) que nous avons utilisé pour la réalisation de notre application.

4.2. Applications Multi-Tâches

4.2.1. Définition

Les applications multi-tâches sont découpées en plusieurs parties afin qu'ils s'exécutent sur un même processeur en partageant son temps de calcul. Et cela donne l'illusion d'un traitement parallèle des données [46].

4.2.2. But d'utilisation

Cette technologie apparaît dés que l'on a besoin d'une grande puissance de calcul tels que pour les Simulateurs de vol, les Simulateurs d'incident nucléaire, les animations vidéo, l'analyse d'image etc. De même que d'ordre économique, du fait que plusieurs machines d'une puissance de calcul P et d'un coût moindre qu'une unique machine de telle puissance [46].

4.3. Introduction au parallélisme

4.3.1. Classification des machines parallèles (Classification de Flynn)

En 1969, « M.J. Flynn » proposa un système de classification des différents modèles de fonctionnement envisageables pour un calculateur. Cette classification est toujours utilisée de nos jours, et elle nous servira à situer plus précisément le système P.V.M (Parallel Virtual Machine) au sein des différents types d'architectures parallèles. Cette classification est fondée sur deux critères [39]:

- La machine a-t-elle un ou plusieurs flux de données? (Single ou Multiple Data stream).
- La machine a-t-elle un ou plusieurs flux d'instructions? (Single ou Multiple Instruction stream).

La quasi-totalité des ordinateurs disponibles à l'heure actuelle sont conçus sur la base d'une de ces quatre architectures [39].

4.3.1.1. Machine SISD

Une machine *SISD* (*S*ingle *I*nstruction *S*ingle *D*ata) est ce que l'on appelle d'habitude une machine de Von Neuman. Une seule instruction est exécutée et une seule donnée (simple, non-structurée) est traitée à tout instant [36].

Le code suivant,

```
int A[100];
...
for (i=1;100>i;i++)   A[i]=A[i]+A[i+1];
```

s'exécute sur une machine séquentielle en faisant les additions A[1]+A[2], A[2]+A[3], etc., A[99]+A[100] à la suite les unes des autres [39].

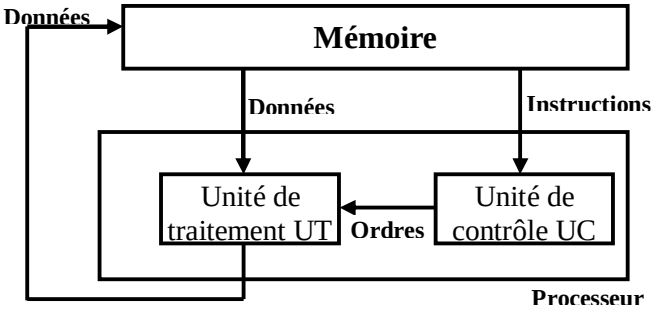


Figure 4-1 : Synoptique d'une machine SISD

4.3.1.2. Machine SIMD

Une machine *SIMD* (*S*ingle *I*nstruction *M*ultiple *D*ata) peut être de plusieurs types, parallèle ou systolique. En général l'exécution en parallèle de la même instruction se fait en même temps sur des processeurs différents (parallélisme de donnée *synchrone*) [36]. Examinons par exemple le code suivant écrit en CM-Fortran sur la Connexion Machine-5 avec 32 processeurs, [36]

```
      INTEGER I,A(32,1000)
CMF$  LAYOUT A(:NEWS,:SERIAL)
       ...
      FORALL (I=1:32,J=1:1000)
$     A(I:I,J:J)=A(I:I,J:J)+A(I:I,(J+1):(J+1))
```

Chaque processeur *i*, $1 \leq i \leq 32$ a en sa mémoire locale une tranche du tableau A: A(i,1), A(i,2), ..., A(i,1000). Il n'y a pas d'interférence dans le calcul de la boucle entre les différentes tranches: tous les processeurs exécutent la même boucle sur leur propre tranche en même temps [36] (cf. figure 4.2).

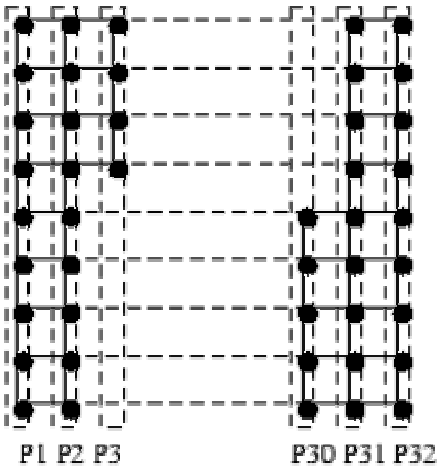


Figure 4-2 : Réparation d'un tableau sur les processeurs d'une machine SIMD

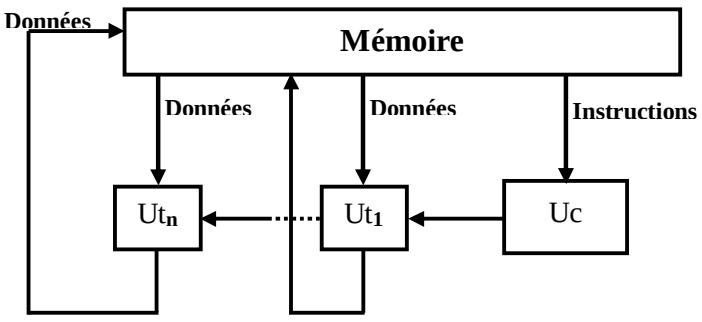


Figure 4-3 : Synoptique d'une machine SIMD

Une caractéristique essentielle de ces architectures est de conserver un synchronisme total au cours de l'exécution d'un programme.

4.3.1.3. Machine MISD

Une machine *MISD* (*M*ultiple *I*nstruction *S*ingle *D*ata) peut exécuter plusieurs instructions en même temps sur la même donnée. Cela peut paraître paradoxal mais cela recouvre en fait un type très ordinaire de micro-parallélisme dans les micro-processeurs modernes : les processeurs vectoriels et les architectures pipelines [36].

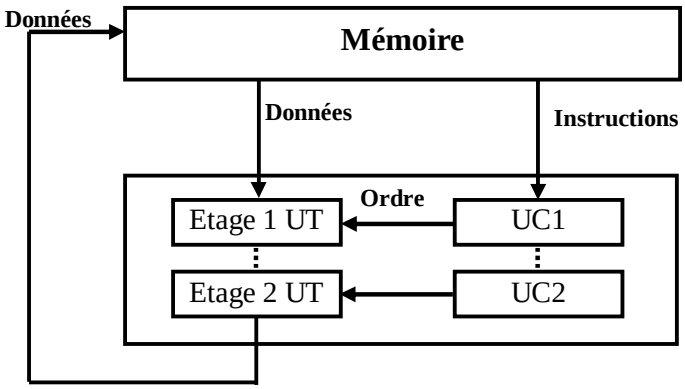


Figure 4-4 : Synoptique d'une machine MISD

4.3.1.4. Machine MIMD

Le cas des machines *MIMD* (*M*ultiple *I*nstruction *M*ultiple *D*ata) est le plus intuitif et est celui qui va nous intéresser le plus dans notre sujet. On a plusieurs types d'architecture possibles [36]:

- Mémoire partagée (Sequent)
- Mémoire locale avec réseau de communication (Transputer, Connexion Machine, local, par réseau d'interconnexion), ou système réparti

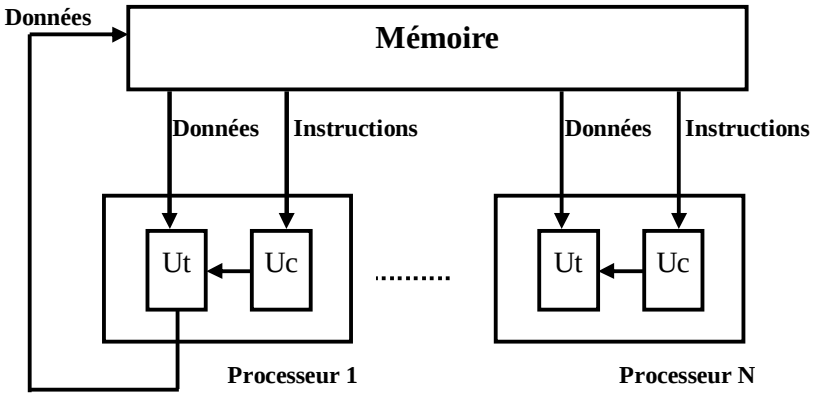


Figure 4-5 : Synoptique d'une machine MIMD

Le schéma, ci-dessus, présente l'architecture d'un système M.I.M.D. à mémoire commune, sur ce principe les différents processeurs communiquent entre eux grâce à cette mémoire (opération de Lecture/Ecriture aux adresses mémoire)[39].

P.V.M., pour sa part peut utiliser un modèle M.I.M.D. avec mémoire distribuée. En effet, chaque processeur possède sa propre mémoire, les échanges d'informations entre les processeurs se font grâce au réseau (opération de Send/Receive) [36].

§ Mémoire Partagée

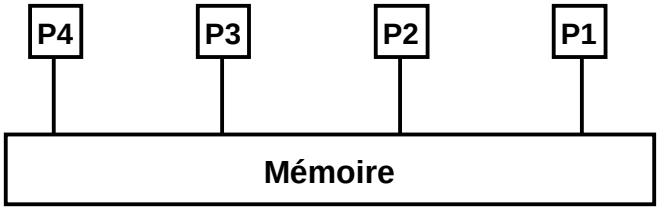


Figure 4-6 : Architecture simplifiée d'une machine à mémoire partagée

L'existence d'une mémoire commune entre les processeurs simplifie le travail de parallélisation des algorithmes. Une parallélisation semi-automatique, assistée par l'utilisateur, permet à partir d'un code séquentiel existant d'obtenir des performances acceptables sur ces machines. En effet, les ordinateurs à mémoire partagée exploitent efficacement le parallélisme de grain fin. Le parallélisme à grain fin est plus facile à extraire et à contrôler que le parallélisme à grain plus gros nécessaire aux machines à mémoire distribuée. Cependant, dans les ordinateurs à mémoire partagée, le nombre de processeurs et limité. Le goulot d'étranglement que consiste l'accès à une mémoire commune limite le nombre de processeurs à une dizaine. De plus, la plupart des applications nécessitant un super-calculateur ont des besoins de mémoire vive très importants (26].

La synchronisation des exécutions des processeurs (ou processus) est nécessaire dans certains cas. Si elle n'était pas faite, il y aurait un risque d'incohérence des données. Partant de x=0, exécutons x:=x+x en parallèle avec x:=1. On a alors essentiellement trois exécutions possibles (en supposant chacune des affectations compilées en instructions élémentaires insécables, ou "atomiques", comme suit) [36]:

Tableau 4-1 : Les trois cas de calcule de x+x en parallèle

LOAD x,R1	WRITE x,1	LOAD x,R1
LOAD x,R2	LOAD x,R1	WRITE x,1
WRITE x,1	LOAD x,R2	LOAD x,R2
WRITE x,R1+R2	WRITE x,R1+R2	WRITE x,R1+R2
Résultat x=0	Résultat x=2	Résultat x=1

Cela n'est évidemment pas très satisfaisant; il faut rajouter des synchronisations pour choisir tel ou tel comportement, et en tout cas éviter le troisième comportement en général. Les deux premiers sont acceptables en un certain sens, ils sont deux possibles *séquentialisations* des processus parallèles [36].

La synchronisation peut se faire par différents mécanismes [36]:

- Barrières de synchronisation,
- Sémaphores : deux opérations *P* et *V*.
- Verrou (Mutex Lock) : sémaphore binaire qui sert à protéger une section critique.
- Moniteurs : construction de haut niveau, verrou implicite.
- Etc…

Les opérations *P* et *V* sur les sémaphores sont parmi les plus classiques et élémentaires. On peut considérer qu'à chaque variable partagée *x* dans la mémoire est associé un "verrou" (du même nom) indiquant si un processus est en train de manipuler la variable, en interdisant son accès. L'opération *Px* exécutée par un processus verrouille ainsi son accès exclusif à *x*. L'opération *Vx* ouvre le verrou et permet à d'autres processus de manipuler *x* à leur tour [36].

La encore des erreurs sont possibles, en voulant trop synchroniser par exemple. Il peut y avoir des cas d'interblocage (deadlock, livelock) en particulier [36].

Supposons qu'un processus T_1 ait besoin (pour effectuer un calcul donné) de verrouiller puis déverrouiller deux variables *x* et *y* dans l'ordre suivant: *Px* puis *Py* puis *Vx* puis *Vy* alors qu'un autre processus, en parallèle, désire faire la séquence *Py*, *Px* puis *Vy* et enfin *Vx*. En fait les deux processus peuvent s'interbloquer l'un l'autre si T_1 acquiert *x* (respectivement T_2 acquiert *y*) puis attend *y* (respectivement *x*) [36].

§ Machine distribuée

Chacun des microprocesseurs possède sa propre zone mémoire. Les différents noeuds de calcul (ensemble microprocesseur, mémoire) sont reliés entre eux par un réseau de communication. Les processeurs accèdent directement seulement à leur zone de mémoire. On inclut dans les machines à mémoire distribuée aussi les ordinateurs comme le CRAY-T3E qui ont un espace d'adressage unique et qui possèdent un mécanisme d'accès-mémoire distant. Ce mécanisme permet à un processeur d'accéder directement à la mémoire d'un autre processeur via le réseau. Mais un accès à la mémoire d'un autre processeur est plus lent qu'un accès à la mémoire locale [26].

Actuellement, les ordinateurs à mémoire distribuée sont les seuls ordinateurs qui permettent d'utiliser simultanément plusieurs dizaines à quelques centaines de processeurs. On parle alors de parallélisme massif. Les ordinateurs à mémoire distribuée sont, en raison de leur architecture, plus extensibles que les ordinateurs à mémoire partagée. C'est-à-dire qu'il est possible d'augmenter le nombre de noeuds de calcul [26].

Il existe aussi aujourd'hui des machines avec une architecture mixte. Il s'agit d'ordinateurs à mémoire distribuée dont les noeuds de calcul sont des ordinateurs parallèles à mémoire partagée. La machine SGI-ORIGIN vendue comme une machine à mémoire partagé est en fait une machine mixte qui possède plusieurs noeuds de calcul reliés par un réseau. Les noeuds sont composés de deux processeurs qui partagent une mémoire locale et ils accèdent à la mémoire des autres noeuds par des requêtes de lectures/écritures à travers le réseau. Nous pensons que ce type d'architecture est amené à se développer, car elle permet d'exploiter plusieurs grains de parallélisme simultanément [26].

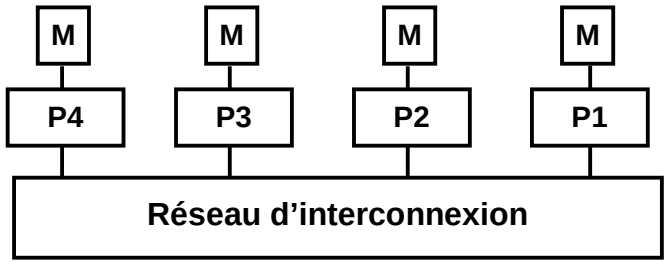


Figure 4-7 : Architecture simplifiée d'une machine à mémoire distribué

4.3.2. Méthodologie de parallélisation

La parallélisation explicite comporte plusieurs étapes : il faut trouver le parallélisme, puis découper le problème en tâches. Il faut ensuite définir un ordre d'exécution et associer à chacune des tâches un processeur chargé de leur exécution [26].

4.3.2.1. Parallélisme de données

La première source de parallélisme est le parallélisme de données. Il existe quand une même opération est appliquée à chacun des éléments d'un ensemble de données. Cette source de parallélisme est par exemple très fréquente dans les problèmes d'algèbre linéaire dense. Des langages tels que HPF (High Performance FORTRAN) permettent aisément d'exprimer et d'exploiter cette forme de parallélisme. Dans ce mode d'expression du parallélisme, le travail des processeurs est guidé par la distribution des données. Les communications sont définies comme des phases de redistribution des données sur les

processeurs. Le programme parallèle est une succession de phases de calculs et de phases de communications pour la redistribution des données. Cependant, le parallélisme de données ne permet pas de décrire efficacement toutes les formes de parallélisme. Il est souvent difficile de connaître a priori si le parallélisme de données est adapté à un problème donné [26].

4.3.2.2. Parallélisme de contrôle

Plus général que le parallélisme de données, le parallélisme de contrôle consiste à décrire un algorithme parallèle sous la forme d'un graphe orienté sans cycle. Les nœuds du graphe sont des suites d'opérations élémentaires exécutées en séquentiel, ce sont les tâches du graphe. Les arcs du graphe indiquent les contraintes de précédence entre les tâches. C'est-à-dire l'utilisation par une tâche d'une donnée calculée par une tâche précédente. Ce graphe, appelé graphe de précédence ou graphe de tâches, définit un ordre partiel sur les tâches. Les tâches non ordonnées par cet ordre partiel peuvent être exécutées en parallèle. Le travail des processeurs est ici guidé par les dépendances entre les tâches [26].

Le parallélisme de contrôle est plus général que le parallélisme de données. Il est ainsi possible de traiter le parallélisme de données comme un cas particulier de parallélisme de contrôle [26].

4.3.2.3. Choix de la granularité

Lors de la description d'un algorithme parallèle sous la forme d'un graphe de tâches, choisir la granularité consiste à définir plus précisément ce qu'est une tâche [26]. Empiriquement, on peut définir la granularité comme le rapport entre le temps moyen d'exécution des tâches divisé par la largeur du graphe de tâches. La largeur du graphe de tâches est le nombre maximum de tâches potentiellement exécutables en parallèle. Elle est fortement liée au volume des communications, c'est pourquoi la granularité est également définie comme le rapport entre le temps de calcul divisé par le temps de communication.

Ainsi pour diminuer la granularité, il faut découper chaque tâche en sous-tâches plus petites. On parle de grain plus fin. En pratique, diminuer la granularité augmente le parallélisme et permet d'utiliser plus de processeurs. Inversement, augmenter la granularité consiste à regrouper des tâches. On parle alors de grain plus gros. Augmenter la granularité permet de réduire le temps nécessaire à la gestion du parallélisme [26].

Le choix de la granularité dépend fortement de la machine-cible et de l'application. Il existe même une opposition importante entre le type des machines et la granularité qu'elles peuvent traiter efficacement. En effet, les ordinateurs parallèles à mémoire partagée

peuvent gérer un parallélisme de grain fin avec un sur-coût faible. Mais le nombre de processeurs de ces machines est limité. Inversement, les ordinateurs à mémoire distribuée permettent d'exploiter le maximum de parallélisme. Mais un grain trop fin entraîne souvent trop de communications entre les processeurs. Et un grain trop gros entraîne des déséquilibres de charge entre les noeuds de calcul. Le choix de la granularité est un compromis entre le maximum de parallélisme potentiel, un sur-coût de gestion minimum, et un bon équilibre de charge entre les noeuds de calcul. C'est une étape primordiale, il est souvent utile de pouvoir modifier facilement le grain d'une application pour pouvoir affiner ce choix [26].

4.4. Parallel Virtual Machine (P.V.M.)

4.4.1. Introduction

P.V.M. (**P**arallel **V**irtual **M**achine) est une librairie qui offre la possibilité de gérer dynamiquement un réseau de machines hétérogènes. Ces machines peuvent elles-mêmes être parallèles ou séquentielles. PVM permet donc de transformer un cluster de machines en une seule machine virtuelle parallèle à mémoire distribuée.

PVM est constitué de deux parties :

- Un démon qu'il faut installer sur une machine particulière. Il faut voir ce démon comme une console qui gère la machine virtuelle en permettant par exemple d'ajouter des machines, de lancer des processus, de les supprimer, etc.
- La librairie à proprement parler qui contient des fonctions pour initialiser des processus et pour changer la configuration de la machine.

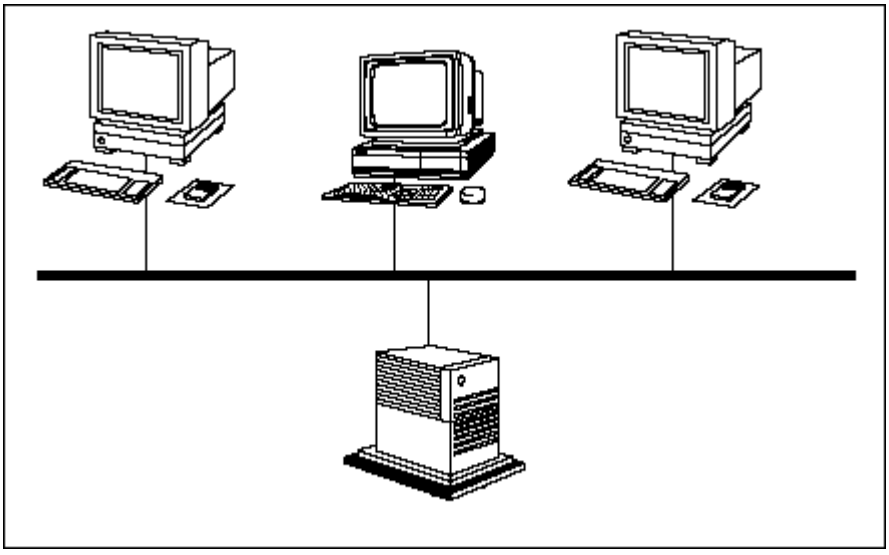


Figure 4-8 : Architecture physique de PVM

4.4.2. Le Principe

P.V.M. permet [39]:

- de définir une *machine virtuelle* composée d'un ensemble de machines (homogènes ou non)
- de démarrer automatiquement des tâches sur la machine virtuelle
- aux tâches de communiquer et de se synchroniser.

Les applications qui peuvent être écrites aussi bien en fortran77 qu'en C, peuvent être parallélisées en utilisant des structures classiques d'envoi de messages que l'on retrouve sur d'autres machines distribuées. En envoyant et recevant des messages, les nombreuses tâches d'une application peuvent coopérer afin de résoudre le problème grâce au parallélisme [39].

Des systèmes de tolérances aux fautes permettent également à P.V.M. d'enlever de la machine virtuelle toute machine qui aurait une défaillance [39].

Puisque le réseau est supposé hétérogène, un mécanisme de conversion entre les différentes représentations de données encapsule les communications. Ce mécanisme n'est malheureusement pas transparent à l'utilisateur qui, pour envoyer (resp. recevoir) des données, doit empaqueté (resp. désempaqueté) son message. De même la gestion des tampons, tâche délicate de tout système de communication asynchrone, est reporté au niveau de l'utilisateur: c'est à lui d'allouer et libérer l'espace mémoire nécessaire. Toutefois en réception, du fait de l'asynchronisme, l'allocation des tampons est automatique. PVM fixe les règles de gestion qui doivent être respectées [22] :

- L'émission de plusieurs messages se fait séquentiellement, c'est-à-dire qu'il ne peut y avoir qu'un seul tampon actif à la fois.
- Dès réception d'un message le tampon précédent est automatiquement libéré, à moins qu'il n'ait été au préalable explicitement sauvegardé.

Ainsi l'émission d'un message se fait en trois étapes :

1- allocation d'un tampon.
2- empaquetage du message dans le tampon.
3- expédition du message.

Inversement la réception d'un message par une tâche se déroule en deux phases :

1- réception du message
2- désempaquetage

Pour l'émission deux primitives [22]: **pvm_send** et **pvm_mcast**, servent respectivement à l'envoi de messages à une seule ou à un ensemble de tâches. Ces deux opérations sont asynchrones, c'est-à-dire que la tâche appelante ne reste suspendue que jusqu'à l'expédition complète du message par le noeud. Afin de permettre la réalisation d'un contrôle de flux, la gestion de priorités ou encore le typage des informations, une étiquette doit être associée à chaque message.

Pour spécifier les tâches réceptrices il faut spécifier les identificateurs uniques que PVM associe à chacune d'elles, la diffusion d'un message par **pvm_mcast** se fait donc en spécifiant tous les récepteurs.

Quel que soit le mode d'émission, point-à-point ou diffusion, les mêmes fonctions sont utilisées pour recevoir les messages: **pvm_nrecv** pour une réception non bloquante, et **pvm_recv** pour une réception bloquante. Parmi les messages disponibles la sélection se fait selon l'identificateur de la tâche émettrice, et selon l'étiquette attendue. Toutes les combinaisons sont possibles suivant que l'un et/ou l'autre des deux critères sont spécifiés ou laissés libres.

Afin de permettre une meilleure coopération entre tâches, notamment pour la synchronisation ou la diffusion, PVM intègre la notion de ***groupe*** de tâches. Les primitives de manipulation des groupes sont assez générales et n'imposent pas de réelles restrictions. De ce fait la sémantique d'un groupe n'est soumise qu'aux trois règles suivantes [22]:

- **Dénomination :** chaque groupe est nommé par l'utilisateur.
- **Dynamicité** : une tâche peut à tout moment se joindre (**pvm_joingroup**) ou quitter (**pvm_lvgroup**) un groupe.
- **multiplicité**: une tâche peut appartenir à plusieurs groupes en même temps, sans limitation.

4.4.3. Utilisation

Le daemon pvmd3:

Le daemon ***pvmd3*** est lancé sur une machine qui fait partie de la machine virtuelle, il permet de contrôler les échanges entre les tâches. Soit il est lancé à partir de la console PVM, soit il est lancé à partir d'un fichier de configuration qui permet de décrire la machine virtuelle [39].

Chaque daemon pvmd3 créé deux (02) fichiers de trace stockés dans le répertoire/tmp de la machine sur laquelle il est lancé. Ainsi toutes les machines consultant la machine virtuelle ont ces fichiers dans leur répertoire local/tmp [39]..

La console PVM :

Elle permet de démarrer le daemon pvmd local s'il n'est pas présent.

Le prompt **>pvm** indique que la console est active [39].

Tableau 4-2 : Commandes principales de la console PVM

Pvm> help	Aide en ligne sur les commandes de la console
Pvm>add ***M1***	lance le daemon pvmd sur la machine ***M1*** et donc la rajoute dans la machine virtuelle
Pvm>delete ***M2***	enlève la machine ***M2*** de la machine virtuelle
pvm>conf	affiche la liste des machines appartenant à la machine virtuelle
pvm>ps	liste les tâches actives de la machine virtuelle
pvm>kill ***taskid***	tue la tâche ***taskid***
pvm>reset	tue toutes les tâches
pvm>spawn ***task***	démarre la tâche ***task***

Donc pour lancer une application avec la console les étapes sont les suivantes:

- lancement de pvm
- description de la machine virtuelle (**add**)
- lancement de l'application (**spawn**)

4.4.4. Développement

P.V.M. permet d'utiliser tous les modèles de la classification de Flynn, néanmoins les plus utilisés sont ceux basés sur le S.I.M.D. et sur le M.I.M.D [39].

Lorsque l'on définit une application parallèle avec PVM basée sur la technique de passage de message, il est important que les différentes tâches ne passent pas tout leur temps dans des fonctions de communication d'informations. Le parallélisme est choisi pour gagner du temps. Par conséquent, si l'application est trop complexe à paralléliser (cas par exemple, d'une trop forte dépendance de données), il ne faut pas perdre de temps si au final, il n'y a aucun gain du fait du temps passé à modifier l'algorithme [39].

Le modèle maître/esclave utilise deux programmes [39].

- un maître qui
  - distribue les données aux esclaves

- lance les tâches que doivent exécuter les esclaves
- reçoit les données traitées
- peut exécuter des opérations séquentielles ou d'Entrées/Sorties

- des esclaves qui
  - reçoivent des données
  - traitent ces données
  - émettent les données traitées.

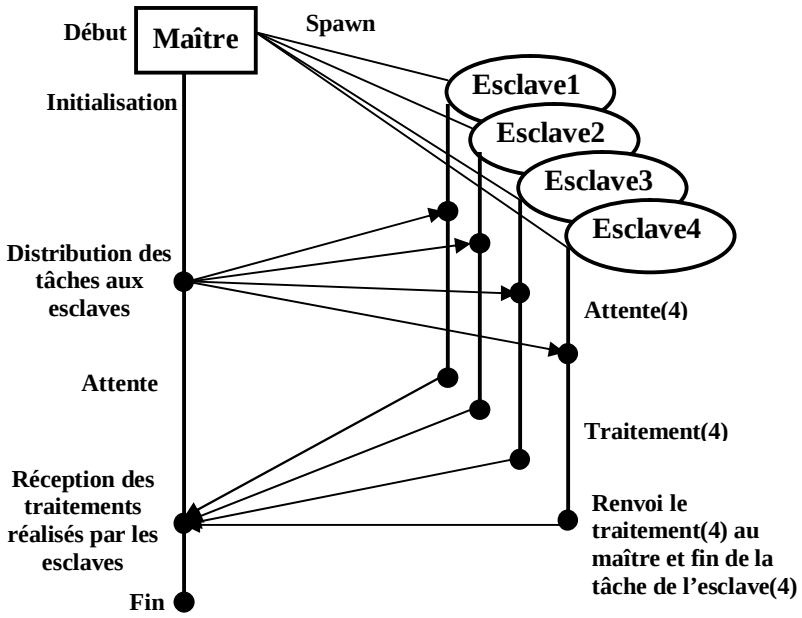


Figure 4-9 : Modèle Maître/Esclave

4.4.5. Programmation avec PVM

4.4.5.1. Les identificateurs de tâches

Les identificateurs de tâches sont des entiers de 32 bits. Cet entier est découpé en champs contenant certaines informations comme sa position dans la machine virtuelle ou encore un identificateur de processus. Cet entier est géré par PVM. L'utilisateur accède aux informations grâce notamment à deux fonctions. La première donnant l'identificateur de la tâche courante et l'autre celui du parent. Voici la syntaxe de ces deux fonctions :

- int myTid = **pvm_mytid()** qui retourne l'identificateur de tâche;
- int parentTid = **pvm_parent()** qui retourne l'identificateur de tâche du processus parent.

4.4.5.2. Créations et destructions de tâches

Lorsqu'une exécution démarre, il n'y a qu'un processus actif. C'est au programmeur de créer les processus qu'il veut mettre en oeuvre. La création se fait par l'intermédiaire de la fonction pvm_spawn dont voici la syntaxe:

int numt = **pvm_spawn**(char *task,char **argv,int flag,char *where, int ntask,int *tids)

Où les différents paramètres sont :

- *numt* : nombre de tâches effectivement démarrées.
- *task* : nom de l'exécutable;
- *argv* : pointeur sur un tableau d'arguments qui serra passé à l'exécutable task;
- *flag* : options de la fonction :
  - PvmTaskDefault : PVM choisis une machine au hasard;
  - PvmTaskHost : la variable where spécifie alors la machine cible;
  - PvmTaskArch : la variable where spécifie alors l'architecture de la machine cible;;
  - ... : ;
- *where :* machine cible;
- *ntask* : nombre de processus à démarrer;
- *tids* : remplis au retour de la fonction par les identificateurs de tâche des processus démarrés;

Inversement, on peut aussi vouloir la détruire. Cela se fait de la manière suivante :

int info = **pvm_kill**(int tid)

- *tid* : tâche à détruire;
- *info* : status retourné par la fonction (< 0 indique une erreur).

4.4.5.3. Communication de base en PVM

Les communications en PVM ont recours à l'*active message buffer*. L'idée de ce buffer de message est de grouper des messages de différents types dans un seul buffer, donc de minimiser le nombre d'envois et par conséquent de minimiser le temps de latence induit par chaque message. Ce buffer est en fait une zone mémoire dans laquelle on peut disposer un ensemble de messages que l'on va envoyer et un espace dans lequel on va en recevoir.

- ***Gestion du buffer de messages***

  La fonction **pvm_initsend**(int encoding) permet d'effacer le buffer de message par défaut et de spécifier le type de codage du buffer (encoding). Ce type de codage peut prendre les trois valeurs suivantes:

  - *PvmDataDefault* utilise un codage compréhensible par différentes architectures,
  - *PvmDataRaw* ne code pas le message (gain de temps) et finalement
  - *PvmDataInPlace* copie dans le buffer seulement un pointeur sur le message et la taille du message.

- ***Envoi du buffer de messages***

  Un buffer de message est envoyé à un autre processeur grâce à la fonction **pvm_send()** dont voici la syntaxe :

  int info = **pvm_send**(int tid,int msgtag)

  - *tid* : identificateur de tâche du processus de destination;
  - *msgtag* : tag du message doit être >= 0;
  - *info* : informations sur l'exécution du send (< 0 indique une erreur).

- ***Réception du buffer de messages***

  Le message est reçu avec la commande **pvm_recv()** dont voici la syntaxe :

  int bufid = **pvm_recv**(int tid,int msgtag)

  - *tid* : identificateur de la tâche qui a envoyé (-1 désigne n'importe quelle tâche);
  - *msgtag* : tag du message doit être >= 0 (-1 désigne n'importe quel tag);
  - *bufid* : identificateur de buffer (< 0 indique une erreur).

- ***Création du message***

  Maintenant, on sait initialiser un buffer de message, l'envoyer et le recevoir. Il reste alors à voir la manière de le remplir. C'est assez simple, on y accède par un jeu de fonctions spécifiques. On dispose des éléments avec le jeu pack et on extrait avec le jeu unpack.

  Pour chaque type de base, il existe une fonction pack et unpack appropriée. Voyons comme exemple les deux fonctions à utiliser pour traiter des entiers signés :

  int info = **pvm_pkint**(int *ip,int nitem,int stride)

  int info = **pvm_upkint**(int *ip,int nitem,int stride)

  - *ip* : pointeur sur un entier;
  - *nitem* : nombre d'éléments à packer (! ne correspond pas au nombre de bytes);
  - *stride* : égal à 1 sauf si on pack des complexes (cplx).

On peut alors remplacer int par d'autres types de base en gardant la même logique des paramètres.

- **pvm_pkfloat** et **pvm_upkfloat** : pour les nombres réels
- **pvm_pklong** et **pvm_upklong** : pour les nombres entiers longs
- **pvm_pkuint** et **pvm_upkuint** : pour les nombres entiers non signés

4.4.5.4. Gestion dynamique des tâches

Le concept de groupe de processus est un aspect important de PVM. Grâce aux groupes, il est possible de synchroniser des processus ou de faire des communications collectives.

Ce ne sont pas les démons pvmd qui gèrent les opérations de groupe, mais un serveur de groupe qui sera lancé sur une des stations de la machine virtuelle, dès la première instruction pvm nécessitant la création d'un groupe. Ce processus s'ajoute alors au « n » processus que vous lancez vous-même.

En PVM, il y a deux manières de modifier la taille d'un groupe : en le joignant ou en le quittant. La première tâche joignant un groupe effectue une création de groupe (transparente pour l'utilisateur) et la dernière qui le quitte en entraîne la destruction. Deux fonctions sont disponibles. La première qui permet de joindre un groupe :

int inum = **pvm_joingroup**(char *group)

- *group* : nom du groupe;
- *inum* : instance de la tâche (< 0 indique une erreur).

et la deuxième qui permet de quitter un groupe :

int info = **pvm_lvgroup**(char *group)

- *group* : nom du groupe;
- *info* : status retourné par la fonction (< 0 indique une erreur).

Pour avoir des informations sur un groupe, on peut utiliser diverses fonctions comme:

int inum = **pvm_getinst**(char *group,int tid)

- *group* : nom du groupe;
- *tid* : identificateur de tâche;
- *inum* : instance de la tâche (< 0 indique une erreur).

qui donne l'instance de la tâche dans le groupe. Inversement, d'une instance on peut obtenir l'identificateur de tâche.

int tid = **pvm_gettid**(char *group, int inum)

- *group* : nom du groupe;
- *tid* : identificateur de tâche (< 0 indique une erreur);
- *inum* : instance de la tâche.

La taille d'un groupe peut être obtenu grâce à la fonction :

int size = **pvm_gsize**(char *group)

- *group* : nom du groupe;
- *size* : nombre de membre présent dans le goupe (< 0 indique une erreur).

Une fois que le groupe est créé, on peut l'utiliser pour faire des communications collectives ainsi que pour effectuer des barrières de synchronisation. Ces dernières se font de la manière suivant :

int info = **pvm_barrier**(char *group,int count)

- *group* : nom du groupe;
- *count* : nombre de membre devant atteindre la barrière;
- *info* : status retourné par la fonction (< 0 indique une erreur).

Un ensemble de communications collectives peuvent être réalisée. La diffusion (one-to-all) se fait de la manière suivante:

int info = **pvm_bcast**(char *group,int msgtag)

- *group* : nom du groupe;
- *msgtag* : tag du message;
- *info* : status retourné par la fonction (< 0 indique une erreur).

4.5. Conclusion

Nous avons présenté dans ce chapitre une introduction aux machines parallèles en décrivant leurs architectures ainsi que leurs classes.

De même, nous avons étudié l'environnement de programmation parallèle PVM (Parallel Virtual Machine). PVM est une plate forme qui permet à plusieurs processus ou machines hétérogènes de communiquer et de coopérer en parallèle. Nous allons utiliser cette plate forme (PVM) pour implémenter notre système multi-agents.

Nous allons maintenant entamer la partie pratique.

CHAPITRE 5
IMPLEMENTATION ET RESULTATS

5.1. Introduction

Notre travail consiste à réaliser un Système Multi-Agents (SMA). Ce système est composé de plusieurs agents, chaque agent est chargé d'une tâche bien spécifiée et peut communiquer avec un ou plusieurs autres agents. L'ensemble des agents (le système) a pour but de traiter des images, en passant par les différentes étapes de l'analyse d'images.

5.2. Description du Système multi-agents

5.2.1. Introduction

Après avoir donné un aperçu des méthodes d'analyse d'images et des systèmes multi-agents dans les chapitres précédents, nous allons maintenant présenter l'implantation d'une méthode de segmentation coopérative que nous avons réalisé par le système multi-agents.

Cette méthode profite de la robustesse en précision dans la localisation des contours par un opérateur de détection de contours, et de la puissance d'une croissance de régions dans la détection de régions homogènes.

Nous expliquons tout d'abord, les règles utilisées pour le prétraitement de l'image, ensuite nous définirons le principe de la méthode de segmentation coopérative suivi par la présentation de l'implémentation de cette méthode à l'aide d'un système multi-agents.

5.2.2. Les règles de décision utilisées pour le prétraitement de l'image

En tenant compte des principes cités au premier chapitre, nous avons choisi comme informations à traiter pour la phase de prétraitement « le contraste, la luminosité et l'homogénéité (opposé : la texture) ».

Nous avons utilisé des règles de décision de type « si-alors » pour adapter le traitement et pour donner au système un certain niveau d'expertise afin de choisir les méthodes et les paramètres adéquats son l'intervention de l'utilisateur, ces règles sont détaillées comme suite :

- Si l'histogramme de l'image est mal réparti dans l'échelle (image trop foncée, très claire ou brumée) alors on applique un recadrage dynamique sur cette image.
- Si l'image est mal contrastée, alors on applique un rehaussement de contraste suivi par un filtrage non linéaire et cela pour éviter d'ajouter du flou à l'image. Le filtre choisi est celui de Nagao.

- Si l'image est bien contrastée et trop texturée, alors on applique seulement un filtrage (sans rehaussement du contraste) en utilisant le filtre moyen, et cela pour diminuer les effets de la texture.
- Si l'image est bien contrastée et moins texturée, alors on applique le filtre gaussien (sans rehaussement du contraste).

La question qui va être posée est comment savoir si l'image est trop foncée ou très clair, bien contrastée ou non, trop texture ou non ?

En réalité, il n'existe aucun formalisme mathématique qui peut résoudre ce problème. Pour cela, nous allons nous servir d'une conception expérimentale pour réaliser notre algorithme en se basant sur une stratégie heuristique.

Au début, nous avons calculé les paramètres concernés sur plusieurs images de différentes natures, et après plusieurs observations et interprétations, nous avons tiré les conclusions suivantes :

Si (La valeur du contraste est supérieure à '1')

Alors

L'image est bien contrastée

Sinon

L'image est mal contrastée

Si (La valeur de l'homogénéité est inférieure à '0.15')

Alors

L'image est trop texturée

Sinon

L'image n'est pas trop texturée

Si (L'histogramme de l'image est décalé vers la droite ou la gauche de plus de 45 degré à 7% pré)

Alors

Appliquer un recadrage dynamique

Sinon

Ne pas appliquer un recadrage dynamique

Les algorithmes de calculs des paramètres sont décrits dans l'annexe B

Remarque : Dans le cas de l'utilisation du filtre de Nagao, les variances calculées ne vont pas être prisonnières à l'intérieur de ce filtre, mais elles seront utilisées pour former le tableau des écart-types de l'image (c'est le principe '4' du premier chapitre « liberté des informations »). Ce tableau sera utilisé pour la segmentation en régions.

5.2.3. Définition de la méthode de segmentation coopérative

On entend par segmentation coopérative, une segmentation dans laquelle on corrèle l'extraction de plusieurs types de primitives ou d'informations. Elle combine les avantages de chacune prise séparément ; par exemple : la précision et la rapidité d'une segmentation en contours et la densité de l'information extraite d'une segmentation en régions.

5.2.3.1. Principe de la méthode

La coopération s'effectue en guidant la croissance des régions par une carte contour de type « Canny », la croissance des régions se produit par l'utilisation d'un seuil variable en fonction de l'écart-type de la zone traitée, et cela pour :

- Détecter les régions existantes dans une zone homogène en utilisant un seuil faible.
- Eviter d'avoir plusieurs petites régions dans une zone trop texturée en utilisant un seuil élevé.

Pour déterminer la fonction du seuil, nous avons utilisé une stratégie heuristique. Nous avons traité plusieurs images en variant le seuil de croissance et nous avons obtenu, celons notre expertise, le tableau suivant :

Tableau 5-1 : Seuils utilisés pour la croissance des régions

Ecart-type	2	4	6	9	15	18	22	25
Seuil	12	18	22	30	47	60	84	110

Pour réaliser cette fonction, nous avons choisi de faire une interpolation par la méthode des moindres carrés. Nous avons obtenu la fonction suivante :

$F(x) = 6{,}1430 + 3{,}2368\,x - 0{,}1066\,x^2 + 0{,}0036\,x^3 + 0{,}0001\,x^4$.....(5.1)

x : représente ici l'écart-type et F(x) c'est le seuil de croissance.

Le principe de la méthode des moindres carrés est décrit dans l'annexe A

5.2.3.2. Stratégie de la méthode

Le plan suivi de la méthode de segmentation coopérative est comme suit **:**

- Création de la carte contours initiale en utilisant la méthode de segmentation de Canny.
- Correction des contours en suivant ces points :
  - Ø Enchaînement de deux chaînes de contour qui paraissent être coupées.
  - Ø Elimination des points parasites qui se situent sur les contours.
  - Ø Suppression des petites chaînes de contour.
- Découpe de l'image en bloc et calcul de l'écart-type de chaque bloc et cela dans le cas où le filtre de Nagao n'a pas été utilisé dans la phase du prétraitement.
- Création de la carte régions initiale en utilisant la méthode de croissance des régions et en tenant compte des points suivants :

- Ø Le seuil utilisé dans l'agrégation des pixels est en fonction de l'écart-type du bloc en cours.
- Ø Si l'agrégation des pixels arrive sur des points contours alors la croissance s'arrêtera dans cette direction.

- Adhésion des petites régions aux régions les plus adéquates.
- Fusion des régions qui ont des moyennes en niveau de gris proche.
- Création de la carte régions finale.
- Création de la carte contours finale.

Le schéma bloc de la méthode de segmentation coopérative proposée est présenté dans la figure suivante :

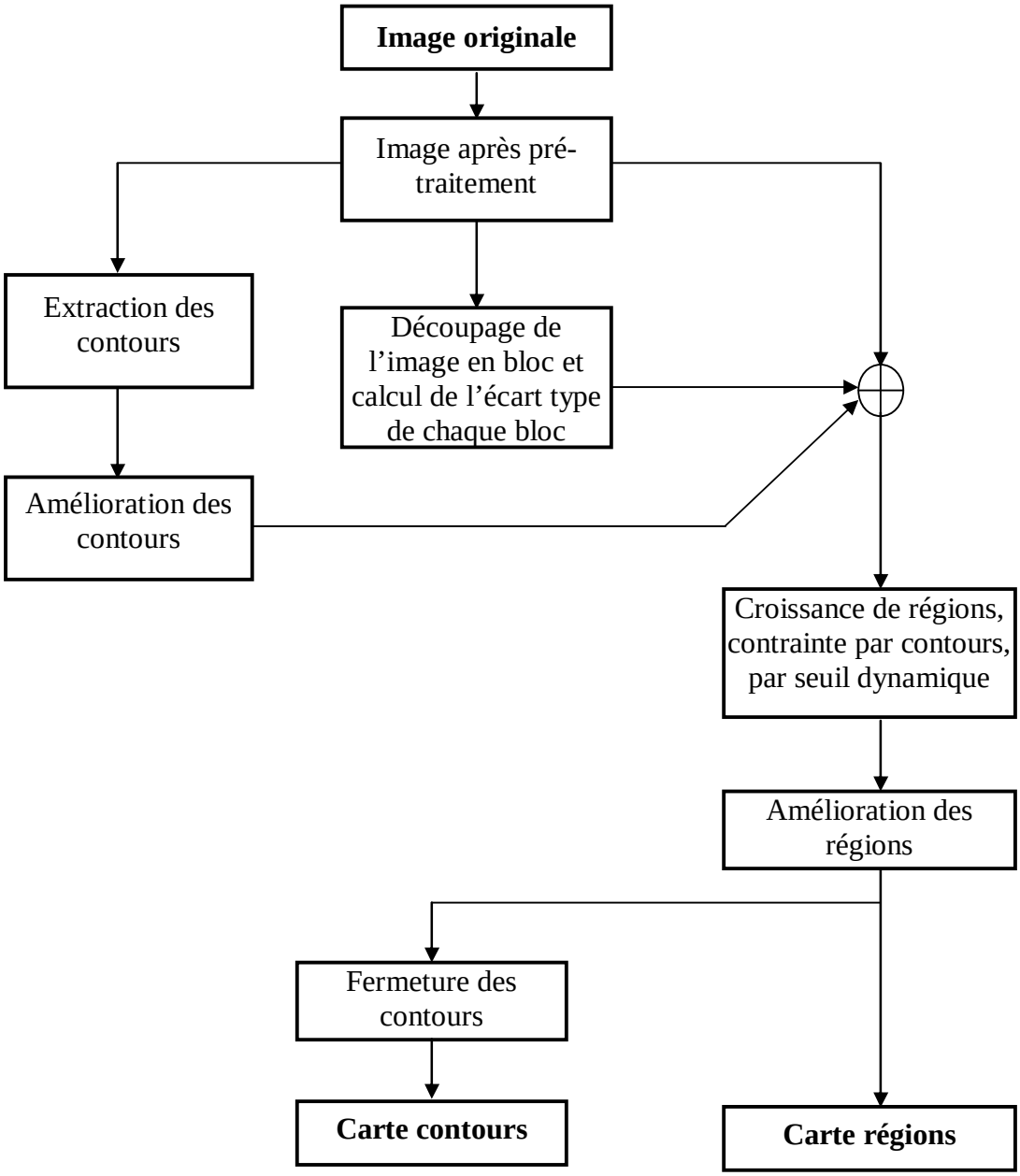


Figure 5-1 : Schéma bloc de la méthode de segmentation proposée

5.2.4. Motivation d'une implantation multi-agents de la méthode coopérative

Il existe de multiples avantages pour implanter cette méthode par un système multi-agents. On peut citer :

- **La flexibilité** : une implantation multi-agents fournit une flexibilité accrue, en séparant les différentes expertises du système. Dans le cadre de notre approche, nous pouvons envisager d'avoir des agents spécialisés pour estimer les caractéristique de l'image, pour prétraiter l'image, appliquer un opérateur de détection de contours et appliquer un opérateur de croissance de régions.

- **Le parallélisme** : le parallélisme peut être envisagé à plusieurs stades du traitement, par exemple dans l'étape d'estimation des caractéristiques de l'image. En effet, plusieurs agents contenants différents types d'estimateurs peuvent travailler en parallèle sur l'image initiale et les résultats peuvent être envisagés dans l'étape de segmentation coopérative contours-régions. Comme il peut être envisagé soit en filtrage, soit calcul des matrices de cooccurrences, soit pour l'opérateur de Canny, soit pour le calcul des matrices des écart type, pour cela on découpe l'image en blocs et chaque bloc est traité d'une façon parallèle par un agent spécialiste.

- **La Coopération** : la coopération peut être envisagée, par exemple, entre l'agent chargé de la segmentation contour et l'agent chargé de la segmentation région à fin d'avoir des meilleurs résultats.

5.2.5. Implantation du système multi-agents

Le système multi-agents que nous proposons est implanté sur dix-neuf processus. Nous avons choisi de partager ce système sur dix-neuf agents pour la raison que chaque agent représente un maillon dans la chaîne de traitement de l'analyse d'images. Nous avons appelé notre système : SMATI (Système Multi-Agents dédié au Traitement d'Images).

Le système a été réalisé en C++ sur des PC connectés par un réseau local .La modélisation par objet nous a permis de concevoir et de maintenir aisément le système.

Le système réalisé est focalisé sur l'aspect de distribution, de parallélisation et d'automatisation des traitements.

5.2.5.1. Architecture du système

L'architecture du système est décrite par la figure suivante :

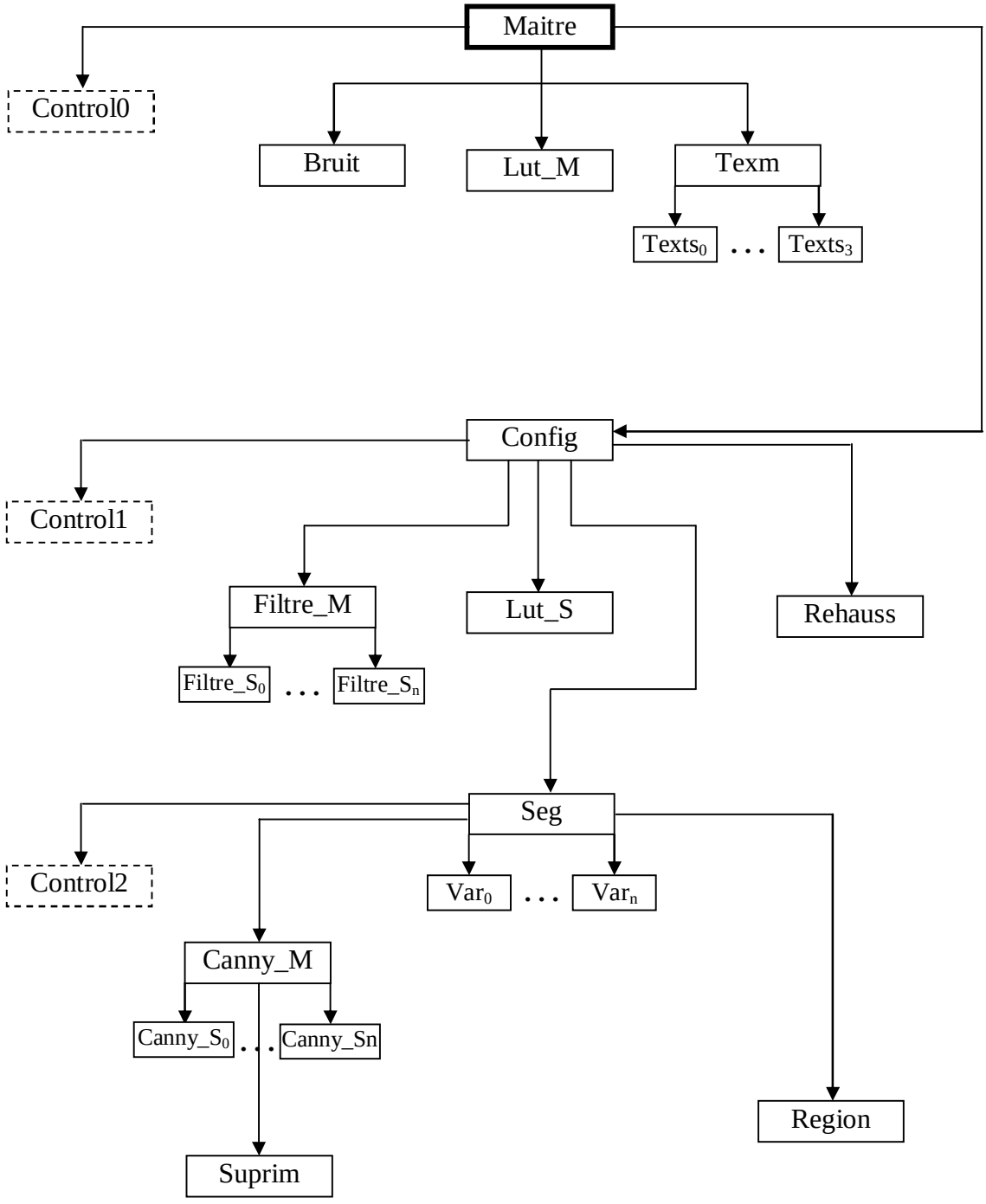


Figure 5-2 : Schéma de création des agents

L'architecture de ce système évolue dans le temps de manière dynamique : la structure présentée par la figure 5.2 représente le schéma de création des agents, de plus

cette architecture change d'une image a l'autre selon le besoin ou non de recadrage dynamique et de rehaussement de contraste. Quatre cas de figure se présente :

1 cas :

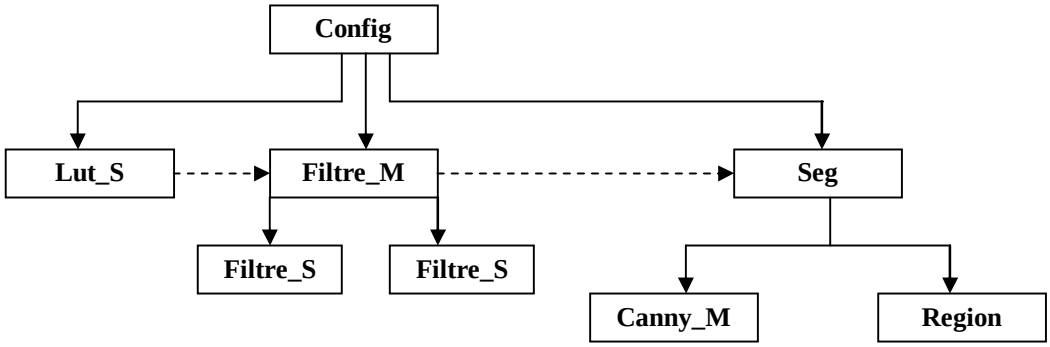


Figure 5-3 : Recadrage dynamique et filtrage

Dans ce cas, l'image traitée conduit l'agent Config à créer les trois agent Lut_S, Filtre_M et Seg (sans rehaussement de contraste).

2 cas :

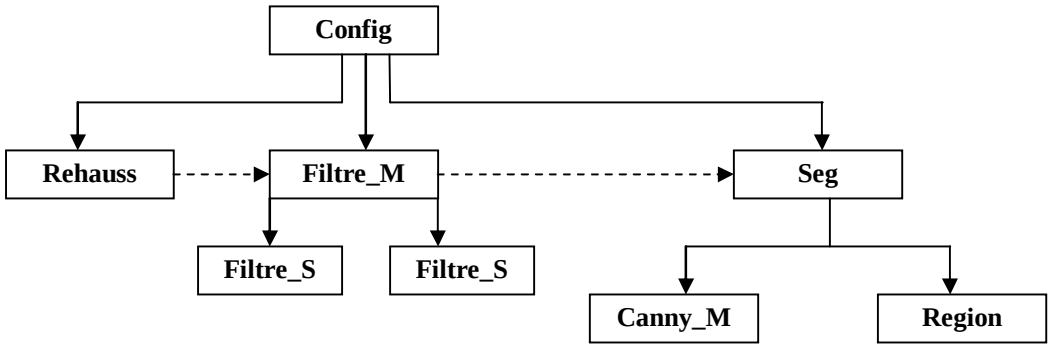


Figure 5-4 : Rehaussement et filtrage

Dans ce cas l'image traitée n'a pas besoin de recadrage dynamique, alors Config crée les trois agent Rehauss, Filtre_M et Seg.

3 cas :

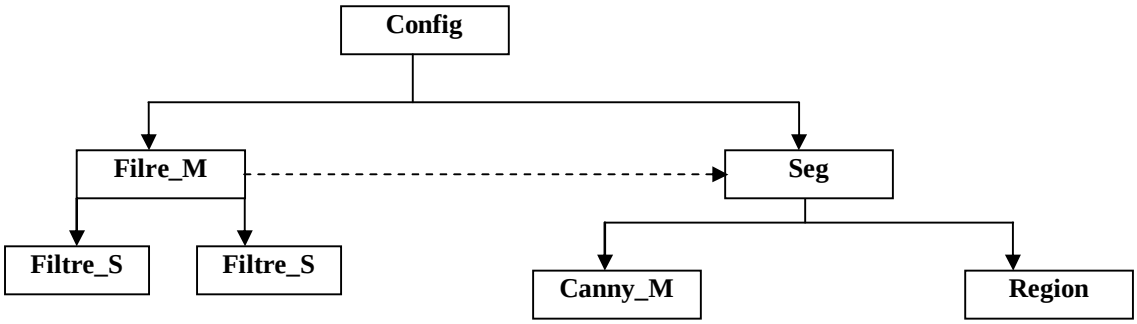


Figure 5-5 : Filtrage seulement

Dans ce cas l'image traitée n'a pas besoin ni de Recadrage dynamique ni de rehaussement de contraste, alors l'agent Config a créé seulement les deux agents Filtre_M et Seg.

4 cas :

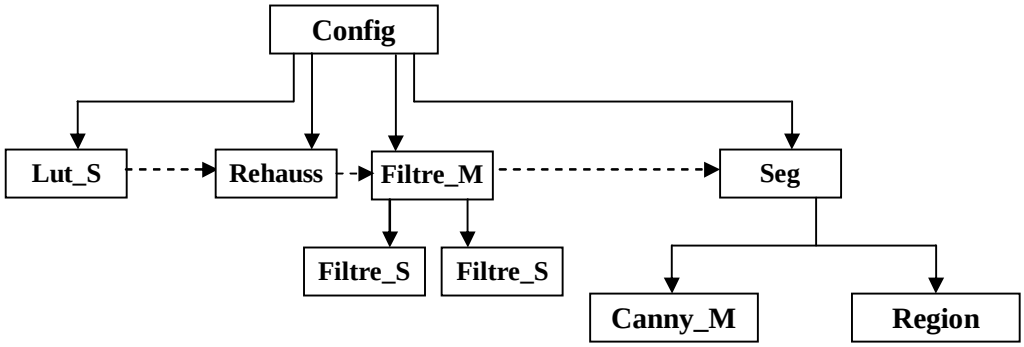


Figure 5-6 : Recadrage dynamique, rehaussement de contraste et Filtrage

L'image à traiter a besoin de recadrage dynamique et de rehaussement de contraste, alors l'agent Config a crée les quatre (04) agents.

5.2.5.2. Fonctionnement global du système

On peut décomposer le système en plusieurs groupes d'agents :

- Un agent pour l'initialisation du système (agent Maitre), il récupère l'image à traiter ainsi que les paramètres entrés par l'utilisateur et lance les agents estimateurs.
- Un groupe d'agents pour le calcule des paramètres de prétraitement (agents : Bruit, Lut_M, Texm et Texts). Ils sont considérés comme des agents de traitement simple.

  Remarque : Le travail des agents de ce groupe s'effectue en parallèle et les agents Texts s'exécutent en parallèle aussi.
- Un groupe d'agents pour le prétraitement (agent Lut_S, agent Rehauss, agent Filtre_M, agents Filtre_S). Ce sont des agents qui prennent des décisions selon les paramètres qu'ils reçoivent en entrée.

  Remarque : Le travail des agents Filtres_S s'effectue en parallèle aussi où chacun d'eux travail sur une portion de l'image.
- Un groupe d'agents pour la segmentation de l'image par la méthode coopérative contours-régions et régions-contours, ce groupe est divisé en trois sous groupes d'agents s'exécutant en parallèle
  - § Groupe d'agents contours pour la segmentation contours (Canny_M, Canny_S et Suprim).
  - § Groupe d'agents Var qui s'exécutent en parallèle pour le calcule de l'écart-type de chaque portion de l'image.

§ Un agent Region pour la segmentation de l'image en région.

- Un groupe d'agents pour le contrôle du système : Control0 pour le bloc de d'estimation, Control1 pour le bloc de prétraitement et Control2 pour le bloc de segmentation.
- Des agents intermédiaires entres les blocs du système : agent Config qui récupère les résultats des estimateurs et lance les agents de prétraitement les plus adéquats et agent Seg qui récupère l'image après le prétraitement et lance les agents segmentation.

5.2.5.3. Fonctionnement des agents

Dans ce qui suit, nous présenterons plus en détail le fonctionnement de chaque agent.

Agent Maitre

- Lecture de l'image et ces paramètres.
- Lecture des paramètres de segmentation.
- Création des agents Config, Bruit, Lut_M, Texm et Control0.
- Envoi des tids (identificateurs) des agents (agent Bruit, agent Lut_M, agent Texm et agent Config) au Control0.
- Envoi de l'image au groupe d'agents (Lut_M, Config et Texm) avec ces paramètres.
- Envoi des paramètres de la segmentation à l'agent config.
- Envoi de l'identificateur de l'agent Control0 à touts les autres agents.
- Envoi des identificateurs des agents (Bruit, Lut_M et Texm) à l'agent Config.
- Réception de confirmation du bon déroulement des agents (agent Bruit, agent Lut_M, agent Texm) de la part de Control0.

  Si (erreur)

  Alors

  Relancer le système.
- Réception des images résultats (région, contours fermés) de l'agent Region à la fin du traitement.

Agent Control0

- Réception des tids des agents à contrôler de la part du Maitre.
- Attente et réception de confirmation des agents sous contrôle.

  Si (pas d 'erreur)

  Alors

  Envoi de confirmation de bon fonctionnement au Maitre.

  Sinon (/*il y a une erreur*/)

  Alors

Envoi un message d ‘erreur au à l’agent Maitre.

Agent Config

- Réception de l’image de l’agent Maitre.
- Réception des paramètres de segmentation de l’agent Maitre.
- Réception du résultat de l’estimation du contraste de l’agent Texm.
- Réception du résultat de l’estimation de la texture de l’agent Bruit.
- Réception du résultat de l’estimation de la luminosité de l’agent Lut_M.
- ***Si*** (recadrage dynamique et filtrage)

  Alors

  - Création de l’agent Seg.
  - Envoi des paramètres de la segmentation à cet agent.
  - Création de l’agent Filtre_M.
  - Création de l’agent Lut_S.
  - Envoi l’identificateur de l’agent Filtre_M à cet agent.
  - Envoi de l’image à cet agent.
  - Envoi le type de filtre à appliquer à l’agent Flitre_M.
  - Envoi l’identificateur de l’agent Seg à l’agent Filtre_M.
- ***Si*** (il y a Rehaussement et filtrage)

  Alors

  - Création de l’agent Seg.
  - Envoi des paramètres de la segmentation à cet agent.
  - Création de l’agent filtre_M.
  - Création de l’agent Rehauss.
  - Envoi l’identificateur de l’agent Filtre_M à cet agent.
  - Envoi de l’image à cet agent.
  - Envoi le type de filtre à appliquer à l’agent Flitre_M.
  - Envoi l’identificateur de l’agent Seg à l’agent Filtre_M.
- ***Si*** (il y a filtrage seulement)

  Alors

  - Création de l’agent Seg.
  - Envoi des paramètres de la segmentation à cet agent.
  - Création de l’agent Filtre_M.
  - Envoi de l’image à cet agent.

- Envoi le type de filtre à appliquer à cet agent.
- Envoi l'identificateur de l'agent Seg à l'agent Filtre_M.

- ***Si*** (il y a Rehaussement, recadrage et filtrage)

  Alors

  - Création de l'agent Seg.
  - Envoi des paramètres de la segmentation à cet agent.
  - Création des agents Lut_S, Rehauss et Filtre_M.
  - Envoi de l'image à l'agent Lut_M.
  - Envoi l'identificateur de l'agent Rehauss à cet agent.
  - Envoi l'identificateur de l'agent Filtre_M à l'agent Rehauss.
  - Envoi le type de filtre à appliquer à l'agent Filtre_M.
  - Envoi l'identificateur de l'agent Seg à l'agent Filtre_M.

Agent Bruit

- Réception de la valeur de texture de la part de l'agent Texm.
- Estimation de la valeur de texture
- Envoi le résultat de l'estimation à l'agent Config.

Si (pas d'erreur)

Alors

Envoi au Control0 la confirmation qu'il n'y a pas d'erreur.

Sinon

Envoi au Control0 un message d'erreur.

Agent Lut_M

- Réception de l'image de l'agent Maitre.
- Estimation de la luminosité de l'image.
- Envoi le résultat de l'estimation à l'agent Config.

Si (pas d'erreur)

Alors

Envoi au Control0 la confirmation qu'il n'y a pas d'erreur.

Sinon

Envoi au Control0 un message d'erreur.

Agent Texm

- Réception de l'image de l'agent Maitre.
- Création des quatre agents Texts.

- Envoi de l'image à ces agents ainsi que les directions de travail.
- Attente et réception de confirmation des agents Texts.

Si (pas d'erreur)

Alors

Continue.

Sinon

Relancer les agents Texts (maximum deux fois).

- Réception des résultats des Texts.
- Estimation de la valeur du contraste.
- Envoi le résultat de l'estimation à l'agent Config.
- Envoi la valeur de la texture à l'agent Bruit.

Si (pas d'erreur)

Alors

Envoi au Control0 la confirmation qu'il n'y a pas d'erreur.

Sinon

Envoi au Control0 un message d'erreur.

Agent Texts(i) (i = 0 à 3)

- Réception de l'image de l'agent Texm.
- Réception de la direction de calcule de la part de l'agent Texm.
- Calcul de la matrice de cooccurrence suivant la direction donnée.
- Calcul de la valeur du contraste et de la texture.
- Envoi les résultats à l'agent Texm.

Si (pas d'erreur)

Alors

Envoi au Texm la confirmation qu'il n'y a pas d'erreur.

Sinon

Envoi au Texm un message d'erreur.

Agent Rehauss

- Réception de l'identificateur de l'agent Filtre_M de la part de l'agent Config.

Si (il y a recadrage dynamique)

Alors

Réception de l'image de la part de l'agent Lut_S.

Sinon

Réception de l'image de la part de l'agent Config.

- Appliquer le rehaussement de contraste sur l'image.

Si (pas d'erreur)

Alors

Envoi au Control1 la confirmation qu'il n'y a pas d'erreur.

Sinon

Envoi au Control1 un message d'erreur.

Agent Lut_S

- Réception de l'image de l'agent Config.
- Réception de confirmation s'il y a rehaussement de contraste ou non de la part de l'agent Config.
- Appliquer le recadrage dynamique sur l'image.
- Envoi de l'image traitée soit à l'agent Filtre_M soit à l'agent Rehauss.

Si (pas d'erreur)

Alors

Envoi au Control1 la confirmation qu'il n'y a pas d'erreur.

Sinon

Envoi au Control1 un message d'erreur.

Agent Filtre_M

- Réception de choix du type de filtre à appliquer.
- Réception de l'image soit de la part de l'agent Lut_S, Rehauss ou Config.
- Création des agents Filtre_S.
- Envoi les portions, de l'image, encadrées aux agents Filtre_S.
- Envoi le type de filtre à appliquer aux Filtre_S.
- Attente et réception de confirmation des agents Filtres_S.

  Si (pas d'erreur)

  Alors

  Continue.

  Sinon

  Relancer les agents Filtre_S (maximum deux fois).
- Réception des blocs d'image filtrés des agents Filtre_S.

  Si (le filtre appliqué est celui de Nagao)

  Alors

- § Réception de l'écart-type de chaque portion de l'image.
- § Création du tableau des écart-types de l'image.
- § Envoi de ce tableau à l'agent Seg.

- Reconstitution de l'image.
- Envoi de l'image filtrée à l'agent Seg.

Si (pas d'erreur)

Alors

Envoi au Control1 la confirmation qu'il n'y a pas d'erreur.

Sinon

Envoi au Control1 un message d'erreur.

Agent Filtre_S(i) (i = 0 à n-1 /n<=81)

- Réception du choix de filtre à appliquer de l'agent Filtre_M.
- Réception d'un bloc de l'image de la part de l'agent Filtre_M.
- Filtrage de la portion d'image reçu avec le type de filtre choisis.
- Calcul de l'écart type de la portion de l'image si le filtre appliqué est celui de Nagao.

Si (pas d'erreur)

Alors Envoi au Filtre_M la confirmation qu'il n'y a pas d'erreur.

Sinon

Envoi au Filtre_M un message d'erreur.

- Envoi les résultats à Canny_M

Agent Control1

- Réception des tids (identificateurs) des agents à contrôler de la part du Config.
- Attente et réception de confirmation des agents sous contrôle.

Si (pas d 'erreur)

Alors

Envoi de confirmation de bon fonctionnement au Config.

Sinon (/*il y a une erreur*/)

Alors

Envoi un message d 'erreur au Config.

Agent Seg

- Réception des paramètres de segmentation de l'agent Config.
- Réception de l'image à traiter de la part de l'agent Filtre_M.

Si (il y avait le filtrage par le filtre de Nagao)

Alors

- § Réception du tableau des écart-types de l'image de la part de l'agent Flitre_M.

Sinon

- § Création des agents Var.
- § Envoi des blocs encadrés de l'image aux agents Var.
- § Attente et réception de confirmation des agents Var.

Si (pas d'erreur)

Alors

Continue.

Sinon

Relancer les agents Var (maximum deux fois).

- § Réception de l'écart-type de chaque bloc.
- § Construction du tableau des écart-types générale.

- Création de l'agent Canny_M.
- Envoi de l'image à cet agent.
- Envoi des paramètres de la segmentation à cet agent.
- Création de l'agent Region.
- Envoi de l'image à cet agent.
- Envoi du tableau des écart-types à cet agent.
- Création de l'agent Control2.
- Envoi des tids des agents (Canny_M, Region) au Control2.

Agent Var(i) (i = 0 à n-1 /n<=81)

- Réception d'une portion de l'image de la part de l'agent Seg.
- Calcul de l'écart-type du bloc reçu.

Si (pas d'erreur)

Alors

Envoi de confirmation au Seg.

Sinon

Envoi un message d'erreur au Seg.

- Envoi la valeur d'écart-type calculée à l'agent Seg.

Agent Canny_M

- Réception de l'image de l'agent Seg.
- Réception des paramètres de segmentation en contours de l'agent Seg.
- Création des agents Canny_S.
- Envoi des paramètres de segmentation en contours aux agents Canny_S.
- Envoi des portions, de l'image, encadrées aux agents Canny_S..
- Attente et réception de confirmation des agents Canny_S.

  Si (pas d'erreur)

  Alors

  Continue.

  Sinon

  Relancer les agents Canny_S (maximum deux fois).
- Réception des blocs d'image segmentés des agents Canny_S.
- Construction de la carte contour initial de l'image.
- Création de l'agent Suprim.
- Attente et réception de confirmation de l'agent Suprim.

  Si (pas d'erreur)

  Alors

  Continue.

  Sinon

  Relancer l'agent Suprim.
- Envoi de la carte contour à cet agent.

  Si (pas d'erreur)

  Alors

  Envoi de confirmation au Control2.

  Sinon

  Envoi message d'erreur au Control2.

Agent Canny_S(i) (i = 0 à n-1 /n<=81)

- Réception d'une portion (bloc) de l'image de la part de l'agent Canny_M.
- Réception des paramètres de segmentation en contours de l'agent Canny_M.
- Création de la carte contour par l'opérateur de Canny.

  Si (pas d'erreur)

  Alors

  Envoi de confirmation au Canny_M.

Sinon

Envoi un message d'erreur au Canny_M.

- Envoi de la carte contour du bloc à l'agent Canny_M.

Agent Suprim

- Réception de l'image de l'agent Canny_M.
- Raccordement des chaînes de contour, proche entre eux, qui ont le même alignement.
- Elimination des petites chaînes dans la carte contours.
- Elimination des points parasites qui se situent sur les contours.

Si (pas d'erreur)

Alors

Envoi de confirmation au Canny_M.

Sinon

Envoi un message d'erreur au Canny_M.

- Envoi de la carte contour à l'agent Region.

Agent Region

- Réception de l'image de l'agent Seg.
- Réception du tableau générale des écart-types de l'image de la part de l'agent Seg.
- Réception de la carte contour de l'agent Suprim.
- Création de la carte région par la méthode « croissance des régions ».
- Fusion des petites régions aux régions adjacentes les plus adéquates.
- Fusion des régions qui ont des moyennes en niveau de gris proche.
- Fermeture des contours.
- Envoi de la carte régions à l'agent Maitre.
- Envoi de la carte contours à l'agent Maitre.

Si (pas d'erreur)

Alors

Envoi de confirmation au Control2.

Sinon

Envoi un message d'erreur au Cotrol2.

Agent Control2

- Réception des tids des agents à contrôler de la part du Seg.
- Attente et réception de confirmation des agents sous contrôle.

Si (pas d ‘erreur)

Alors

Envoi de confirmation de bon fonctionnement au Seg.

Sinon (/*il y a une erreur*/)

Alors

Envoi un message d ‘erreur au Seg.

5.2.6. Modélisation Orientée Objet du système

Un langage orienté objet apporte une solution satisfaisante aux différents problèmes rencontrés dans la programmation procédurale classique. La taille des programmes et les problèmes de maintenance ont mis en évidence la nécessité de promouvoir l’abstraction des données, la modularité et la réutilisation des logiciels.

La modélisation orientée objet de ce système est basée sur les traitements où chaque agent représente un maillon dans une chaîne de traitement effectuées lors de l’analyse d’images. Comme les agents ont certaines connaissances et primitives communes, la création d’une **Class** générique regroupant ces primitives et connaissances s’est avérée nécessaire.

Dans ce contexte la Class (CAgent) regroupe un ensemble de données communes entre les agents qui représentent les connaissances sur soi même (pour chaque agent) et une partie de l’environnement.

- Les connaissances sur soi même sont :
  - ***Mytid*** : l’identificateur attribué par PVM à l’agent, utilisé le long du programme pour l’identifier.
  - ***Name*** : contient le nom de l’agent.
- La partie sur l’environnement :
  - ***Parent*** : identifie l’agent père de cet agent.
- Les fonctions communes :
  - ***SendD(****paramètres****)*** : Elle est utilisée pour l’envoi d’une donnée de type entier.
  - ***SendI(****paramètres****)*** : Elle est utilisée pour l’envoi d’une donnée de type image.
  - ***SendMsg(****paramètres****)*** : Elle est utilisée pour l’envoi d’une donnée de type chaîne de caractères .
  - ***RecievD(****paramètres****)*** : Elle est utilisée pour la réception d’une donnée de type entier.
  - ***RecievI(****paramètres****)*** : Elle est utilisée pour la réception d’une donnée de type image.

- ***RecievMsg(paramètres)*** : Elle est utilisée pour la réception d'une donnée de type chaîne de caractères .

Structure de la class CAgent

```
Class CAgent
{
    public :
    //les connaissances sur soit de l'agent
    int Mytid ;
    char *Name ;
    //La partie sur l'environnement
     int Parent ;
     // Les fonctions communes entre les agents
     SendD(int D, int Mytid, int Tidemmis) ;
     SendI(Image ima.data, int Mytid, int Tidemmis) ;
     SendMsg(char *Msg, int Mytid, int Tidemmis) ;
     RecievD(int D, int Mytid, int Tidemmis, timeval *tmout) ;
     RecievI(Image ima.data, int Mytid, int Tidemmis, timeval *tmout) ;
     RecievMsg(char *Msg, int Mytid, int Tidemmis, , timeval *tmout) ;
} ;
```

D'autre part, comme les traitements (expertise de l'agent) différent d'un agent à un autre et comme chaque agent possède une connaissance partielle de son environnement, chaque agent du système est modélisé comme une instance d'une classe regroupant ces derniers ; en plus de ce qu'elle a hérité de la classe générique (CAgent) décrite précédemment.

Chaque classe d'un agent spécialisé regroupe :

- Des connaissances partielles sur l'environnement : représente soit les images, soit des données nécessaires au traitement qu'il reçoit, soit les identificateurs des autres agents avec lesquelles ils doit communiquer.
- Fonction d'expertise (traitement) basée sur les compétences et la spécialité de chaque agent, et les fonctions de données spécialisées.
- Les variables et les méthodes membres de la class varient d'une classe agent à un autre en fonction du rôle que joue l'agent dans le système.

***Exemple de classe d'un agent (Filtre_S)* :**

```
Class Filtre_S : public CAgent
{       public :    //l'expertise de l'agent
        void Gauss(paramètres) ;
        void Moy(paramètres) ;
        void Nagao(paramètres) ;
} ;
```

5.2.7. Communication entre les agents

La communication entre les agents est sélective (chaque agent communique avec des agents bien précis) non sollicitée (un agent ne peut pas demander à un autre agent de lui fournir une information) sans accusé de réception et avec transmission simple (pas de répétitions d'envoi d'information dans le cas ou y a pas d'erreur) selon le modèle de communication par envoi de messages qui est supporté par PVM.

On peut distinguer deux types de messages :

- Les messages nécessaires pour le fonctionnement des agents : l'envoi d'une image, les paramètres d'estimations (luminosité, contraste, texture)…
- Les messages de création des agents : l'envoi des identificateurs des agents pour qu'ils puissent communiquer entre eux.

Les méthodes implantées pour la communication sont basées sur l'utilisation des primitives de base d'envoi et de réception de PVM.

L'envoi et la réception des données se fait via les six fonctions définies dans l'agent CAgent dont tous les agents sont hérités. En plus de ces fonctions on cite deux autres fonctions pour l'envoi et la réception des données réels :

  - ***SendIf(****float d, int Mytid, int tidemis****)***
  - ***ReceivIf(****float d , int Mytid, int tidemis,timeval *tmout****)***
- Les fonctions de l'envoi des données ont la structure suivante :
  - Initialisation du buffer du message à envoyer.
  - Compression du message dans le buffer.
  - Envoi du message.
- Les fonctions de réception des données ont la structure suivante :
  - Initialisation du buffer de réception.
  - Réception des données dans le buffer.
  - Décompression du message dans le buffer.

- Comme on peut distinguer deux types de communication :
    - Directe c'est à dire que l'émetteur sait dés le début l'identificateur de récepteur et vis versa.
    - Entre groupe c'est à dire que l'émetteur doit savoir le nom du groupe à envoyer.

Pour bien comprendre voir la figure ci-dessous :

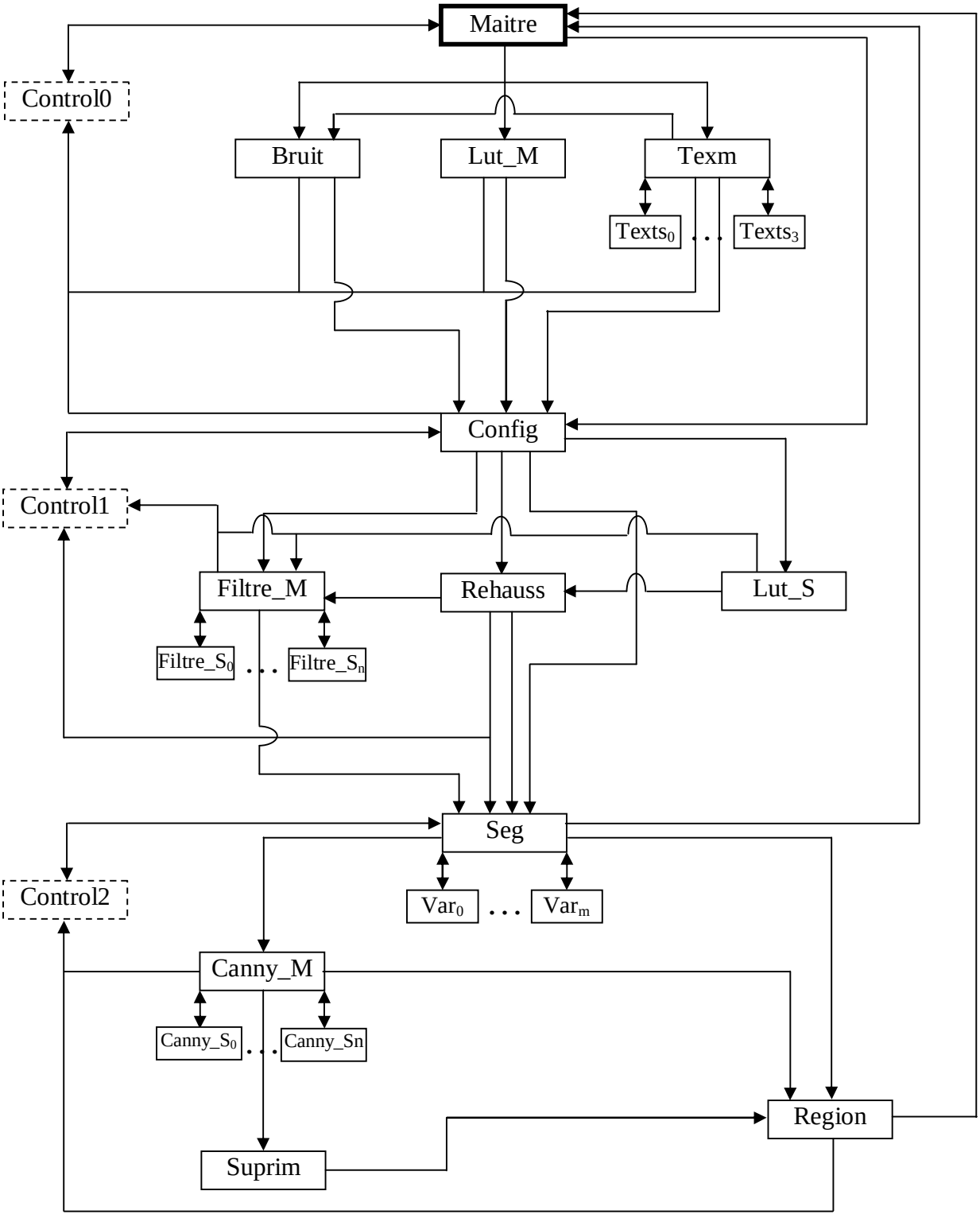


Figure 5-7 : Schéma de communication entre agents

5.2.8. Coopération entre les agents

Dans notre système la coopération est très importante car nous avons repris une méthode coopérative de traitement d'images par l'approche multi-agents. Cette coopération se fait par partage des résultats entre les agents coopérant. Et ce voit renforcé entre les agents Canny_M, Suprim, Region, Seg où chaque agent coopère avec les autres pour aboutir à son but de manière à avoir un meilleur résultat possible.

L'agent Canny_M crée la carte contours initial, ensuite il va la transmettre à l'agent Suprim, ce dernier améliore les contours en éliminant les petites chaînes et les points parasites sur les contours, et en raccordant les chaînes de contours qui paraissent être découper. Il envoi l'image résultat à l'agent Region, ce dernier utilise cette carte pour diriger la détection des régions et cela après la récupération du tableau générale des écarts type de chaque portion de l'image de la part de l'agent Seg, en suite il transmet les deux cartes, contours et régions, à l'agent Maitre après fusionnement des petites régions et des régions mal segmentées et aussi après la fermeture des contours.

On peut aussi observer la coopération au niveau d'autres agents, par exemple :

- Entre l'agent Texm et l'agent Bruit pour Estimer la texture.
- Entre l'agent Texm et les agents Texts pour estimer la valeur du contraste.
- Entre l'agent Filtre_M et les agents Filtre_S pour créer la carte contour initiale…

5.2.9. Contrôle des agents

Dans notre système le contrôle est très important, son rôle est de contrôler les agents pour éviter des problèmes qui peuvent l'arrêter (blocages). Parmi les problèmes que nous avons rencontré durant la réalisation de notre système :

- Erreur de création des agents : cette erreur apparaît lorsqu'un agent « Maître » crée un autre agent « esclave » mais ce dernier n'apparaît pas ; cette erreur se produit en générale lorsque l'agent « esclave » n'existe pas ou lorsque son nom est mal écrit dans le programme source de l'agent « Maître ».
- Erreur d'allocation de mémoire : c'est lorsque plusieurs agents veulent allouer au même temps de l'espace mémoire (conflit entre les agents),

c’est aussi le problème d’insuffisance en mémoire si l’image traitée est très grande.

- Erreur de création, d’ouverture et de fermeture des fichiers : ce type d’erreur se produit lorsque le fichier n’existe pas ou lorsqu’il est utilisé par un autre agent.
- Erreur de communication entres les agents : c’est le cas ou l’agent ne peut pas envoyer ou recevoir des données. Ces problèmes sont dus à l’occupation du tampon du « PVM » ou à la différence des identificateurs (tid agent et tid message) chez les agents communicant.

Pour pallier ces problèmes, nous avons implémenté une stratégie de contrôle basée sur le contrôle individuel et contrôle social. Chaque agent contrôle l’état de la création, d’ouverture et de la fermeture des fichiers ainsi l’état de la création des agents. S’il rencontre un problème il essaie de refaire l’opération maximum deux fois.

Pour le problème d’allocation mémoire, nous l’avons géré selon les besoin, par exemple : pour un entier inférieur à 256 (c’est le cas des niveaux de gris dans une image) nous utilisons le type ***unsigned char*** (1 octet = 8bits) à la place du type ***int*** (entier à 16bits = 2 octets), et dans le cas ou l’opération d’allocation échoue l’agent répète cette opération deux fois au maximum.

Pour éviter des conflits dans la communication entre les agents, nous utilisons des identificateurs d’agents similaires à leurs noms, par exemple : Config → tidcgf, Seg → tidseg…de plus, l’agent récupère le résultat de son émission, s’il est négatif il répète l’opération (deux fois maximum). Pour la réception, l’agent attend un certain temps, si rien n’arrive il déclare une erreur.

Si l’agent n’arrive pas à résoudre les problèmes rencontrés, alors il envoie une erreur à l’agent qui se charge du contrôle de son bloc, dans ce cas nous utilisons le contrôle social, plus précisément le contrôle centralisé. Pour cela nous avons créé trois (03) agents (Control0, Control1, Control2) pour contrôler les trois (03) blocs du système (Estimation, Prétraitement et Segmentation), chacun de ces agents reçoit des messages de la part des agents sous son contrôle, si se sont des messages d’erreur alors il demande à l’agent qui gère le bloc (l’agent créateur du bloc) de le relancer.

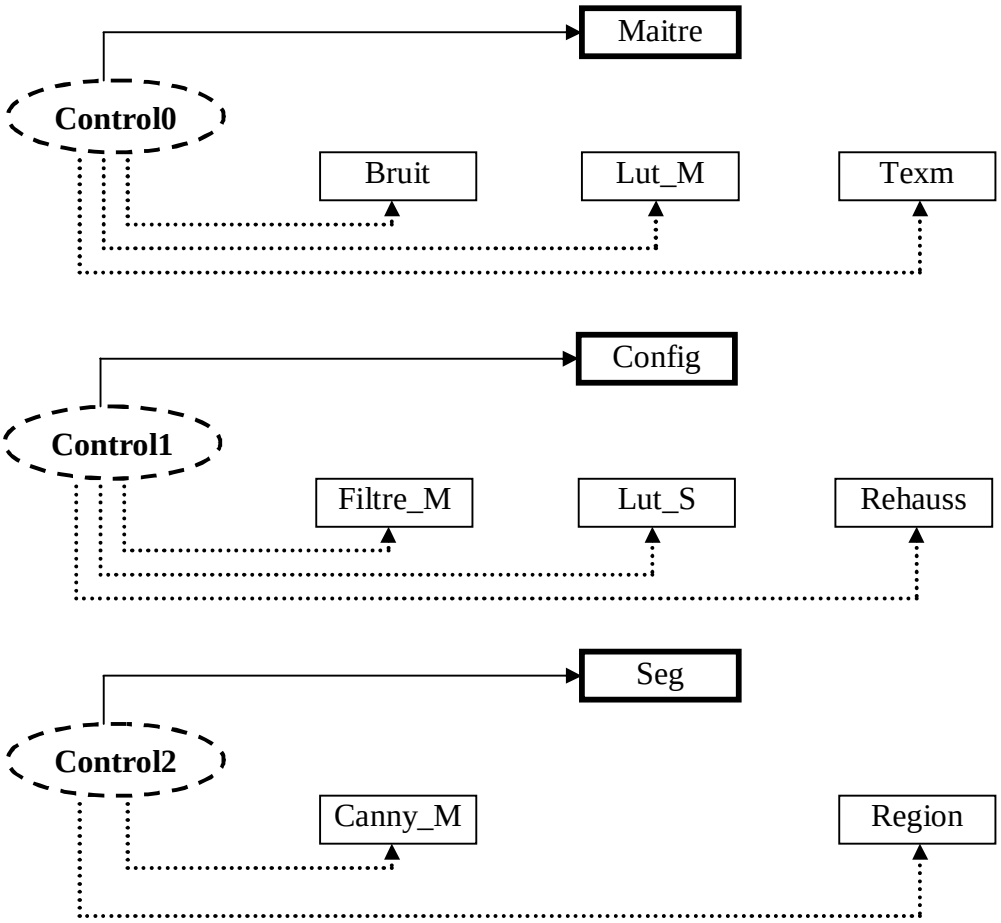


Figure 5-8 : Le control centralisé du système multi-agents

Au cours des essais de notre système, nous avons remarqué que le contrôle centralisé est insuffisant, pour cela, nous avons aussi introduit le contrôle décentralisé (distribué) en distribuant la tâche du contrôle sur un ensemble d'agents qui crée beaucoup d'agents slaves ; c'est le cas des agents Filtre_M, Texm, Canny_M et Seg.

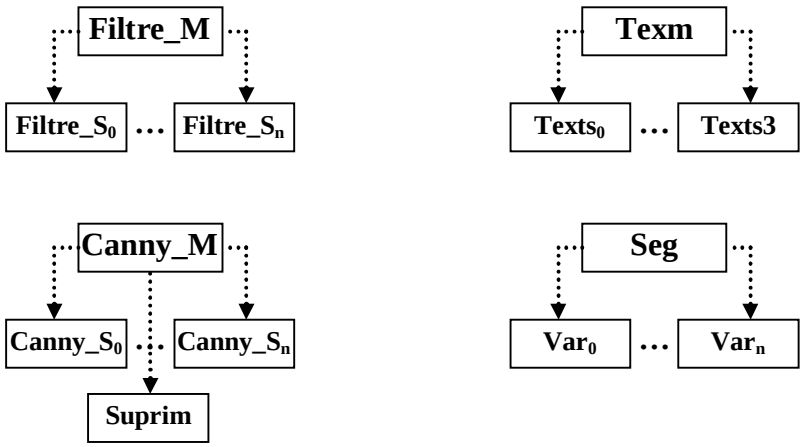


Figure 5-9 : Le control décentralisé du système multi-agents

5.2.10. Implantation logicielle du système

5.2.10.1. Environnement de programmation

Le système est conçu pour être utilisé sur n'importe quel environnement, tel que le système d'exploitation Windows ou Unix.

5.2.10.2. Langage de programmation

Afin de créer un système facile à maintenir et à réutiliser, nous avons opté pour le modéliser à l'aide d'un langage orienté objet et puisque les primitives de PVM sont programmées à l'aide du langage « C », nous avons opté pour le « C++ ».

Le grand succès du langage « C » s'explique par les avantages suivants ; C'est un langage :

1) **universel :** « C » n'est pas orienté vers un domaine d'applications spéciales, comme par exemple FORTRAN (applications scientifiques et techniques) ou COBOL (applications commerciales).
2) **compact :** « C » est basé sur un noyau de fonctions et d'opérateurs limité, qui permet la formulation d'expressions simples, mais efficaces.
3) **moderne :** « C » est un langage structuré, déclaratif et récursif; il offre des structures de contrôle et de déclaration comparables à celles des autres grands langages de ce temps (FORTRAN, ALGOL68, PASCAL).
4) **près de la machine :** comme « C » a été développé en premier lieu pour programmer le système d'exploitation UNIX, il offre des opérateurs qui sont très proches de ceux du langage machine et des fonctions qui permettent un accès simple et direct aux fonctions internes de l'ordinateur (ex: la gestion de la mémoire).
5) **rapide :** comme « C » permet d'utiliser des expressions et des opérateurs qui sont très proches du langage machine, il est possible de développer des programmes efficients et rapides.
6) **indépendant de la machine :** bien que « C » soit un langage près de la machine, il peut être utilisé sur n'importe quel système en possession d'un compilateur « C ». Au début « C » était surtout le langage des systèmes travaillant sous UNIX, aujourd'hui « C » est devenu le langage de programmation standard dans le domaine des micro-ordinateurs.
7) **portable :** en respectant le standard ANSI-C, il est possible d'utiliser le même programme sur tout autre système (autre hardware, autre système d'exploitation), simplement en le recompilant.

8) **extensible :** « C » ne se compose pas seulement des fonctions standard; le langage est animé par des bibliothèques de fonctions privées ou livrées par de nombreuses maisons de développement.

Remarque : Notre système a été testé sous Windows 9x/2k, et compilé avec Visual C++6.0. Une application utilisant la bibliothèque MFC présente des conflits avec la librairie libpvm3.lib de PVM .Ainsi il n'est pas possible de créer une interface graphique utilisant MFC, et supportant les primitives du PVM.

5.2.10.3. Matériel utilisé

A plusieurs stades du traitement, le système fonctionne de manière parallèle, et pour assurer une bonne efficacité, il faut disposer d'un nombre minimal de stations de travail, constituant le réseau PVM.

Nos essais ont été réalisés sur un réseau de micro-ordinateurs à base Windows 9x/2k.

5.2.10.4. Configuration et installation du système

Pour faire fonctionner correctement ce logiciel, un certain nombre de points concernant la configuration de l'environnement de travail doivent être respectés :

- Le PVM doit être installé sur tous les hosts de la machine virtuelle (les différentes machines du réseau).
- Copie des fichiers exécutables du logiciel sur tous les hosts si on ne spécifie pas le hosts ou doit s'exécuter chaque agent du système, dans le cas contraire, il suffit d'installer seulement les fichiers nécessaires.
- Le protocole TCP/IP doit être installé sur tous les hosts.
- Le service RSHD doit être installé sur les hosts .C'est un service qui permet le lancement de commandes de programmes pouvant s'exécuter sur une machine distante.

5.3. Description de l'interface graphique

Nous avons implanté une interface graphique en utilisant le langage de programmation C++Builder5, cet interface possède deux (02) caractéristiques importantes :

a. La possibilité d'ouvrir plusieurs formats d'images (bmp, jpg, wmf, raw…) : pour les formats bitmap (bmp 24bits, 256couleurs…), les formats compressés JPEG (jpeg, jpg) et les formats vectoriels (emf, wmf) C++Builder possède des bibliothèques qui permettent de lire ces formats, mais pour les traiter il faut les convertir en un format Bitmap 24 bits. Pour les images brutes (raw, ima), il faut

tout d'abord les ouvrir en utilisant des instructions du langage « C » de base, ensuite les convertir au format bitmap 24bits.

b. Le lancement et la supervision du système multi-agents : dans cette partie, l'utilisateur ouvre l'image à traiter et clique sur le bouton du lancement du système. En réalité, l'interface lance seulement l'agent « Maitre », ce dernier récupère l'image ouverte par l'interface graphique et lance le système.

   Dans le déroulement du système, l'agent qui lance un slave (un autre agent) crée au même temps un fichier pour indiquer le lancement de cet agent, et quand ce dernier termine sa tâche et avant de quitter le système, il crée un fichier pour indiquer son arrêt. L'interface graphique interprète ces fichiers et indique l'état du système (des agents) par un schéma bloc et par des paragraphes saisis dans une zone de texte.

   La figure 5.10 nous montre l'état du système au milieu du traitement, les agents en claire sont les agents qui ont terminé leurs tâches, ceux en foncé sont en train d'exécuter leurs fonctions et les agents qui n'apparaissent pas (ex : Region…) ne sont pas encore lancés par le système.

Des détailles en plus sur ce logiciel sont représentés dans l'annexe C.

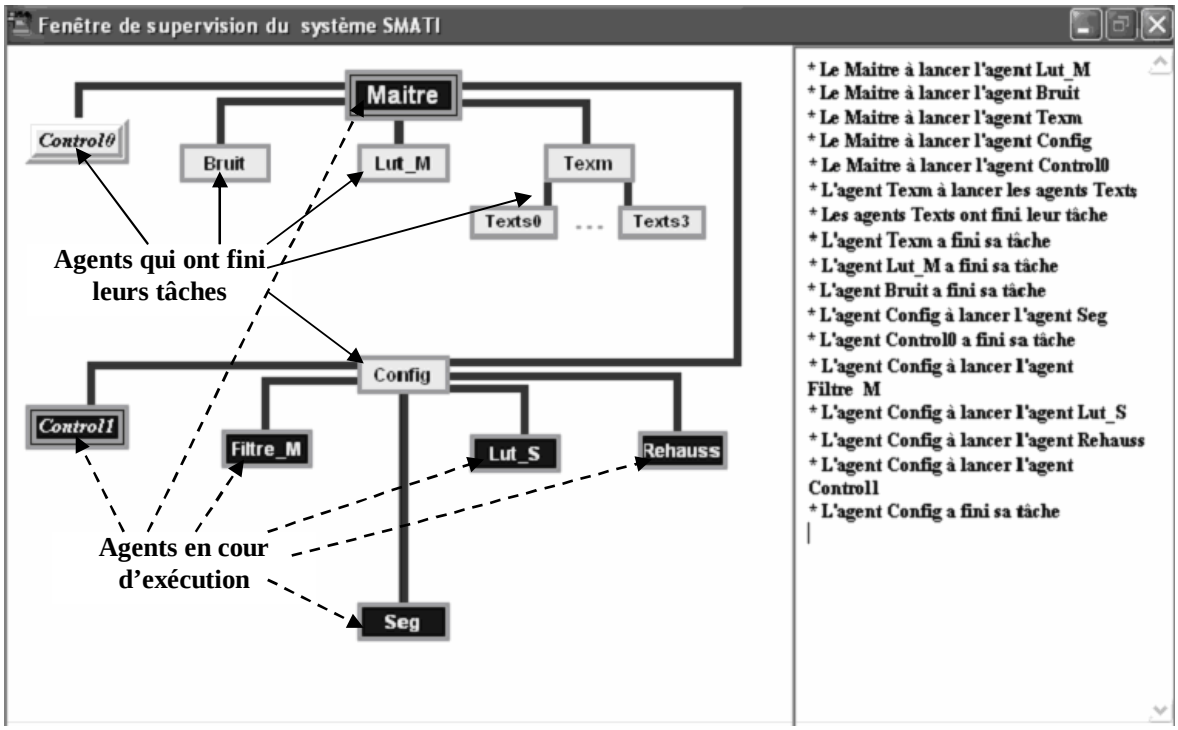


Figure 5-10 : La supervision du système SMATI par l'interface graphique

5.4. Résultats et interprétation

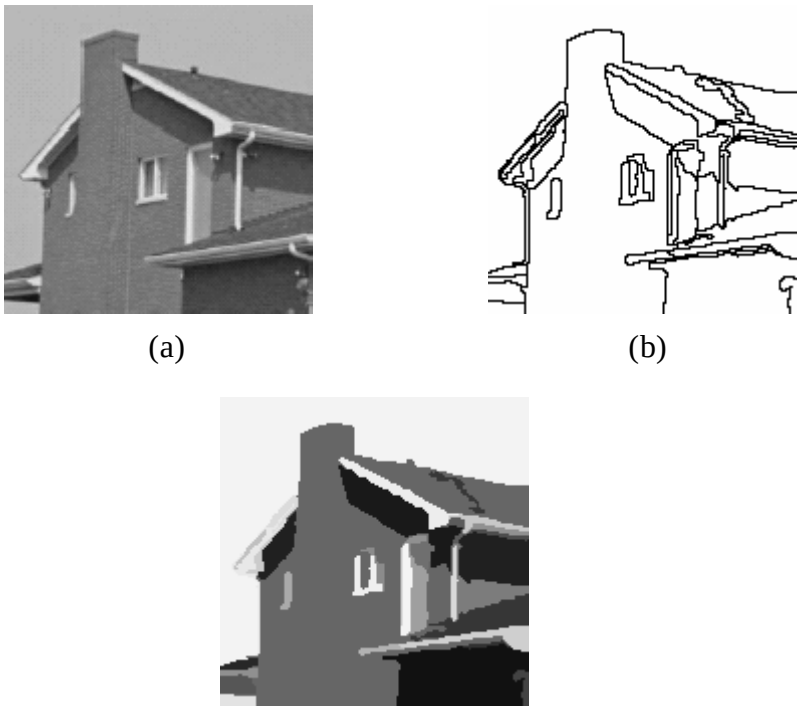
Les tests ont été faits sur un réseau de deux PC Poste01 (PIII Celeron 500Mhz) et Poste02 (PII Celeron 466Mhz) connectés à travers un câble croisé.

En premier lieu, nous présentons quelques images résultats obtenus par le système, ensuite nous montrons le temps d'exécution total (du système) et partiel (pour chaque agent).

5.4.1. Les images

Remarque : nous considérons :

- (a) L'image originale.
- (b) La carte-contours finale.
- (c) La carte-régions finale.
- Sb : le seuil bas du seuillage par hystérésis (le seuil haut est le double).
- Sf : le seuil de fusion des régions adjacentes.
- Tm : taille minimale d'une région.

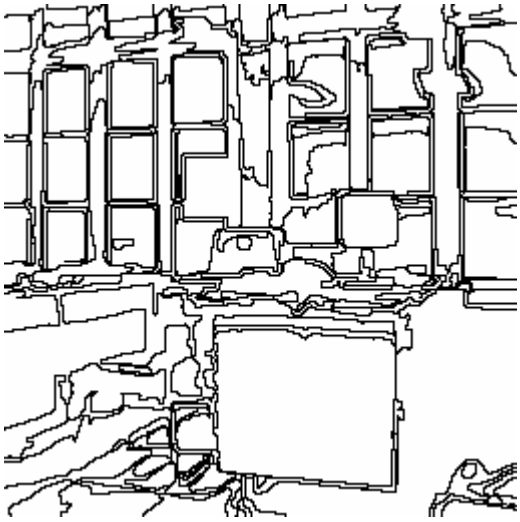


(a) (b)

(c)

Figure 5-11 : La carte région et la carte contour obtenues de l'image Maison **Sb = 3, Sf = 35, Tm = 35.**

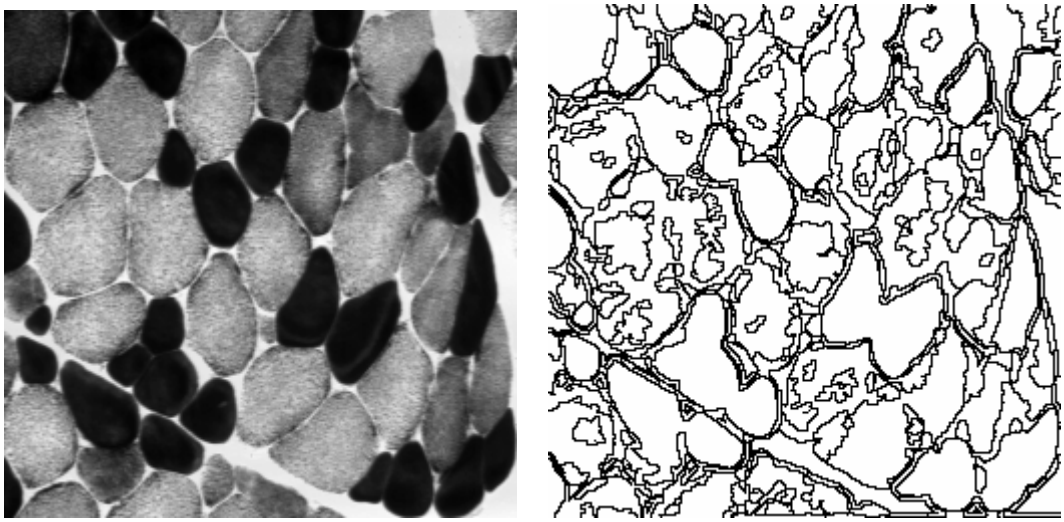
(a)



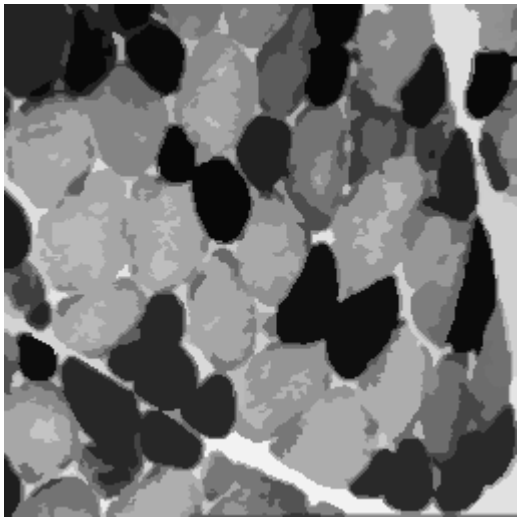
(b)

(c)

Figure 5-12 : La carte région et la carte contour obtenues de l'image Bureau **Sb = 5, Sf = 21, Tm = 21.**



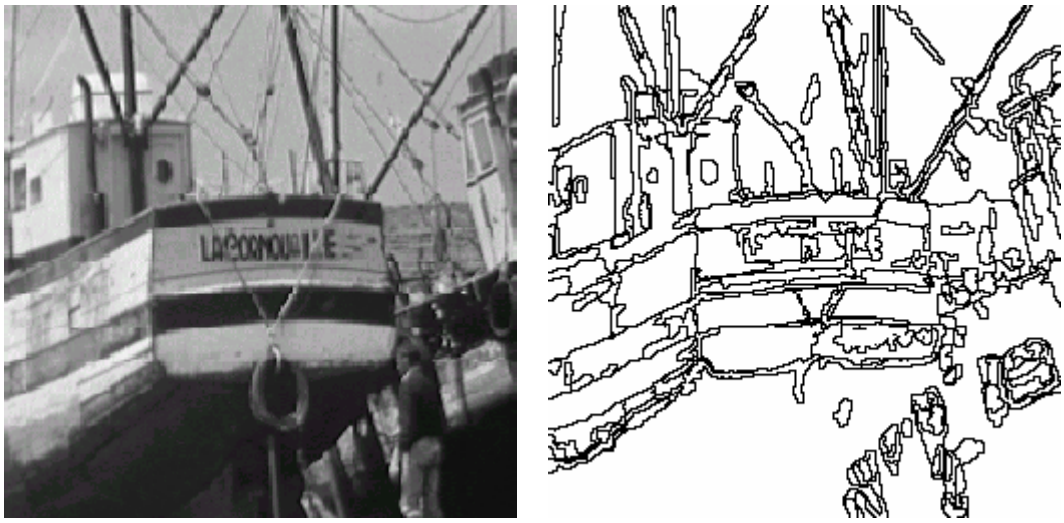
(a) (b)



(c)

Figure 5-13 : La carte région et la carte contour obtenues de l'image Cellules

Sb = 3, Sf = 19, Tm = 11

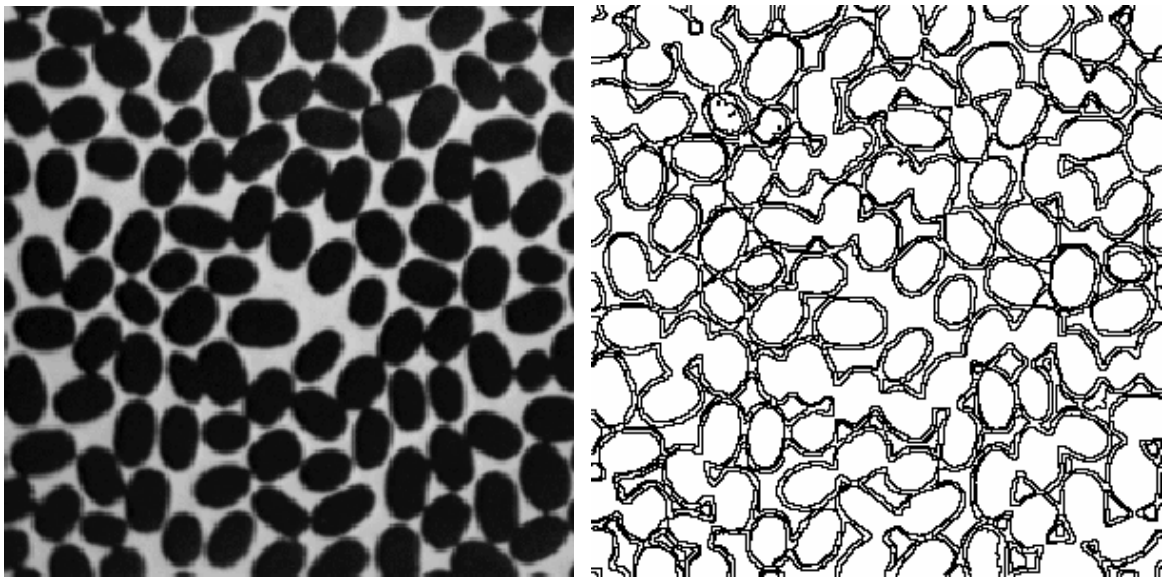


(a) (b)

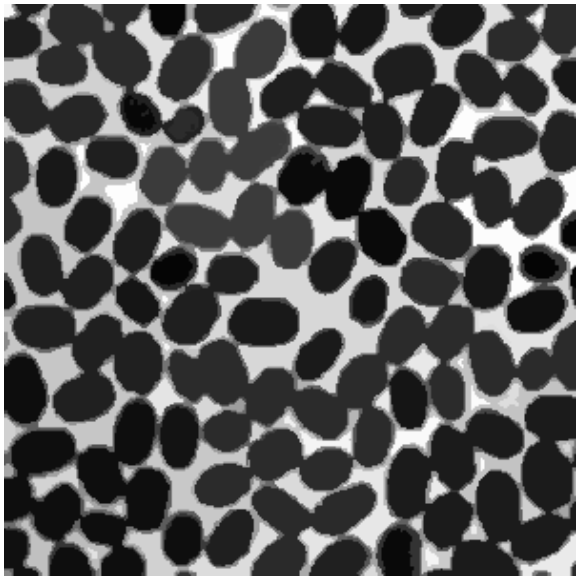
(c)

Figure 5-14 : La carte région et la carte contour obtenues de l'image Bateau

Sb = 3, Sf = 19, Tm = 11

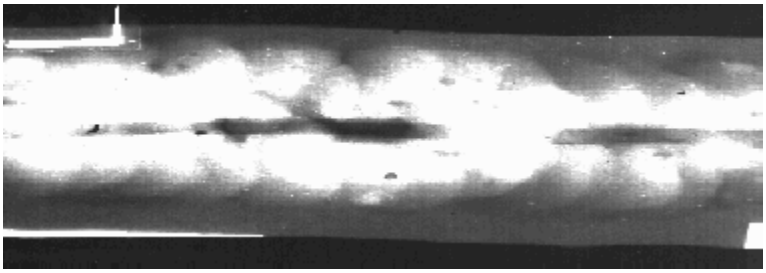


(a) (b)

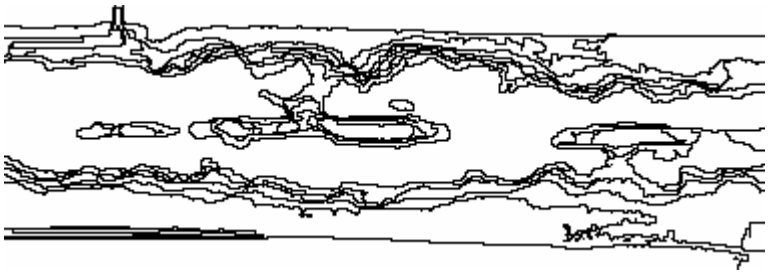


(c)

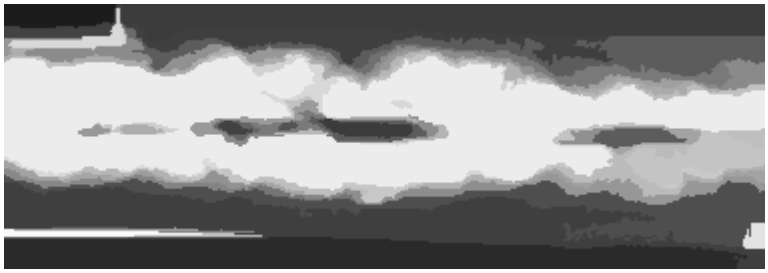
Figure 5-15 : La carte région et la carte contour obtenues de l'image Cellules2 **Sb = 4, Sf = 19, Tm = 11**



(a)



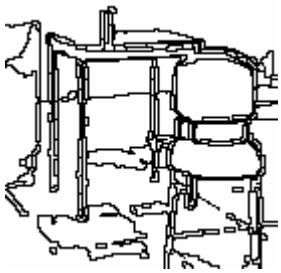
(b)



(c)

Figure 5-16 : La carte région et la carte contour obtenues de l'image Radio

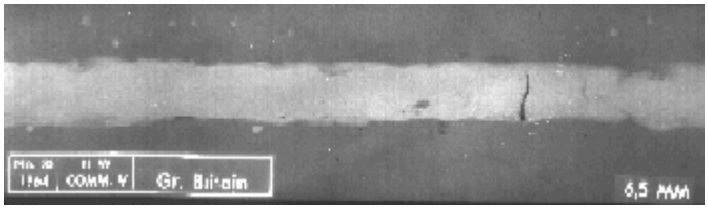
Sb = 5, Sf = 25, Tm = 25



(a) (b) (c)

Figure 5-17 : La carte région et la carte contour obtenues de l'image Chaise

Sb = 3, Sf = 25, Tm = 3



(a)

(b)

(c)

Figure 5-18 : La carte région et la carte contour obtenues de l'image Radio2

Sb = 5, Sf = 33, Tm = 7

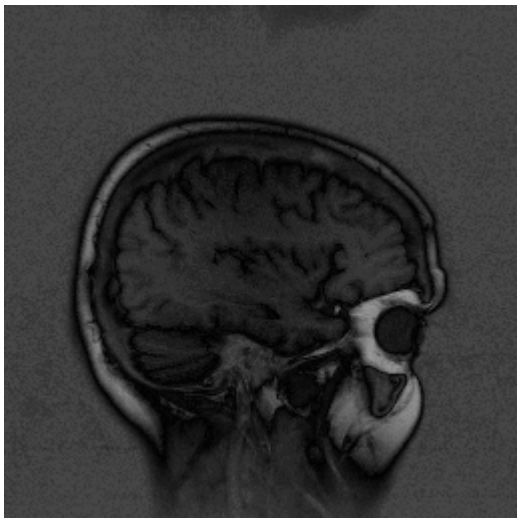
(a)

(b)

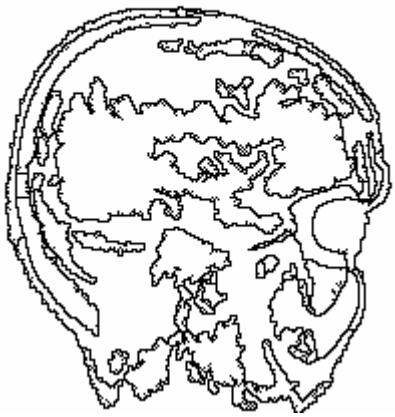
(c)

Figure 5-19 : La carte région et la carte contour obtenues de l'image Chapeau

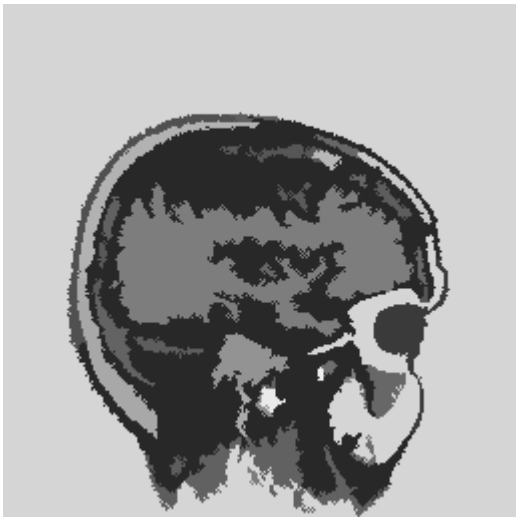
Sb = 2, Sf = 19, Tm = 9



(a)



(b)



(c)

Figure 5-20 : La carte région et la carte contour obtenues de l'image Cerveau

Sb = 9, Sf = 23, Tm = 17

5.4.2. Les tests sous réseau

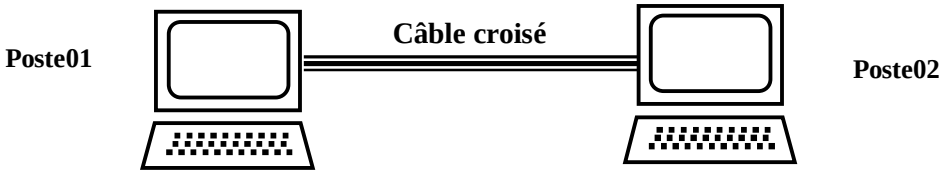


Figure 5-21 : Schéma du réseau de la machine virtuelle

Nous avons choisis trois images de taille différentes pour les testes (voir les figures 5.22 ; 5.23 ; 5.24).

Durant les testes, nous avons, au début, utilisé un seul PC et ensuite les deux PC. Les différents résultats sont exposés dans des tableaux et des histogrammes.

Figure 5-22 : Image teste 1 (Image Nénuphar 132 Ko 450*300)

Figure 5-23 : Image teste 2 (Image Bureau 64 Ko 256*256)

Figure 5-24 : Image teste 3 (Image Maison 24 Ko 154*154)

Les figures suivantes montrent le temps d'exécution de quelques agents pour les trois (03) images de test choisisses.

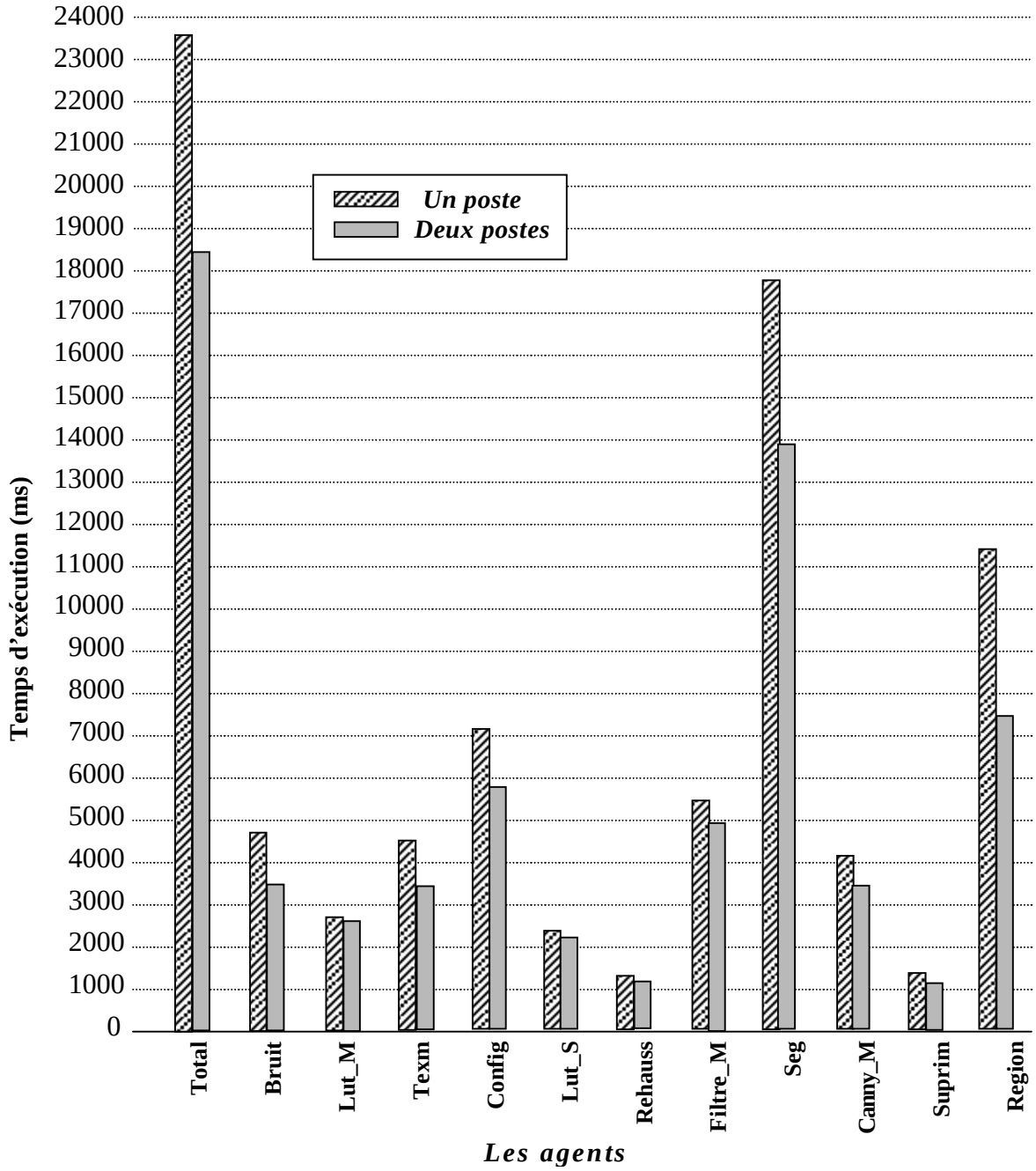


Figure 5-25 : Temps d'exécution de quelques agents pour l'image Nénuphar

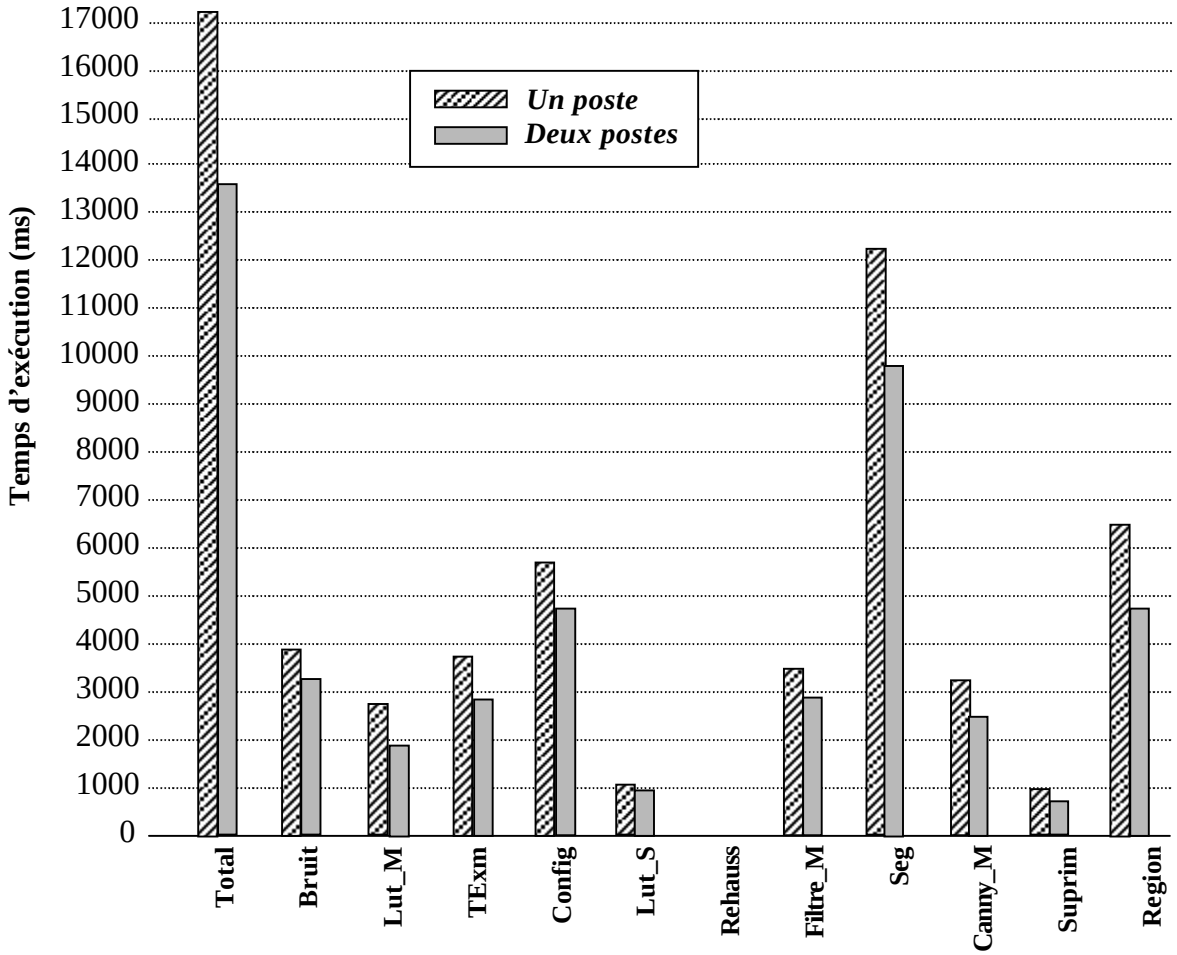


Figure 5-26 : Temps d'exécution de quelques agents pour l'image Bureau

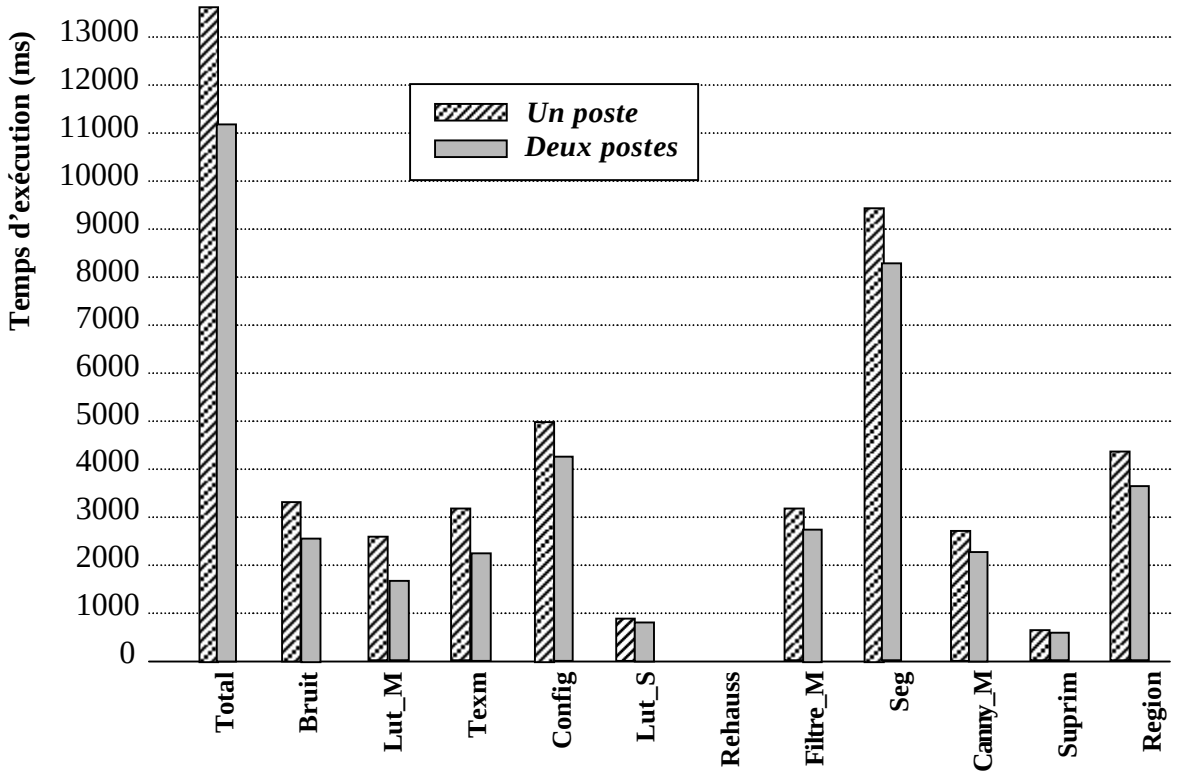


Figure 5-27 : Temps d'exécution de quelques agents pour l'image Maison

Les figures 5.25 5.26 et 5.27 représentent l'évolution du temps d'exécution par rapport au nombre de poste constituants la machine virtuelle. On remarque que plus le nombre de poste augmente et plus le temps d'exécution diminue.

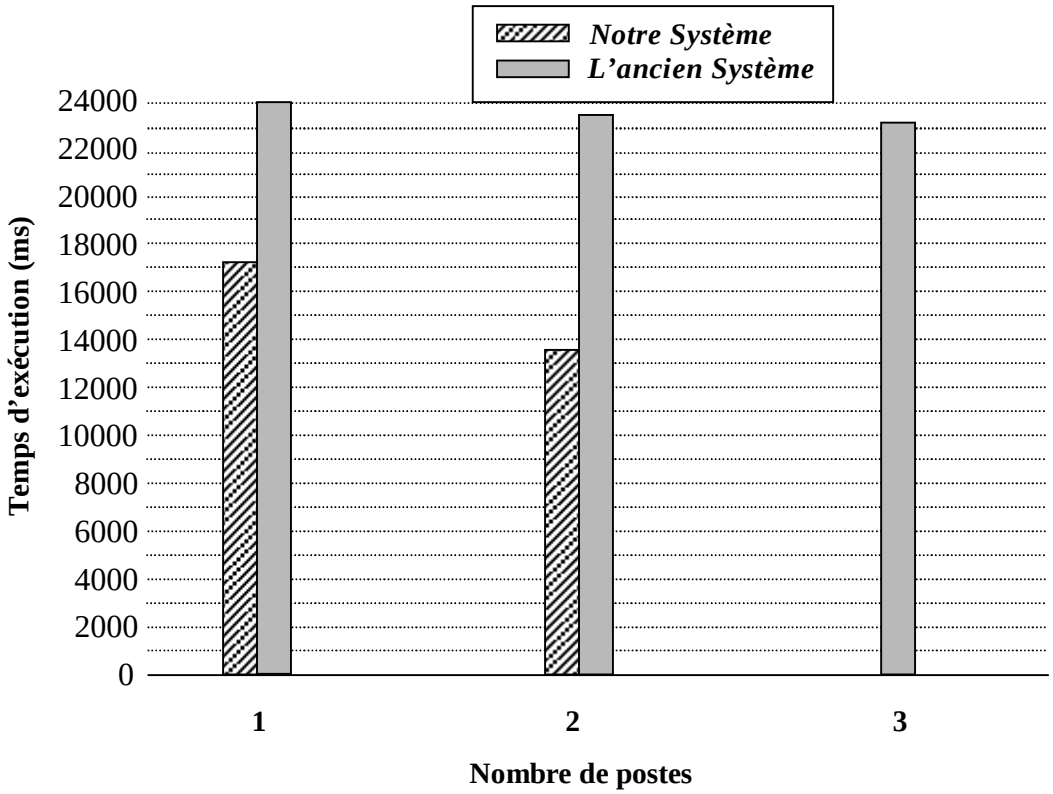


Figure 5-28 : Comparaison des temps d'exécutions de l'ancien système et de notre système

La figure ci-dessus (Fig 5-28) représente une comparaison de temps d'exécution du système multi-agents que nous avons réalisé par rapport à celui du système réalisé auparavant qui a été testé sur un réseau de trois PC Penthium III 1000 MHz reliés par un Hub [46].

D'après la figure 5-28, nous constatons que le temps d'exécution de notre système est largement inférieur à celui de l'ancien système ; ce qui est un de nos principaux objectifs.

CONCLUSION

Dans le cadre de la réalisation de notre projet : Approche Multi-Agents dans l'analyse d'images ; nous avons commencé par une analyse de la gestion des informations pour les algorithmes de détection d'indices visuels primaires, ce qui nous a permis de dégager quelques principes importants. Nous proposons le cumul des informations avant la prise de décision, la nécessité de guider les méthodes afin d'obtenir des informations complémentaires, l'exploitation d'informations "objectives" pour résoudre les problèmes d'émergence des indices visuels et enfin la "liberté" de l'information pour que la structure de contrôle soit efficace et permette par exemple le changement de contexte ou le report des décisions difficiles.

Ensuite, nous avons étudié l'analyse d'images en décrivant les différentes étapes du traitement d'images en commençant par l'acquisition de l'image, les prétraitements et à la fin les technique de segmentation coopérative (l'extraction de contours **à** la détection de régions homogènes et vice-versa).

Après, nous nous sommes intéressé à l'Intelligence Artificielle Distribuée (IAD), un domaine qui permet à résoudre des problèmes complexes en les divisant sur un ensemble d'entités distribué et coopérante. Puis nous avons ensuite présenté en détail les systèmes multi-agents, discipline qui découle de l'IAD. Ces systèmes font coopérer un ensemble d'agents dotés d'un comportement intelligent dont l'objectif est de coordonner leurs buts et leurs plans d'actions pour la résolution d'un problème.

Puisque les systèmes multi-agents sont basés sur la distribution des tâches entre plusieurs processus (un traitement parallèle des données), nous avons jugé utile de donner quelques notions de base concernant le parallélisme en général et les différentes architectures impliquées dans ce genre de traitement. Nous avons aussi présenté un ensemble d'outils logiciels destinées à faciliter la

programmation distribuée " multi-agents " sur des machines hétérogènes, en l'occurrence PVM (Parallel Virtual Machine).

L'étape finale s'est soldée par l'implantation d'une méthode de segmentation d'images coopérative :

- Contour-région : en bloquant la croissance des régions à proximité des chaînes de contours obtenues par une méthode de segmentation en contour (méthode de Canny).
- Régions-contours : en suivant les frontières des régions obtenues par la méthode de croissance des régions pour former la carte contour.

Cette méthode de segmentation coopérative a été réalisée par une approche multi-agents, en utilisant la plate-forme PVM comme un moyen de communication entre les agents du système.

Les résultas obtenus nous ont donné entière satisfaction vu les images obtenues, le temps de calcul parcouru et l'efficacité de la gestion et le contrôle des informations dans notre système.

Les méthodes développées et les algorithmes implantés peuvent sûrement être améliorés. Pendant la réalisation de ce travail, beaucoup d'idées sont apparues suite à des réflexions sur des problèmes rencontrés. Certaines ont été exploitées, d'autres demandent encore réflexion. Pour bien mûrir et donner naissance à de nouvelles conceptions, un certain nombre de remarques et perspective peuvent être faites pour des travaux futures, nous en citons quelque unes :

- l'utilisation d'une large base de connaissances, pour le choix des critères (paramètres), peut améliorer la qualité du résultat des traitements.
- Une étude approfondie des types de bruit, pouvant affecter une image, et les moyens de les estimer, peut aider à choisir avec précision le type de prétraitement que l'image peut subir.
- Utilisation de plusieurs informations extraites de l'image peut améliorer les résultats de la segmentation (contraste, texture, luminosité, écart-types, gradient, formes…)

- L’implantation d’autres méthodes de segmentation coopérative, par exemples : la coopération entre la méthode de Dériche ou de Kirsh ou autre pour la segmentation en contour d’une part, avec la méthode de Voronoî ou de division/fusion ou autre pour la segmentation en région d’autre part, et faire une étude comparative.

APPENDICE

APPENDICE A
METHODE DES MOINDRES CARRES

La régression polynomiale

Soit une famille de couples de points (x_i,y_i) tel que :

$y_i = f(x_i)$: y_i est une fonction de xi

Tableau A-1 : Echantillon de points de la fonction $y_i = f(x_i)$

X	x_1	x_2	- -	x_M
Y	y_1	y_2	- -	y_M

La régression polynomiale est caractérisée par la fonction :

$$y = a_0 + a_1 x^1 + a_2 x^2 + a_3 x^3 + \ldots\ldots\ldots\ldots\ldots + a_k x^k$$

Ou encore $y = \sum_{j=0}^{k} a_j x^j$ soit $y_i = \sum_{j=0}^{k} a_j x^k{}_i$

Les coefficients a0, a1, a2, …ak sont déterminés par la méthode des moindres carrés, il faut rendre minimale l'expression suivante :

$$D = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{k} a_j x_i^j \right)^2$$

Ceci revient à annuler toutes les dérivées partielles par rapport à a_j.

$$\frac{\partial D}{\partial a_j} = -2 \sum_{i=1}^{M} x_i^j \left(y_i - \sum_{j=0}^{k} a_j x_i^j \right)$$

$$\sum_{i=1}^{M} x_i^j \left(y_i - \sum_{j=0}^{k} a_j x_i^j \right) = 0$$

$$\sum_{i=1}^{M} x_i^j y_i - \sum_{i=1}^{M} x_i^j \sum_{j=0}^{k} a_j x_i^j = 0$$

$$\sum_{i=1}^{M} x_i^j \sum_{j=0}^{k} a_j x_i^j = \sum_{i=1}^{M} x_i^j y_i$$

$$\sum_{i=1}^{M} x_i^j \left(a_0 + a_1 x_i^1 + a_2 x_i^2 + \ldots\ldots\ldots + a_k x_i^k \right) = \sum_{i=1}^{M} x_i^j y_i$$

$$a_0 \sum_{i=1}^{M} x_i^j + a_1 \sum_{i=1}^{M} x_i^{j+1} + \ldots\ldots\ldots a_k \sum_{i=1}^{M} x_i^{j+k} = \sum_{i=1}^{M} x_i^j y_i \quad \ldots(1)$$

On fait un changement de variables :

$$X_j = \frac{1}{M} \sum_{i=1}^{M} x_i^j$$

.

.

.

.

.

$$X_{j+k} = \frac{1}{M} \sum_{i=1}^{M} x_i^{j+k}$$

$$E_j = \frac{1}{M} \sum_{i=1}^{M} x_i^j y_i$$

On divise tous les termes de l'équation (1) par M et on introduit les *X* et les *E* :

$a_0 X_j + a_1 X_{j+1} + \ldots\ldots\ldots\ldots + a_k X_{j+k} = E_j$

Nous sommes en présence d'un système d'équations à k+1 inconnues et k+1 équations :

$J=0 \quad a_0 X_0 + a_1 X_1 + \ldots\ldots\ldots\ldots + a_k X_k = E_0$

$J=1 \quad a_0 X_1 + a_1 X_2 + \ldots\ldots\ldots\ldots + a_k X_{k+1} = E_1$

.

.

.

.

.

$J=k \quad a_0 X_k + a_1 X_{k+1} + \ldots\ldots\ldots\ldots + a_k X_{2k} = E_k$

Ce système s'écrit sous forme matricielle comme suit :

$$\begin{bmatrix} X_0 & X_1 & \ldots\ldots & X_k \\ X_1 & X_2 & \ldots\ldots & X_{k+1} \\ . & . & & . \\ . & . & \ldots\ldots & . \\ . & . & & . \\ X_k & X_{k+1} & \ldots\ldots & X_{2k} \end{bmatrix} \text{x} \begin{bmatrix} a_0 \\ a_1 \\ . \\ . \\ . \\ a_k \end{bmatrix} = \begin{bmatrix} E_0 \\ E_1 \\ . \\ . \\ . \\ E_k \end{bmatrix}$$

A * R = E

Soit A^{-1} la matrice inverse de A

$A^{-1} * A * R = A^{-1} E$

Comme $A^{-1} * A = I$ (matrice identité)

Alors la relation finale est : **$R= A^{-1} * E$**

Nous avons implanté cette méthode dans un logiciel.

Nous avons utilisé les valeurs du tableau 5.1 et nous avons obtenu les coefficients de notre polynôme comme il est indiqué dans la figure suivante :

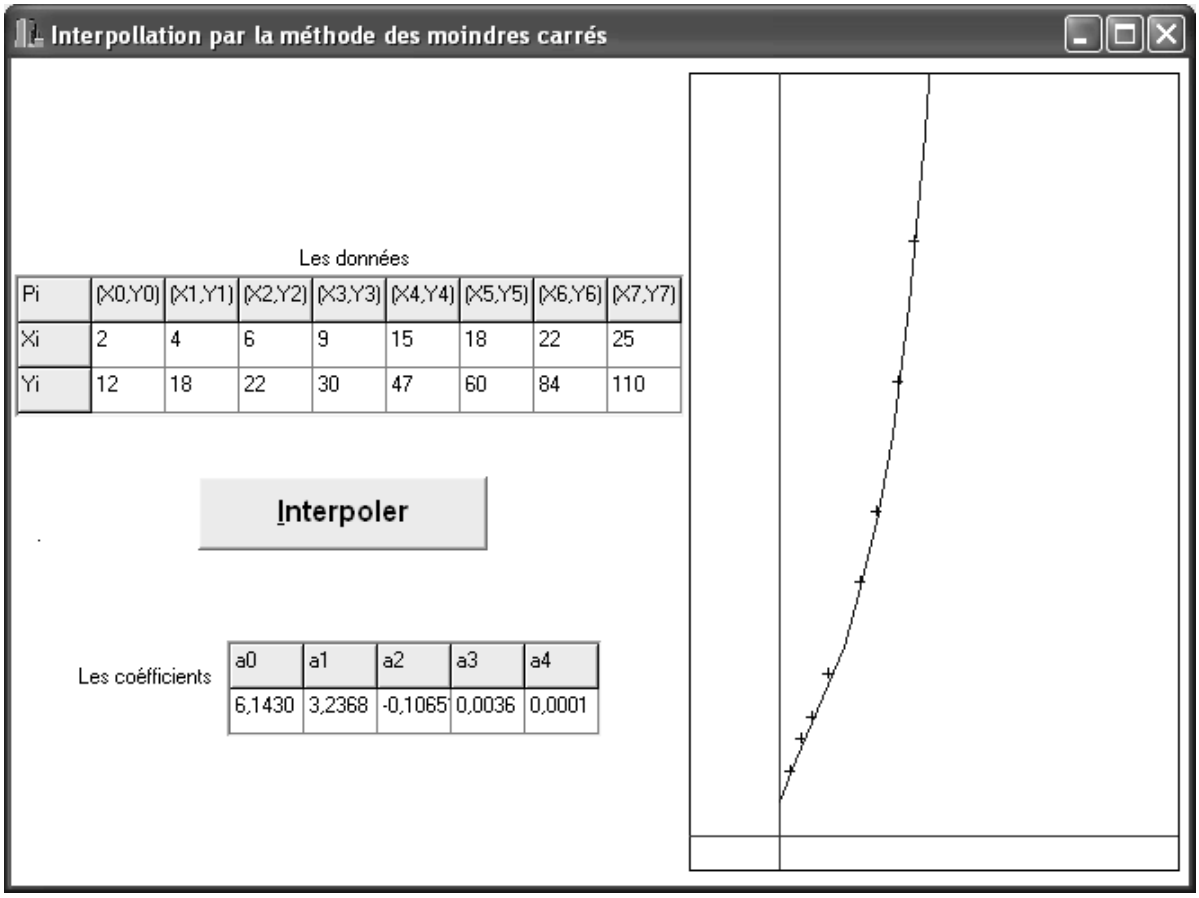


Figure A-1 : Logiciel d'interpolation par la méthode des moindres carrés

APPENDICE B
CALCULS DES PARAMETRES DE PRETRAITEMENT

Début

W ; *// largeur de l'image*

H ; // *hauteur de l'image*

Ima[i][j] ; // *l'image tel que :0 <= i < W, 0 <= j < H*

//Calcul de l'histogramme

Pour i = 0 *à* W

Pour j = 0 *à* H

Histo**[**Ima[i][j]**]** = Histo**[**Ima[i][j]**]** + 1 ;

//Calcul de la matrice de cooccurrence sur les 4 directions

//Coc0 pour 0°, Coc1 pour 45°, Coc2 pour 90°, Coc3 pour 135°

Pour i = 1 à W-1

Pour j = 1 à H-1

{

Coc0**[**Ima[i][j]**][**Ima[i+1][j]**]** = Coc0**[**Ima[i][j]**][**Ima[i+1][j]**]** + 1 ;

Coc1**[**Ima[i][j]**][**Ima[i+1][j-1]**]** = Coc1**[**Ima[i][j]**][**Ima[i+1][j-1]**]** + 1 ;

Coc2**[**Ima[i][j]**][**Ima[i][j-1]**]** = Coc2**[**Ima[i][j]**][**Ima[i][j-1]**]** + 1 ;

Coc3**[**Ima[i][j]**][**Ima[i-1][j-1]**]** = Coc3**[**Ima[i][j]**][**Ima[i-1][j-1]**]** + 1 ;

}

//estimation de la luminosité

Taux = (7*W*H)/100 ; *// 7%*

Fin = faux ;

i = 0 ;

S = 0 ;

Tanque (Fin = faux)

{

S = S + Histo[i] ;

Si (s > Taux)

Fin = Vrai ;

```
        Sinon
                i = i + 1 ;
}
min = i ;
i = 255 ;
S = 0 ;
Fin = faux;

Tanque ( Fin = faux )
{
        S = S + Histo[i];
        Si ( S > Taux )
                Fin = Vrai;
        Sinon
                i = i – 1 ;
}
max = i ;
Si ( ( min > 45) ou ( max < 210 ) )
Appliquer un recadrage dynamique ;
//Estimation du contraste
Ctr0 = 0 ;
Pour i = 1 à 255
{
        c0 = 0 ;        c1 = 0 ;
        c2 = 0 ;        c3 = 0 ;
        Pour j = 0 à 255
        Pour k = 0 à 255
        {
                Si ( abs ( j – k ) = i )
                {
                        Si ( Coc0[j][k] != 0 )
                        {
                                c0 = c0 + Coc0[j][k] ;
                        }
```

```
                    Si ( Coc1[j][k] != 0 )
                    {
                        c1 = c1 + Coc1[j][k] ;
                    }
                    Si ( Coc2[j][k] != 0 )
                    {
                        c2 = c2 + Coc2[j][k] ;
                    }
                    Si ( Coc3[j][k] != 0 )
                    {
                        c3 = c3 + Coc3[j][k] ;
                    }
                }
            }
            Ctr0 = Ctr0 + c0*i*i ;
            Ctr1 = Ctr1 + c1*i*i ;
            Ctr2 = Ctr2 + c2*i*i ;
            Ctr3 = Ctr3 + c3*i*i ;
}
Nc0 = 256*li*col ;
Ctr = ( (Ctr0 + Ctr1 + Ctr2 + Ctr3 ) / Nc0 ) / 4 ;

//Estimation de la texture
c0 = c1 = c2 = c3 = 0 ;
Pour j1 = 0 à 256
Pour k1 = 0 à 256
{
        c0 = c0 + Coc0[j1][k1] / (1 + ( k1 - j1 )*( k1 - j1 ) ) ;
        c1 = c1 + Coc1[j1][k1] / (1 + ( k1 - j1 )*( k1 - j1 ) ) ;
        c2 = c2 + Coc2[j1][k1] / (1 + ( k1 - j1 )*( k1 - j1 ) ) ;
        c3 = c3 + Coc3[j1][k1] / (1 + ( k1 - j1 )*( k1 - j1 ) ) ;
}
Tex = ( ( c0 + c1 + c2 + c3 ) / ( ( li – 2 )*( col – 2 ) ) ) / 4 ;
Fin.
```

APPENDICE C
L'INTERFACE GRAPHIQUE IMPLANTEE LTI

L'interface graphique que nous avons implantée s'intitule « LTI » (Logiciel de Traitement d'Images), ce logiciel est composé de :

- une partie englobe toutes les méthodes d'analyse d'images programmées déjà dans notre système multi-agents « SMATI » et autres. Cette partie nous permet de visualiser directement le résultat de chacune des méthodes sur différents types de format d'images (bmp, jpg, wmf, raw…) avec une grande simplicité. Par exemple l'utilisateur peut voir le résultat de la binarisation d'une image en variant le seuil de '0' jusqu'à '255' et ceci en quelque seconde.

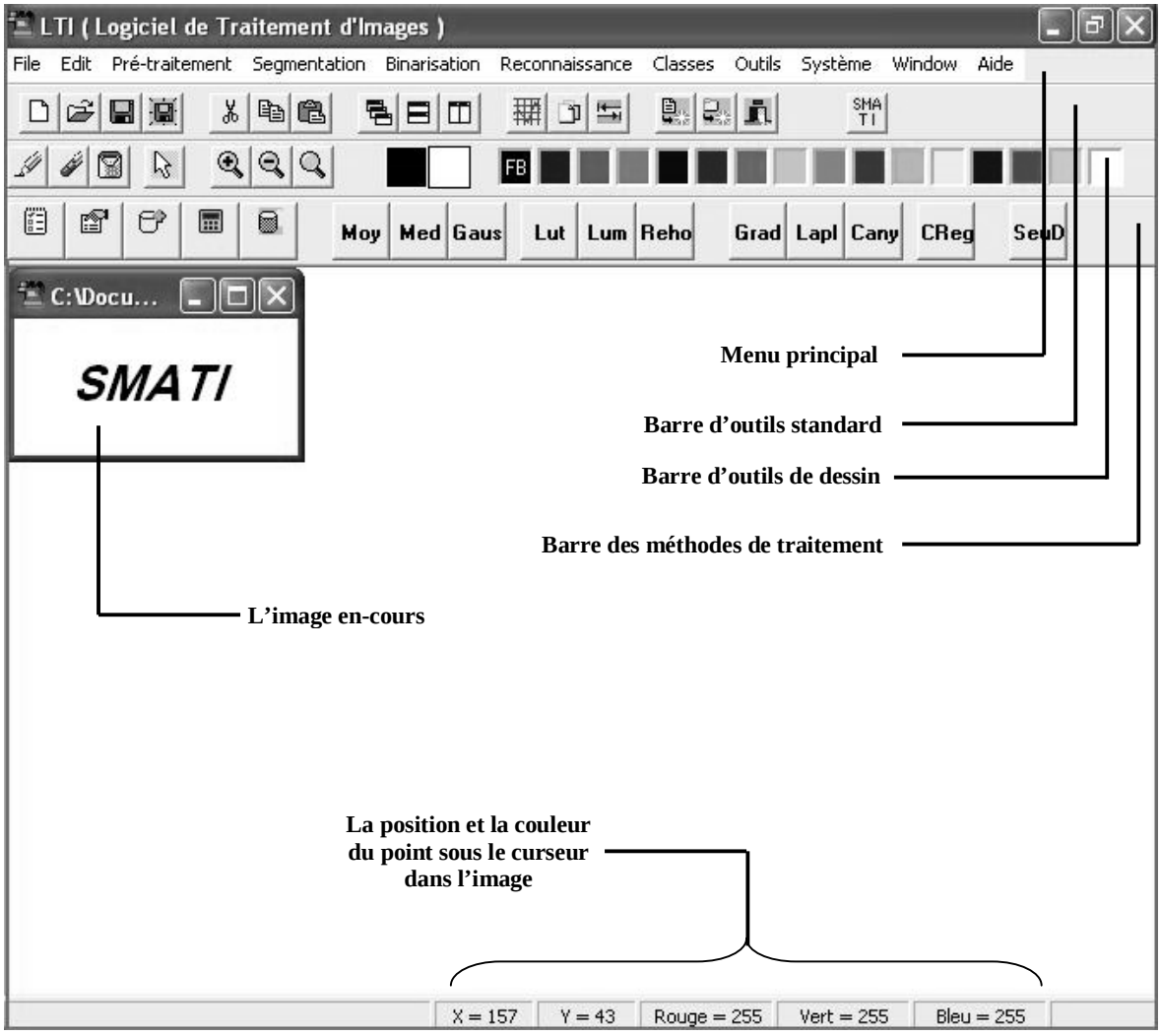


Figure C-1 : Fenêtre principale de l'interface graphique

- Une partie comporte des outils qui peuvent aider l'utilisateur, on peut citer :
    - les outils de dessin : Crayon, Remplisseur de zone, gomme, palette des couleurs,
    - l'histogramme,
    - l'enregistrement sous trois Formats : Standard Bitmap 24bits « bmp », Compressé Jpeg « jpg » et Brute « raw »,
    - le Zoom : zoom avant et zoom arrière,
    - et autres,

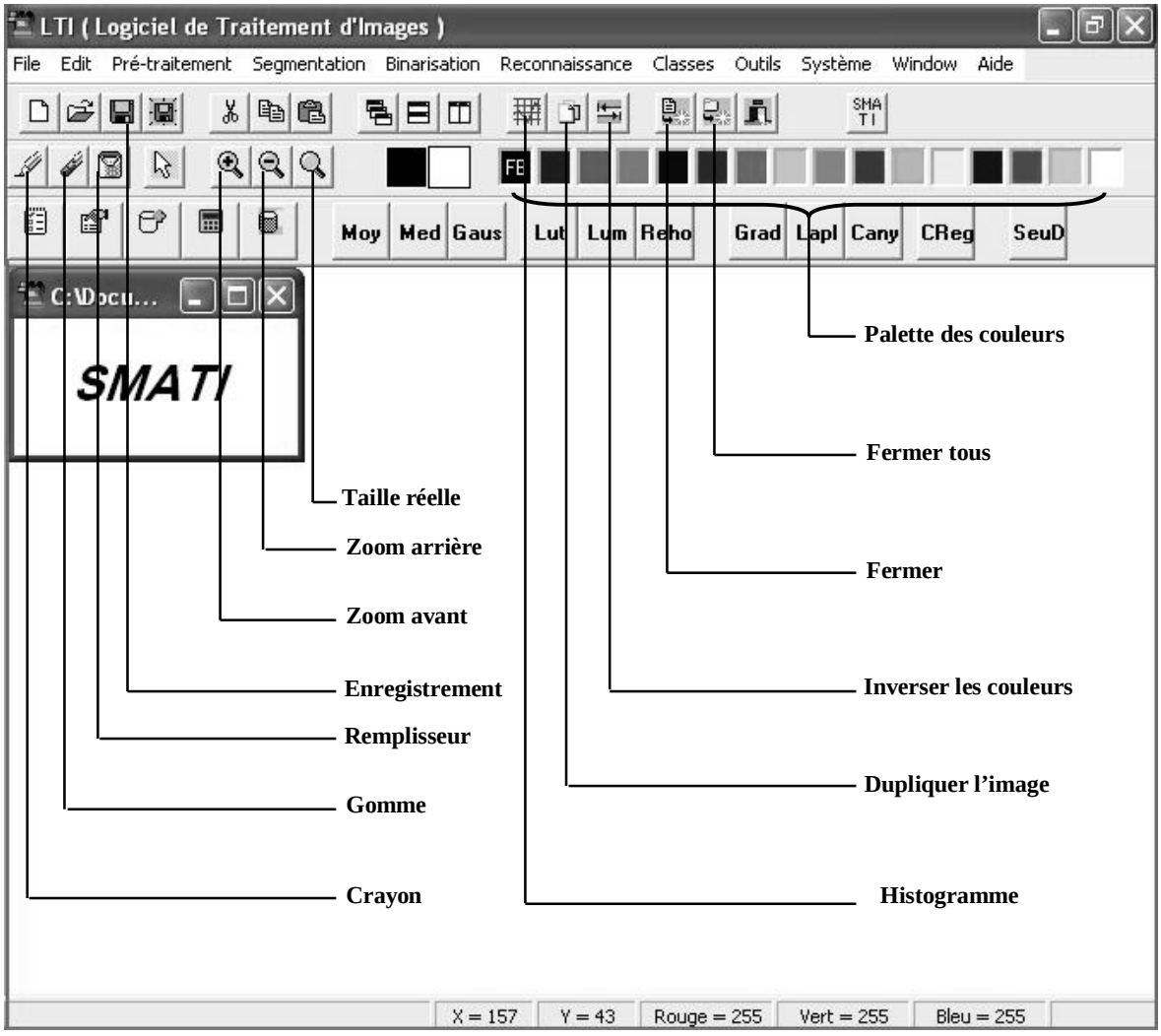


Figure C-2 : Les outils du logiciel LTI

- Une partie pour le lancement du système multi-agents : elle est composée de deux éléments :

    - le premier est charger de lancer l'agent « Maitre » qui est l'agent le plus haut dans la hiérarchie de notre système (il est lancer par l'utilisateur et non pas par un autre agent).

    - Le deuxième se charge de transférer l'image en-cours à l'agent « Maitre » pour la traiter.

  Cette partie est très importante, car elle donne la possibilité au système multi-agents de traiter n'importe quel type d'images, ce qui n'est pas possible en utilisant directement le système puisque il traite seulement les fichiers d'images brutes (.ima et .raw).

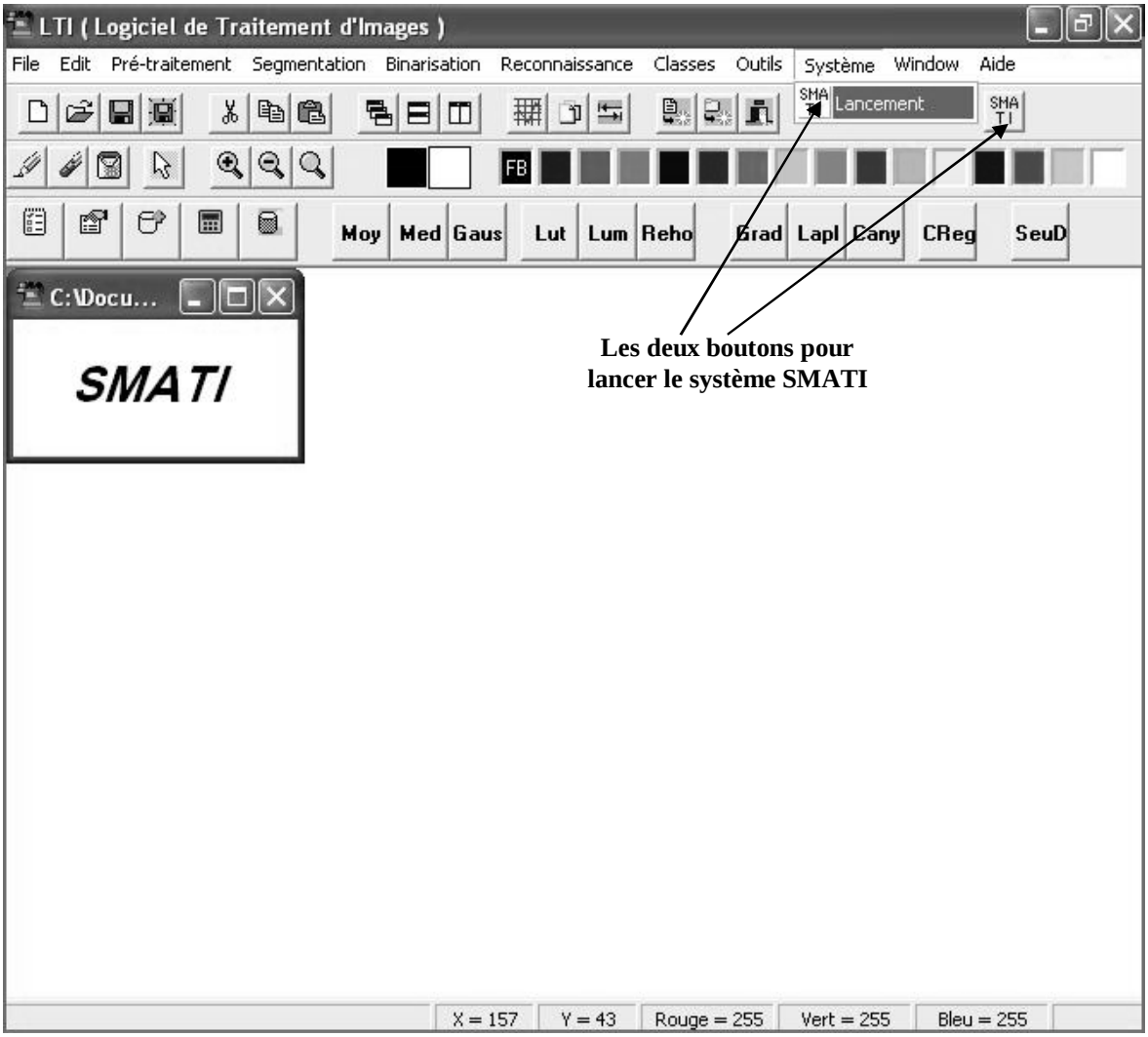


Figure C-3 : Lancement du système multi-agents par l'interface graphique

- Une partie pour la visualisation du déroulement des agents du système « SMATI » cette partie nous permet de voir l'état du système instant par instant, du début du lancement jusqu'à la fin. Elle nous indique à chaque moment les agents qui sont en-cours d'exécution, les agents qui ont terminés leurs tâches et les agents qui n'ont pas encore été lancés.

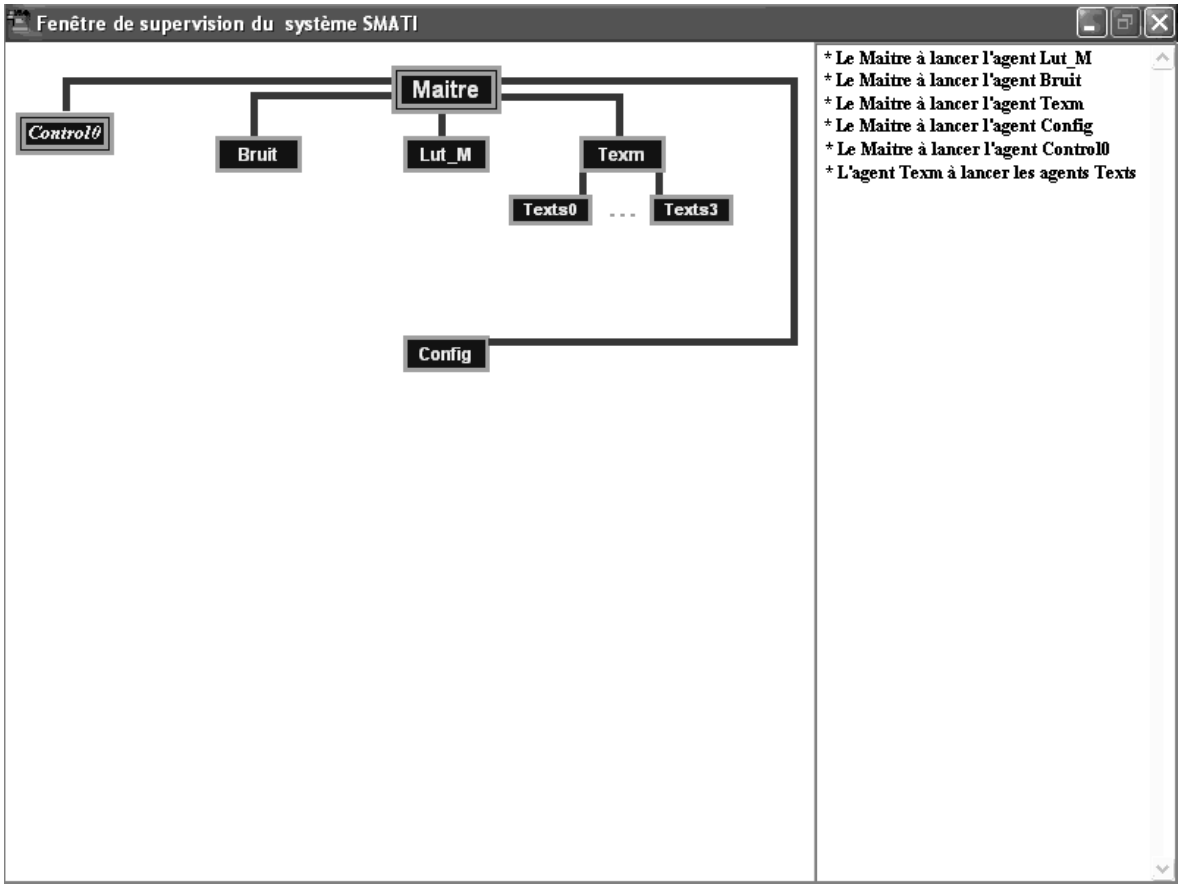


Figure C-4 : Supervision du système SMATI au début du traitement

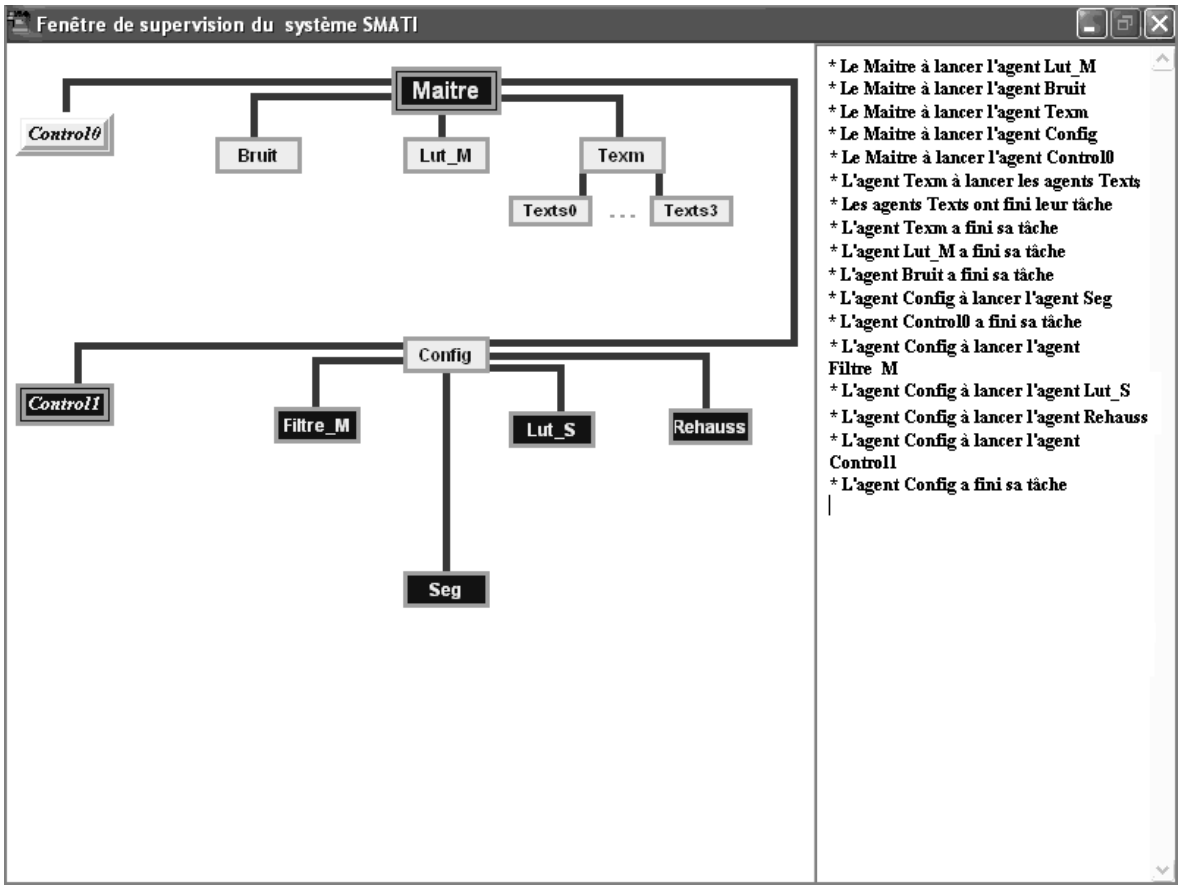


Figure C-5 : Supervision du système SMATI au milieu du traitement

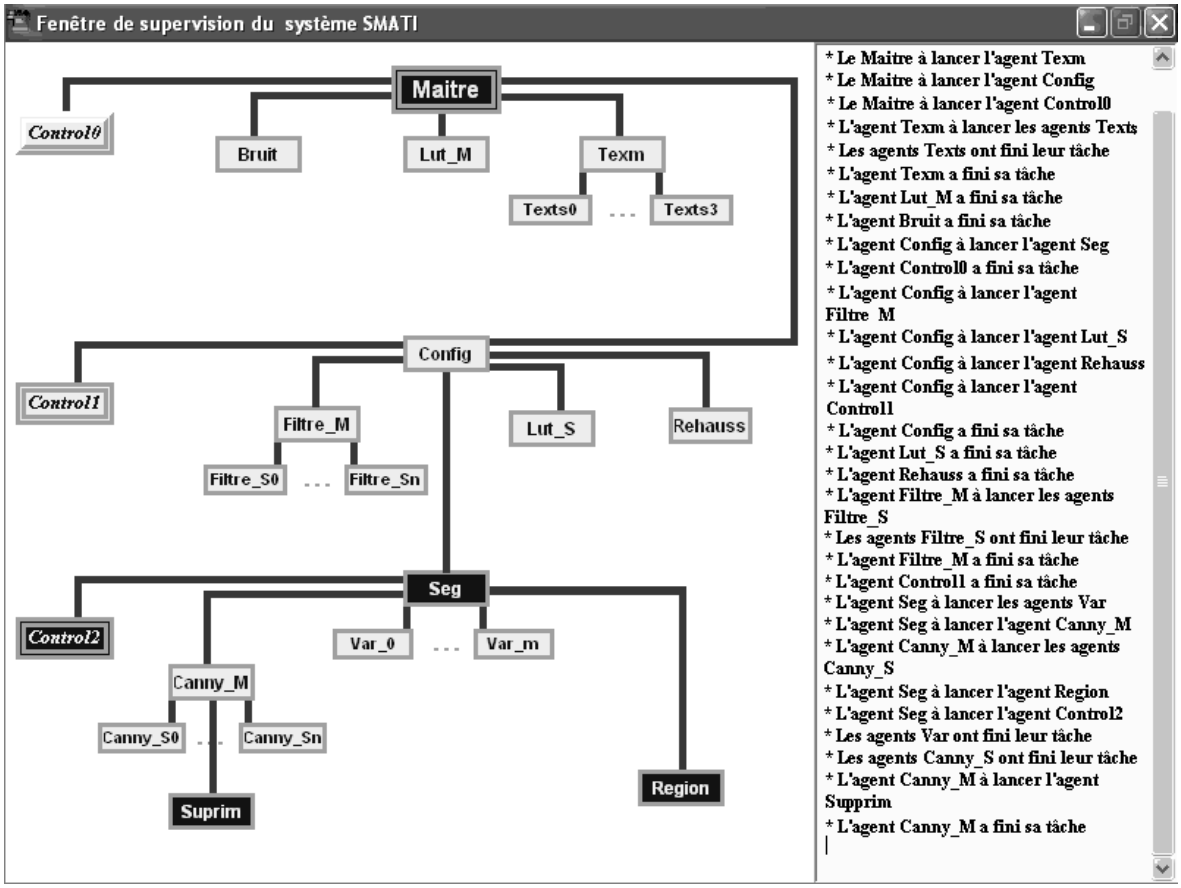


Figure C-6 : Supervision du système SMATI à la fin du traitement

APPENDICE D
PRIMITIVES DE PVM

a) Routines de contrôle de tâches

Tableau D-1 : Les routines de contrôle de tâches de PVM

routine C	routine f77	Description rapide
pvm_mytid	pvmfmytid	fourni le tid de la tâche courante
pvm_exit	pvmfexit	indique au pvmd local que la tâche quitte PVM
pvm_kill	pvmfkill	termine une tâche PVM
pvm_spawn	pvmfspawn	débute une nouvelle tâche
pvm_parent	pvmfparent	indique le tid de la tâche qui a enfanté la tâche courante
pvm_pstat	pvmfpstat	indique l'état de la tâche spécifiée par son tid
pvm_mstat	pvmfmstat	indique l'état d'une machine faisant partie de la machine virtuelle
pvm_config	pvmfconfig	fourni des informations sur la machine virtuelle
pvm_tasks	pvmftasks	fourni des informations sur les tâches tournant sur la machine virtuelle

b) Routines de manipulation de buffer

Tableau D-2 : Les routines de manipulation de buffer de PVM

routine C	routine f77	Description rapide
pvm_mkbuf	pvmfmkbuf	crée un buffer pour message
pvm_freebuf	pvmffreebuf	désalloue la mémoire réservée à un buffer
pvm_getsbuf	pvmfgetsbuf	retourne le bufid du buffer actif émetteur
pvm_getrbuf	pvmfgetrbuf	retourne le bufid du buffer actif receveur
pvm_setsbuf	pvmfsetsbuf	change de buffer émetteur actif
pvm_setrbuf	pvmfsetrbuf	change de buffer receveur actif
pvm_initsend	pvmfinitsend	initialise le buffer spécifié

c) Routines de manipulation de messages en émission

Tableau D-3 : Routines de manipulation de messages en émission par PVM

Routine C	Routine f77	Description rapide
pvm_pkbyte	pvmfpack	défini et crée le type des données du buffer actif
pvm_pkcplx	pvmfpack	idem
pvm_pkdcplx	pvmfpack	idem
pvm_pkdouble	pvmfpack	idem
pvm_pkfloat	pvmfpack	idem
pvm_pkint	pvmfpack	idem
pvm_pklong	pvmfpack	idem
pvm_pkshort	pvmfpack	idem
pvm_pkstr	pvmfpack	idem
pvm_send	pvmfsend	envoie les données dans le buffer actif à destination d'une tâche
pvm_mcast	pvmfmcast	envoie les données dans le buffer actif à destination d'un ensemble de tâches

d) Routines de manipulation de messages en réception

Tableau D-4 : Routines de manipulation de messages en réception par PVM

routine C	routine f77	Description rapide
pvm_upkbyte	pvmfunpack	décompacte le buffer actif selon le type donnée
pvm_upkcplx	pvmfunpack	idem
pvm_upkdouble	pvmfunpack	idem
pvm_upkfloat	pvmfunpack	idem
pvm_upkdcplx	pvmfunpack	idem
pvm_upkint	pvmfunpack	idem
pvm_upklong	pvmfunpack	idem
pvm_upkshort	pvmfunpack	idem
pvm_upkstr	pvmfunpack	idem
pvm_recv	pvmfrecv	réception d'un message
pvm_nrecv	pvmfnrecv	réception de message non bloquante
pvm_recvf	Non Disponible	voir le manuel
pvm_bufinfo	pvmfbufinfo	fourni des informations sur un buffer

e) Routines de configuration dynamique de la machine virtuelle ou d'utilisation de groupes

Tableau D-5 : Routines de groupe et de configuration dynamique de PVM

routine C	Routine f77	Description rapide
pvm_addhosts	pvmfaddhosts	ajoute une machine à la machine virtuelle
pvm_delhosts	pvmfdelhost	ôte une machine de la machine virtuelle
pvm_joingroup	pvmfjoingroup	ajoute une tâche dans un groupe
pvm_lvgroup	pvmflvgroup	ôte une tâche d'un groupe
pvm_gsize	pvmfgsize	retourne le nombre de membres d'un groupe
pvm_gettid	pvmfgettid	retourne le tid d'une tâche d'un groupe
pvm_getinst	pvmfgetinst	retourne l'instance d'une tâche dans un groupe
pvm_barrier	pvmfbarrier	routine de syncronisation des tâches d'un groupe
pvm_bcast	pvmfbcast	envoie les données du buffer actif à un groupe de tâches

f) Routines de gestion d'erreurs

Tableau D-6 : Routines de gestion d'erreurs de PVM

routine C	routine f77	Description rapide
pvm_perror	pvmfperror	retourne un message correspondant à la dernière erreur
pvm_serror	pvmfserror	active ou desactive les messages d'erreurs

g) Routines de signal

Tableau D-7 : Routines de signal de PVM

routine C	routine f77	Description rapide
pvm_sendsig	pvmfsendsig	envoie un signal à une autre tâche PVM
pvm_notify	pvmfnotify	demande d'information aux événements PVM

h) Routines diverses: qu'on qualifierais de moins utilisées

Tableau D-8 : Des routines diverses de PVM utilisées rarement

routine C	routine f77	Relatif à
pvm_advise	pvmfadvise	la communication entre tâches
pvm_archcode	pvmfarchcode	l'architecture de la machine virtuelle
pvm_catchout	pvmfcatchout	aux Entrées/Sorties
pvm_delete	Non Disponible	aux bases de données PVM
pvm_gather	pvmfgather	la communication dans un groupe
pvm_getopt	pvmfgetopt	la bibliothèque PVM
pvm_halt	pvmfhalt	la gestion de la machine virtuelle
pvm_hostsync	pvmfhostsync	l'heure
pvm_insert	Non Disponible	aux bases de données PVM
pvm_lookup	Non Disponible	aux bases de données PVM
pvm_precv	pvm_fprecv	la communication entre tâches
pvm_probe	pvm_fprobe	la communication entre tâches
pvm_psend	pvmfpsend	la communication entre tâches
pvm_reduce	pvmfreduce	la gestion d'un groupe
pvm_reg_hoster	Non Disponible	la gestion des tâches
pvm_reg_rm	Non Disponible	la gestion des tâches
pvm_reg_tasker	Non Disponible	la gestion des tâches
pvm_scatter	pvmfscatter	la communication dans un groupe
pvm_setmwid	Non Disponible	la communication entre tâches
pvm_setopt	pvmfsetopt	la bibliothèque PVM
pvm_settmask	Non Disponible	la gestion des tâches
pvm_tidtohost	Non Disponible	la gestion des tâches
pvm_trecv	pvmftrecv	la communication entre tâches

REFERENCES

1. Brice. C.R et Fennema. C.L « Scene analysis using regions », Artificial Intelligence, vol. 1, (1970), p. 205-226.

2. Nazif. A.M et Levine. M.D, « Low level Image Segmentation : An Expert System ». IEEE Trans. Patt. Anal. Mach. Int., Vol. 6, n°5, USA, (Mars 1984), p. 555-577.

3. Lux. A, « Algorithmique et contrôle en vision par ordinateur », Thèse de doctorat, l'Institut National Polytechnique de Grenoble, France, (1985).

4. Canny. J.F, « A computational approach to edge detection », IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, n°6, USA, (Novembre 1986), p. 679-698.

5. Baujard. O et Garbay. C, « KISS : un système de vision multi-agents », 7ème congré Reconnaissance des formes et intelligence artificielle,, AFCET/INRIA, (1990).

6. Clement. V, « Raisonnements cognitifs appliqués au pilotage d'algorithmes de traitement d'images », Thèse INRIA, Nice-Sophia-Antipolis, France, (1990).

7. Pavlidis. T et Liow. Y, « Integrating region growing and edge detection », IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, USA, (Mars 1990), p.225-233.

8. Beucher. S « Segmentation d'images et morphologie mathématique » Thèse de doctorat Ecole nationale supérieure des Minces de paris, France, (Juin 1990).

9. Toumazet. J.S « Traitement de l'image par l'exemple » Edition SYBEX, France, (Juillet 1990).

10. Erceau. J et Ferber. J, « L'intelligence artificielle distribuée ». La Recherche, 22, n°233, France, (1991), p. 750-758.

11. Labidi. S et Lejouad. W, « De l'intelligence artificielle distribuée aux systèmes multi-agents » Rapport de recherche N° 2004. INRIA, France, (1993).

12. Boissier. O « Problème du contrôle dans un système intègre de vision utilisation d'un système multi-agents » Thèse de doctorat, l'Institut National Polytechnique de Grenoble, France, (Janvier 1993).

13. Baujard. O and Garbay. C « KISS : A multi-agent segmentation system », SPIE Optical Engineering, vol. 32, n° 6, USA, (june 1993), p. 1235-1249.

14. Chu. C.C et Aggarwal. J.K, « The integration of image segmentation maps using region and edge information », PAMI IEEE Computer Society, Vol. 15, n° 12, Washington DC, USA, (Decembre 1993), p. 1241-1252.

15. Salotti. J.M, « Gestion des informations dans les premières étapes de la vision par ordinateur », Thèse de doctorat, l'Institut National Polytechnique de Grenoble, France, (Janvier 1994).

16. Garbay. C et Salotti. J, « Détection de contours : les heuristiques remplacent avantageusement les modèles », Actes du congrès RFIA de l'AFCET, Paris, France, (janvier 1994).

17. Ferber. J, « Les systèmes Multi-Agents : Vers une intelligence collective ». Inter Editions, France, (1995).

18. Cocquerez. J.P et Philipp. S, « Analyse d'images : filtrage et segmentation », Edition MASSON, France, (1995).

19. Sichman. J.S, « Du raisonnement social chez les agents : une approche fondée sur la théorie de la dépendance », Thèse de doctorat, l'Institut National Polytechnique de Grenoble, France, (Septembre 1995).

20. Brassac. C et Pesty. S, « La Pelouse Fourmilière : De la Coaction à la Coopération.. » Dans Quatrièmes Journées Francophones IAD-SMA, France, (1996).

21. Amzal. D, « Intelligence artificielle distribuée et approche multi-agents pour le développement d'un système d'information » Thèse de Magister en informatique, INI, Algérie, (1993).

22. Despons. R « Conception d'une Machine Virtuelle pour les Systèmes Parallèles à Diffusion », Thèse de doctorat, l'Institut National Polytechnique de Grenoble, France, (Décembre 1996).

23. Demazeau. Y, « Steps Toward Multi-Agent Programming ». Dans IWMAS-97, Edition Asm Press, New York, USA, (1997).

24. Sabah.G, « Dialogue et Sciences Cognitives ». Edition, Le Dialogique, Sciences pour la communication, Berne, Suisse, (1997), p 323-346.

25. Attoui. A « Les système multi-agents et le temps réel » Edition Eyrolles, France, (1997)

26. Bernard. P.E, « Parallélisation et multiprogrammation pour une application irrégulière de dynamique moléculaire opérationnelle », Thèse de doctorat, l'Institut National Polytechnique de Grenoble, France, (Octobre 1997).

27. Cho. K.J et Meer. P, « Image segmentation from consensus information », CVIU, 68, n° 1, USA, (Octobre 1997), p. 72-89.

28. Baeijs. C « Fonctionnalité émergente Dans Une Société D'agents Autonomes. Etude Des Aspects Organisationnels Dans Les Systèmes Multi-Agents Réactifs ».Thèse de doctorat, INP Grenoble, France, (1998).

29. Bellet. F « Une approche incrémentale, coopérative et adaptative pour la segmentation des images en niveaux de gris ». Thèse de doctorat, Institut National Polytechnique de Grenoble, France, (Juin 1998).

30. Boucher. A, « Une approche décentralisée et adaptative de la gestion d'informations en vision », Thèse de doctorat, l'université de Joseph Fourier - Grenoble I, France, (1999).

31. Ouellhadj. D « Système multi agents pour le pilotage distribué de cellules flexible de production autonomes » Thèse de Magister en Informatique, INI, Algérie, (1999).

32. Francis. V.A, « Les systèmes multi-agents minimaux » Thèse de doctorat de l'INPG Spécialité : Informatique Systèmes et Communications, France, (Mars 1999).

33. Fjortoft. R, « Segmentation d'image radar par détection de contours », Thèse de doctorat, l'Institut National Polytechnique de Toulouse, France, (Mars 1999).

34. Kebir. Y, « Segmentation d'image de films de radiographie dédiée au contrôle non destructif » Thèse de Magister en électronique, l'université de Blida, Algérie, (Juillet 1999).

35. Laurence. G, « Trois principes de coopération pour la segmentation en imagerie de résonance magnétique cérébrale », Thèse de doctorat, l'université de Joseph Fourier - Grenoble I, France, (Novembre 1999).

36. Goubault. É, Nataf. F et Schoenauer. M « Calcul Parallèle » Cours de l'école Polytechnique de Grenoble, France, (2000/2001),
Site : www.enseignement.polytechnique.fr

37. Matoug. S, « Détection et reconnaissance des défauts de joints de soudures d'images radiographiques. » Thèse magister Université. Annaba, Algérie, (2000).

38. Schüpp. S, « Prétraitement et segmentation d'images par mise en oeuvre de techniques basées sur les équations aux dérivées partielles : application en imagerie microscopique biomédicale », Thèse de doctorat, l'université de Caen, France, (2000).

39. Roques. D « Un petit guide sur P.V.M.3 »
E-Mail : Didier.Roques@brive.unilim.fr

40. Chaib-draa. B, Jarras. I et Moulin. B, « Système multiagents : Principes généraux et applications », rapport de recherche, Département d'informatique, Université Laval, Canada, (2001).

41. Briot. J.P et Demazeau. Y, « Principes et Architecture Des Systèmes Multi-Agents. Hermes, Paris, France, (2001).

42. Fernandes. K, « Systèmes Multi-Agents Hybrides : Une Approche Pour la Conception de Systèmes Complexes ». Thèse de doctorat, Université Joseph Fourier, Grenoble, France, (2001).

43. Fontaine. M, « Segmentation non supervisée d'images couleur par analyse de la connexité des pixels » Thèse de doctorat, l'université de Lille1, France, (Décembre 2001).

44. Duchesnay. E, « Agents situé dans l'image et organisés en pyramide irrégulière : Contribution à la segmentation par une approche d'agrégation coopérative et adaptative » Thèse de Doctorat, Université de Rennes1, France, (Décembre 2001).

45. Chicoisne. G, « Dialogue entre agents naturels et agents artificiels » Thèse de doctorat, l'Institut National Polytechnique de Grenoble, France, (Décembre 2002).

46. Boukaci. A et Chelbeb. A, « Paramllélisation des algorithmes de traitement d'images sous environnement PVM », Mémoire de PFE, Université des Sciences et de la Technologie Houari BOUMEDIENE, Bab-Ezzouar, Algérie, (Décembre 2003).