N* Enregistrement

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPOLAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SIENTIFIQUE

Université Elhadj Lakhder, Batna

Faculté des sciences de l'ingénieur

Département d'informatique

Thèse

Pour l'obtention du diplôme de magister en Informatique
Option: Informatique Industrielle

Intitulée

Analyse du mouvement et suivi de cibles mobiles dans les séquences vidéo

Présentée et soutenue par

KADACHE NABIL

Le : 25/09/2008

Directeur de Thèse : Pr M.C.BATOUCHE

Devant le jury composé de:

Dr A.ZIDANI	**Maître de conférence**	**U. Batna**	**Président.**
Dr A.BILAMI	**Maître de conférence**	**U. Batna**	**Examinateur**
Dr S.CHIKHI	**Maître de conférence**	**U. Constantine**	**Examinateur**

Remerciements

;

Je tiens avant tout à exprimer ma reconnaissance à Mr M.C Batouche, Professeur a l'université de Constantine, qui a assuré la direction de ma thèse, pour son suivi et ses précieux conseils,

Je remercie également Mrs ZIDANI, Maître de Conférence à l'Université de Batna, de m'avoir fait l'honneur de présider le jury. Mr BILAMI Maître de Conférence à l'Université de Batna et Mr Chikhi Maître de Conférence à l'Université de Constantine, qui ont accepté d'être membres du jury,

Mes vives remerciements aux enseignants de l'université de Batna, en particulier Mr Belattar mon promoteur du projet de fin d'étude,

Sans oublier Mr Dekhinet pour ses encouragements et son aide précieuse tant au niveau scientifique que personnelle,

Je remercie ma grande et petite famille pour leur soutien moral indéfectible,

Louange a dieu le tout puissant.

Résumé:

Le Suivi du mouvement est Parmi les problèmes cruciaux en vision artificielle; En plus de la complexité due a la dimension spatiale des images d'une séquences prises séparément, on doit prendre en considération la dimension temporelle qui mette en relation les objets en mouvement d'une image a l'autre. L'expansion des IAD (Intelligence Artificielle Distribuée) et de la programmation orientée Agent (POA) a ouvert une nouvelle perspective quant a la conception des méthodes de suivi. La vision active, issue du mariage de la vision artificielle et des SMA, permet une résolution distribuée des problèmes de la vision artificielle. Elle consiste à concevoir des entités autonomes qui coopèrent pour aboutir à un objectif en commun. Notre travail, porte sur la conception d'un SMA pour le suivi d'objet dans les séquences d'images, l'ensemble des agents communiquent par envoi de messages. On a démontré la faisabilité d'un tel type de solution.

Mots Clés:

Suivi, Segmentation, Vision cognitive ,SMA.

TABLE DES MATIERES

Liste des Figures:

INTRODUCTION

Qu'est ce que la Vision Artificielle ?

La vision par ordinateur est une discipline très vaste au croisement de différents domaines des sciences de l'ingénieur tels que les mathématiques appliquées, l'intelligence artificielle, le traitement de signal, l'automatique et l'informatique afin de réaliser différentes tâches perceptives.

Au niveau théorique, elle a pour but de fournir des modèles de la perception visuelle des formes, du mouvement et des objets. Au niveau applicatif, elle a pour but d'automatiser différentes opérations telles que la détection des obstacles pour la navigation d'un Robot, la reconstruction tridimensionnelle d'une scène, la détection d'objet en mouvement, l'indexation d'images d'une séquence vidéo, la compression …etc.

Plusieurs raisons rendent la vision artificielle par ordinateur difficile :
Tout d'abord, la plupart des problèmes en vision par ordinateur sont mal définis, à savoir que même s'ils paraissent évidents, il est souvent très difficile de trouver une formulation technique judicieuse. D'autre part, même si ces problèmes techniquement bien définis, ils sont mal posés au sens de HADAMARD[1], ceci est du principalement à la perte considérable d'une quantité importante d'informations lors du passage du monde réel 3-D aux images 2-D.

Un autre obstacle qui rend les problèmes en vision artificielle plus ardus est le fait qu'on doit manipuler une multitude de données : le minimum d'additions et de multiplications par pixel nécessite un temps important sur une image de 500X500 Pixels (la complexité algorithmique est de l'ordre de $O(10^5)$. L'application d'un algorithme quadratique ou de complexité supérieure ralenti l'exécution d'une manière considérable. On doit donc, résoudre des problèmes difficiles avec des outils mathématiques préalablement étudiés pour leur coût.

Enfin, il convient de souligner qu'on ne peut pas parler d'image sans invoquer la perception : ce sont nos sens qui nous permettent de juger si une image en sortie est correcte " satisfaisante " ou pas. De nombreuses études d'ordre psycho-visuel ont été menées pour tenter d'expliquer nos processus d'analyse perceptif. Cette considération devra toujours être prise en compte, notamment dans l'analyse des résultats obtenus.

Un problème bien posé au sens de HADAMARD est un problème qui n'admet qu'une seule solution.

Vers la fin des années 70, DAVID MARR a proposé un modèle calculatoire pour le traitement et la représentation de l'information visuelle. Ce paradigme s'énonce comme suit :

1. A partir d'une ou plusieurs images, un processus d'extraction de caractéristiques produit une description en terme d'attributs bidimensionnels, ce niveau de représentation est appelé première ébauche.
2. La première ébauche constitue l'entrée d'un certain nombre de processus plus ou moins indépendants qui calculent des propriétés tridimensionnelles locales relatives à la scène, il s'agit d'une représentation centrée sur l'observateur appelée ébauche 2.5D. Ces processus opèrent sur une séquence d'image (Analyse de mouvement), une paire d'images (Stéréoscopie) ou sur une seule image.
3. L'ébauche 2.5D est mis en correspondance avec des connaissances 3-D préalablement établies afin de construire une description de la scène en terme d'objet et de relations entre ces objets, il s'agit maintenant d'une description centrée sur la scène indépendamment de la position de l'observateur.

En pratique, le paradigme de DAVID MARR se traduit par trois étapes de traitement : Segmentation, Reconstruction et Reconnaissance.

Notre travail traite l'un des problèmes les plus célèbres en vision artificielle qui est le suivi du mouvement d'objet dans les sequences d'images. Ce domaine ajoute aux difficultés inhérentes aux traitements d'une image unique sus-sités une difficulté liée à la dimension temporelle t (la première dimension étant la dimension spatiale qui concerne les différentes images prises séparément).

Notre travail, s'articule sur l'utilisation d'un systèmes multiagents dans le cadre du suivi du mouvement d'objet avec arrière plan fixe, plusieurs champs d'applications sont répertoriés dans ce cas de figure :la vidéosurveillance, la surveillance des trafics routiers, l'autonavigation …etc.

Dans le premier chapitre on présentera un Etat de l'art des techniques classiques (n'impliquant pas un SMA) utilisées pour le suivi, les techniques de segmentation d'images sont présentées au début (détection et segmentation des contours, segmentation en régions homogènes).

Le deuxième chapitre survole brièvement les systèmes multiagents, l'accent est mis sur les différentes architectures développées servant de modèles de construction des systèmes multiagents.

Le troisième, concerne la vision cognitive, domaine de recherche récent qui résulte de l'utilisation du paradigme agent pour résoudre les problèmes de vision artificielle et particulièrement ceux du suivi du mouvement.

Le quatrième chapitre, portera sur la conception détaillée de notre système (cadre applicatif, définition des agents, détection, Suivi, coopération, …etc.). L'architecture qu'on a propose est validée par l'implantation, les résultats obtenues et les temps de réponse acceptables ont démontré clairement la viabilité d'une telle solution.

Dans le bilan de la thèse, on présentera les perspectives et les extensions de notre travail.

CHAPITRE I.

I ETAT DE L'ART.

L'analyse du mouvement est un domaine de recherche particulièrement actif en vision par ordinateur, en effet, elle constitue une analyse dynamique des scènes (une dimension de plus figure dans une séquence d'image par rapport à l'ensemble des images de la même séquence prises séparément : la dimension temporelle) d'une manière générale, l'analyse du mouvement peut être scindée en deux problèmes: la détection du mouvement et son interprétation.

I.1 PROBLEMATIQUE.

Le suivi du mouvement d'objets dans les séquences d'images est l'un des champs d'application les plus abordé en vision artificielle, son but est de rechercher a reconstruire par calcul les évolutions d'une scène donnée au cours du temps. Sa résolution soulève une pléiade de problèmes de domaines, de type d'informations et de moyens algorithmiques. Plusieurs chercheurs subdivise le problème en trois étapes: Détection, reconnaissance et suivi proprement dit, l'étape de reconnaissance peut être supprimée si on s'intéresse uniquement au mouvement. Avant d'entamer le suivi il convient d'expliquer comment les images sont segmentées et ce dans le but de comprendre mieux les approches utilisées pour le suivi et particulièrement celles basées sur la segmentation.

I.2 SEGMENTATION DES IMAGES.

La segmentation est une étape pratiquement indispensable à toute application de vision artificielles (excepté quelques cas ou des modèles stochastiques sont utilisés), son rôle est de fournir aux couches supérieures des primitives évolués (au lieu qu'un simple flux de pixels) en vu d'autres traitements spécifiques. Les primitives les plus célèbres sont les contours et les régions homogènes. Les différents algorithmes et descriptions sont inspirés de [HM95] et les articles et documents liés.

I.2.1 Détection des contours

Une image peut être exprimée par une fonction de $\Re^2$ dans $\Re$ I(x,y) qui exprime l'intensité dans chaque pixel p(x,y) de l'image.

Filtrer une image revient à convoluer l'image (ou exactement sa fonction) I(x,y) avec une fonction f(x,y) qui s'appelle réponse impulsionnelle du filtre (convolution de f(x,y) avec δ(x)). Dans la cas continu l'image filtrée est données par:

$$I_f(x,y) = (I * f)(x,y) = \int_{-\infty-\infty}^{+\infty+\infty} f(x',y') I(x-x', y-y') dx' dy' = \int_{-\infty-\infty}^{+\infty+\infty} f(x-x', y-y') I(x', y') dx' dy' \quad ....(1)$$

Dans la cas discret, les domaines de I et de f sont bornés, soit le domaine de T [-N/2,N/2]et le domaine de f [-K/2,K/2]. La convolution de I par f s'écrit :

$$I_f(x,y) = (I * f)(x,y) \sum_{i'=-K/2}^{i'=+K/2} \sum_{j'=-K/2}^{j'=+K/2} f(i-i', j-j') I(i', j') \quad .....(2)$$

L'interprétation de l'expression (1) est comme suit : La convolution de I par *f* consiste à remplacer chaque point de I par une combinaison linéaire des pixels voisins, les coefficients de cette combinaison sont définis par le filtre*f*.

En chaque point de l'image on peut définir le gradient ∇par :

$I_x(x,y)= \sigma I_x(x,y)/\sigma x$, $I_y(x,y)= \sigma I_y(x,y)/\sigma y$,

Le vecteur gradient $\nabla I(x,y)=(I_x ,I_y)$ et son module par $||\nabla I(x,y)||= ((I_x)^{2+}(I_y)^2)^{1/2}$

Dans les implantations pratiques, le module du gradient peut être approximé par $Max(I_x ,I_y)$ ou $| I_x|+| I_y|$, La direction ϕ du gradient est donnée par : $\phi = arctg(I_y /I_x)$.

De point de vue géométrique, le module du gradient est nul partout dans l'image sauf dans les pixels qui présentent des discontinuité dans les niveaux de gris (ou dans les couleurs) par rapport aux pixels voisins, d'autre part, on peut ramener la détection des contours a calculer le passage par zéro de la dérivée seconde de la fonction d'image, l'opérateur de dérivation de deuxième degrés est le Laplacien défini par :

$L(x,y)= \sigma^2 I/ \sigma x^2 + \sigma^2 I/ \sigma y^2$.

L'extraction des contours revient à approximer soit le gradient ou le Laplacien de la fonction image par des filtres linéaires.

Pour remédier aux problèmes de bruit, un filtre de lissage est souvent appliqué avant le filtre de détection.

I.2.1.1 Détection par filtrage

- Filtres de Prewitt et Sobel [HM95] et al**:**

L'un des premiers algorithmes de détection des contours dans les images bidimensionnelles fut introduit par Prewitt, la détection de contours par cet algorithme revient a approximer le gradient par convolution de l'image avec deux masques :

$$H_i = \begin{vmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{vmatrix}, H_j = \begin{vmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{vmatrix}$$

L'image convoluée est calculée par : Ii[i,j]=Hi*I[i,j] et Ij[i,j]=Hj*I[i,j].

Il est possible de calculer la norme et l'orientation du gradient sur les deux images. Une variante du filtrage de Prewitt appelée filtre de Sobel se déduit en remplaçant les 1 et –1 par 2 et –2 respectivement dans Hi et Hj.

Les deux filtres de Prewitt et Sobel sont très sensibles aux bruits et ne permettent pas de détecter correctement tous les contours (particulièrement les coins).

Les opérateurs de gradient directionnels de Kirch [HM95] et al:

Kirch a proposé huit opérateurs de gradients directionnels dans les huit directions faisant un angle de k*π/4 (avec k variant entre 0 et 7) avec l'horizontal. L'expression des huit filtres est donnée par :

$$K_0 = \begin{vmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{vmatrix}, K_1 = \begin{vmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{vmatrix}, K_2 = \begin{vmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{vmatrix}, K_3 = \begin{vmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{vmatrix}$$

$$K_4 = \begin{vmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{vmatrix}, K_5 = \begin{vmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{vmatrix}, K_6 = \begin{vmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{vmatrix}, K_7 = \begin{vmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{vmatrix}$$

Ces filtres permettent de calculer la norme et l'orientation du gradient, de plus, ils apportent une amélioration considérable par rapport aux filtres de Prewitt & Sobel en particulier au voisinage d'un coin. Par ailleurs, ils restent très sensibles aux bruits.

Les opérateurs MDIF & NAGDIF:

Les opérateurs de MDIF: Les opérateurs de MDIF sont une combinaison d'un filtre moyenneur de noyau m et d'un opérateur de gradient défini par les masques H_0 et H_1 (d'où son appellation) avec :

$$m = \begin{vmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{vmatrix}, H_0 = \begin{vmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{vmatrix}, H_1 = \begin{vmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{vmatrix}$$

Après avoir effectuer les filtrages avec les combinaisons du filtre m avec chacun des deux autres masques, on calcul la norme et l'orientation du vecteur gradient. Cette combinaison de filtres est moins sensible aux bruits que les filtres précédents, mais le traitement reste assez local (toujours sur une fenêtre de 3x3), de plus, les coins (discontinuités d'ordre supérieur) ne sont pas correctement détectés, mais auront une forme arrondie.

Les opérateurs de NAGDIF.

L'opérateur de NAGDIF est plus complexe par rapport à MDIF, car il effectue tout d'abord un lissage linéaire de l'image en utilisant le filtre de NAGAO [HM95] et al qui opère sur neuf domaines Dk (K varie de 1 à 9) appartenant à une fenêtre de (5x5) centrée sur le pixel courant:

$$D_1 = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix}, D_5 = \begin{vmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix}, D_9 = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Les six autres domaines se déduisent facilement par symétrie, en effet, D2, D3 et D4 s'obtiennent par rotation successive de D1 par une angle de $\pi/2$, de même D6, D7 et D8 sont obtenus par rotation successive du domaine D5 par une angle de $\pi/2$. En chaque pixel s, on calcul la moyenne $\mu(Dk)$ et la variance Var(Dk) pour chacun des domaines Dk, on remplace en suite la valeur du niveau de gris en s par la moyenne du domaine ayant la plus faible variance. Cet opérateur permet d'une part de minimiser le bruit et d'autre part d'amplifier les contrastes entre différents objets.

Opérateur de NAGDIF:

Après avoir lisser l'image par le filtre de NAGAO (sans considérer le domaine D9). Cocquerez & Devars [HM95 et al] et al applique un opérateur différentiel du premier ordre qui détecte les variations du niveau de gris moyen dans les huit directions perpendiculaires à celles de Freeman [HM95 et al].

Un pixel est considéré comme élément d'un contour si :

Max { |B[i,j]-B[i-ε1,j-ε2]|}>seuil où :

B correspond à l'image lissée

ε1 et ε2 sont deux valeurs entières appartenant à [-1,1].

D'autres opérateurs utilisent le gradient.

I.2.1.2 Les opérateurs basées sur le Laplacien.

Dans le cas discret des images 2-D, l'analogue du Laplacien s'écrit sous la forme :

Δ I(i,j)= 4*I(i,j)-I(i+1,j)-I(i-1,j)-I(i,j+1)-I(i,j+1)

Ceci revient à convoluer l'image avec le filtre:

$$L = \begin{vmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 1 & -1 & 0 \end{vmatrix}.$$

I.2.1.3 Autre approche de détection des contours : approche de Haralick

L'idée de cet algorithme est d'approximer localement le signal discret (bidimensionnel) formant l'image par une fonction dérivable à deux variables puis de dériver cet image, la fonction utilisée souvent pour approximer localement l'image est une fonction polynomiale de troisième degrés : $f(x,y)=k_1+ k_2x+ k_3y+ k_4x^2+ k_5xy + k_6y^2+ k_7x^3+ k_8x^2y+k_9xy^2+ k_{10}y^3$

Le problème qui reste à résoudre donc, est celui de déterminer les coefficients k_i, ceci revient toujours à résoudre un système d'équations linéaire à 10 inconnus ce qui implique l'utilisation d'un minimum de 10 pixels.

I.2.1.4 Extraction des extrema locaux du gradient.

Dans le but de ne garder que les points de contours significatifs (Réduire davantage le bruit), on effectue une sélection des points dont la norme est supérieure à un seuil fixé:

$N(x,y)= (\delta I/\delta x)^2 + (\delta I/\delta y)^2 > S$ (S est un seuil fixé)

Dans le cas des images où la norme du gradient aux points des contours varie fortement, cette méthode se révèle inefficace, un moyen de détourner cette lacune est d'extraire non pas les points de norme de gradient élevée, mais les extrema locaux de la norme du gradient dans la direction des gradient : Soit un point M de gradient G(M) et d une distance seuil, M_1 et M_2 deux points de la droite passant par M et dont la direction est celle du gradient G(M) ; M_1 est pris dans le sens du gradient, M_2 dans le sens inverse

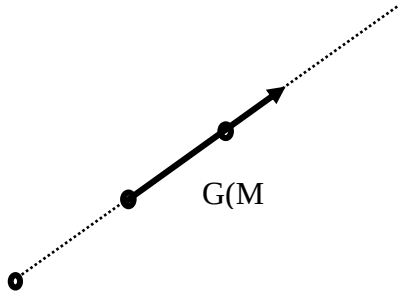


Fig I-1 extraction des extrema locaux.

Le point M est sélectionné si $N(M)>N(M_2)$ et $N(M)\geq N(M_1)$.

I.2.1.5 Segmentation des contours

Dans la plupart des applications, la description matricielle des contours fournis par un opérateur quelconque n'est pas suffisante, car les mises en correspondances utilisées dans les niveaux supérieurs requièrent des représentations plus enrichies (droites, courbes, formes géométriques bien spécifiées,…etc).

Soit C={c_i , i=0,…n} l'ensemble des points d'une chaîne où $c_i[x_i,y_i]$ est le $i^{ème}$ point de la chaîne avec ses coordonnées dans l'image. Le problème de segmentation consiste à trouver une partition S constituée d'un ensemble de segments dont chacun d'eux optimise le mieux possible un critère d'homogénéité au sens d'une approximation analytique, les couples résultants (segment, expression analytique) constitueront la segmentation recherchée.

Approximation d'un segment par une droite :

L'équation d'une droite étant donnée par :

x sin(θ) +y cos(θ) = d (θ est l'angle de la droite avec l'axe X, d distance du point de repère à la droite)

La distance d'un point $[x_i,y_i]$ à cette droite est : e_i=| x sin(θ)+y cos(θ)-d|

Le problème est de trouver les paramètres θ et d qui minimisent la grandeur :

$E = \sum_{i=0}^{n} (e_i)^2$ pour tous les points du segment, après calcul on trouve :

Equation de la droite : d =V_x sin(θ) +V_y cos(θ) ; θ = ½ arctan ($2V_{xy}$/ V_{yy}-V_{xx}),

Avec : $V_x = \frac{1}{n}\sum_{i=0}^{n} x_i$, $V_y = \frac{1}{n}\sum_{i=0}^{n} y$, $V_{xx} = \frac{1}{n}\sum_{i=0}^{n} (x_i - V_x)^2$, $V_{yy} = \frac{1}{n}\sum_{i=0}^{n} (y_i - V_y)^2$,

$V_{xy} = \frac{1}{n}\sum_{i=0}^{n} (x_i - V_x)(y_i - V_y)$

Approximation d'un segment par un arc de cercle :

L'équation d'un cercle étant donnée par : $(x-a)^2+(y-b)^2=r^2$, l'équation peut être réécrite autrement : $x^2+y^2+Ax+By+C=0$ avec A=-2a, B=-2b, C= $a^2+b^2- r^2$

La distance algébrique entre un point $[x_i,y_i]$ et le cercle considéré est :

$e_i = \left| x_i^2 + y_i^2 + Ax_i + By_i + C \right|$

Le problème dans cas revient à déterminer les paramètres A,B,C qui minimise le critère quadratique Pour les points du segment.

$$E = \sum_{i=0}^{n} (e_i)^2$$

La solution de ce problème de minimisation est équivalent à la résolution du système linéaire suivant : $U_{xx}A+U_{xy}B+U_xC=-U_{xxx}-U_{xyy}$, $U_{xy}A+U_{yy}B+ U_yC=-U_{xxy}-U_{yyy}$, $U_xA+U_yB+nC=-U_{xx}-U_{yy}$

Avec $U_x = \frac{1}{n}\sum_{i=0}^{n} x_i$, $U_y = \frac{1}{n}\sum_{i=0}^{n} y_i$, $U_{xx} = \frac{1}{n}\sum_{i=0}^{n} x_i^2$,$U_{yy} = \frac{1}{n}\sum_{i=0}^{n} y_i^2$,$U_{xy} = \frac{1}{n}\sum_{i=0}^{n} x_i\, y_i$

$$U_{xxx} = \frac{1}{n}\sum_{i=0}^{n} x_i^3 \text{ ,} U_{xxy} = \frac{1}{n}\sum_{i=0}^{n} x_i^2 y_i \text{ , } U_{xyy} = \frac{1}{n}\sum_{i=0}^{n} x_i\, y_i^2 \text{ , } U_{yyy} = \frac{1}{n}\sum_{i=0}^{n} y_i^3$$

Synthèse : Algorithme de découpage récursif.

- en segment de droite:

1- Pour une chaîne de point : est-ce un segment de droite

2- Si oui aller à 4.

3- Sinon diviser la chaîne en deux sous chaînes et aller à 1 pour chaque sous-chaîne.

4- Fin

La manière la plus naturelle de diviser la chaîne est de choisir le point le plus éloigné de la droite définie par les deux extrémités de la chaîne.

- Extension de l'algorithme aux arcs

Si on veut prendre en compte les cercles dans le découpage l'algorithme devient :

1- Pour une chaîne de point : est-ce un segment de droite

2- Si oui aller à 6.

3- Sinon : est-ce un arc de cercle

4- Si oui aller à 6

5- diviser la chaîne en deux sous chaînes et aller à 1 chaque sous-chaîne

6- Fin.

I.2.2 Segmentation En Régions

La segmentation en régions homogènes consiste à partitionner l'image en zones d'intérêts correspondants à des objets de la scène requis par une application donnée. Le problème de la segmentation en régions s'énonce comme suit :

Etant donné :

- Un ensemble d'entités (souvent des pixels)
- Un ensemble d'attributs caractérisant ces entités (luminosité …)
- Des relations topologiques entre ces entités (connexité par exemple)

- Des attributs relationnels entre ces entités (relation d'adjacence par exemple).

On cherche alors une (ou plusieurs) partition sur cet ensemble d'entités ayant des propriétés intéressantes par rapport aux attributs et aux relations topologiques et relationnelles. Cette problématique est caractérisée par :

- La définition des critères d'homogénéité ➔ Modélisation analytique
- Un algorithme de utilisant ces critères ➔Modélisation algorithmique.

Formulation mathématique:

Etant donnée une image I, la segmentation de I par un prédicat (critère) P est une partition $S=\{R_1, R_2, \ldots, R_i\}$ telle que :

- $\bigcup_{i=1}^{n} R_i = I$.
- R_i est connecxe
- $P(R_i)$ = vrai $\forall$ i
- $P(R_i \cup R_j)$ = faux $\forall$ i ≠ j

Ces quatre conditions ne définissent pas, en général, une segmentation unique, cette lacune peut être contourner en ajoutons une contrainte d'optimisation d'une fonction C caractérisant la qualité d'une segmentation. Donc, s'ajoute aux quatre conditions la contrainte suivante :

- Parmi toutes les segmentations possible S_i on cherche une segmentation optimale S_{opt} telle que : $C(S_{opt}) \leq C(S) \; \forall \; S \in S_i$

I.2.2.1 Segmentation en régions par classification.

Les méthodes s'appuyant sur cette approche, déterminent d'abord une partition de l'espaces de luminances en utilisant les niveaux de gris : chaque pixel est associé à une classe de niveaux de gris. Les régions sont définies par suite par les ensembles maximaux de pixels connexes appartenant à la même classe. Souvent, les classes sont déterminées statistiquement à partir de l'histogramme cumulé des niveaux de gris présents dans l'image, cette procédure donne des bonnes performances dans les images contenant peu d'objets distincts ayant des niveaux d'intensité suffisamment différents, plusieurs améliorations de cette méthode ont été proposées :

Chow & Kaneko [HM95] et al utilisent des histogrammes pour des régions locales (voisinage de 56*65 par exemple) pour des blocs d'image définis à priori. Cette variante permet de calculer des seuils locaux et adaptatifs à chaque bloc de pixels. Weszca, Nagel & Rosenfeld [HM95] et al calculent les histogrammes seulement pour

les pixels ayant une faible valeur du Laplacien (seules les pixels en dehors des zones de fort gradient sont pris en compte, Watanabe [HM95] et al propose de choisir une valeur seuil maximisant la somme des gradients calculés sur tous les pixels dont le niveaux de gris égal à la valeur du seuil.

Ces méthodes fonctionnent bien pour les images "Parfaites" contenant peu d'objets entièrement distincts par leur intensité. Malheureusement, souvent dans la réalité les images contiennent un nombre important d'objets et bruités. Dans ce cas, ces méthodes se révèlent inefficaces, des relations d'ordre spatial doivent être utilisées.

I.2.2.2 Segmentation par croissance de régions.

Ce type d'approche, consiste à regrouper itérativement des ensembles de points connexes en régions plus importantes en utilisant des propriétés d'homogénéité. Cette approche regroupe deux grandes classes de méthodes : celles qui opèrent par "agrégation de points" et celles qui opèrent par "regroupement itératif de points"

Agrégation des points: Cette méthode consiste à l'association d'un vecteur de propriétés à chaque pixel, deux pixels sont regroupés si leurs vecteurs sont assez similaires. Le résultat de la segmentation est l'ensemble des composants connexes déterminés. Plusieurs vecteurs de propriétés on été proposés: Le premier consiste a considéré seulement le niveau de gris comme vecteur de propriété et la différence comme mesure, deux pixels sont agrégés si la différence de leur niveau de gris est inférieur à un seuil, une autre mesure proposé pour ce même type de vecteurs est la norme des différences considérée avec tous les voisins du pixel en cours, Nagao & Matsuama [] utilisent se basent sur les composants (R,V,B) comme vecteur de propriétés, un pixel est regroupé avec une région si la norme SUP entre son vecteur et un vecteur d'un pixel quelconque de cette région est majoré par un seuil. D'autres critères plus sophistiqués sont envisageables, souvent on utilise les informations issues d'un détecteur de contour comme vecteur de propriétés.

L'inconvénient de cette approche réside dont le fait qu'un seul pixel ne véhicule pas assez d'information pour le classer convenablement, pour cela, d'autres approches sont utilisées.

Regroupement itératif d’ensembles de points:

Dans cette approche, on défini tout d’abord une partition initiale de l’image, puis on regroupe successivement les régions on utilisant des critères globaux entre des ensembles de point et non seulement entre un ensemble et un pixel isolé.

Une première méthode fut introduite par Muerle & Allen [HM95] et al: deux ensembles de points adjacents sont regroupés si leurs distribution de niveaux de gris sont similaires, un test statistique (tel que celui de Smirnov-Kolmogorov) est utilisé.

Bryce et Fet Fennema [HM95] et al effectuent la croissance de régions en partitionnant l'image en ensembles initiaux de points de même intensité, les couples de régions adjacentes sont regroupés séquentiellement dont une fraction significative de frontière a un contraste faible, Une stratégie de "Split & Merge" utilisant les Quadtrees est ensuite mise en œuvre, ainsi, on obtient un ensemble de régions dont l'amplitude de variation des luminances est majorée par un seuil.

La plupart de ces méthodes regroupent itérativement tous les couples de régions adjacentes vérifiant certains critères. Le processus s'arrête quand plus aucun couple de région ne vérifie ces critères.

I.2.2.3 Segmentation de région par fermeture de contours.

Cette méthode sert plutôt à détecter les régions qu'à leur segmentation, cependant, dans la plupart des cas réelles, les régions homogènes sont délimitées par un seul contour, ce qui rend cette méthode valable aussi pour la segmentation des régions. L'inconvénient de cette méthode réside dans le fait que les contours issus d'un détecteur quelconque sont rarement fermés, ce qui rend la propriété de la connexité des région non vérifiée, un algorithme de fermeture des contours a été proposé comme suit:

Description de l'algorithme de fermeture des contours :

L'image est balayer ligne par ligne par un opérateur de 3x3, les extrémités des contours sont identifiée par un simple examen du voisinage 3x3 d'un point. Selon la topologie de l'extrémité du contour, 3 voisinages sont à examiner comme le montre la figure ci-après :

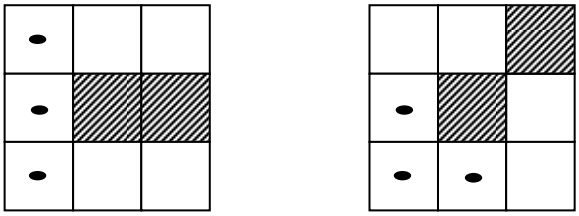


Fig. I-2. Exemple de voisins d'extrémités de contour.

Légende :

Points de contours

Voisins a examiner

I.3 DETECTION DU MOUVEMENT APPARENT.

Dans cette étape on procède à l'extraction brute de l'information du mouvement. Le mouvement apparent peut se présenter sous deux formes différentes: Le champ de vitesse (flot optique) ou le champ de déplacement. Ces de formes ont des caractéristiques en commun:

- Elles sont extraites directement de l'image (d'où l'appellation mouvement apparent i.e: perceptible directement de l'image).
- Elles sont 2D.
- Elles ne peuvent être considérées que des informations brutes à partir desquelles on déduit le mouvement réel 3D

Cependant ces deux formes d'analyse du mouvement conduisent à deux approches de conception bien distinguées: l'une appelée approche continue et l'autre l'approche discontinue.

I.3.1 Classification des approches:

On peut distinguer trois grandes classes utilisées pour le suivi du mouvement:

- **Approche continue:** La première approche, appelée également approche différentielle, s'appuie sur les dérivées partielles spatiales et temporelles à chaque position de l'image. Le mouvement apparent détecté (champ de vitesse) est appelé flot optique. Un avantage remarquable de cette approche est qu'elle fournie une information de mouvement dense (valorisée à chaque point de l'image)

Les hypothèses fondamentales de cette approche sont [Ser96]:

(H1): L'intensité de l'image dans un point 3D reste constante entre deux instants.

(H2): L'intervalle de temps est très faible.

Tandis que la faisabilité de (H2) est discutable (mouvement <2pixels), (H1) s'avère une hypothèse trop forte, elle est dans la plupart des cas violée (le flot optique n'est pas égale à la projection de la vitesse 3D). Quant aux calculs, on peut citer des problèmes délicats tels que la validité de l'approximation des équations et des dérivées partielles, l'instabilité de l'opérateur de différentiation, l'utilisation de contraintes supplémentaires avec les problèmes des occultations.

Approche discontinue (discrète): La deuxième approche est dite discrète car on ne considère plus ici la variation continue spatio-temporelle mais un déplacement apparent discret d'une trame à la suivante dans une séquence. Pou cela, des appariement d'indices

2D sont effectuées, ces indices 2D résultant en générale d'une forte variation d'intensité tels que points d'intérêts, contours , segments de droite et courbes …etc.

L'approche discrète, via l'utilisation des primitives, autorise des déplacements plus importants que ceux imposés dans l'approche continue. Le processus d'appariement dans ce cas est souvent ramené au problème de mise en correspondance.

Les méthodes récentes de détection du mouvement utilisent des longues séquences d'image car:

- Leur utilisation est plus significative pour des applications temps réel.
- Le faible mouvement impliqué par des longues séquences réduit considérablement l'air de recherche d'une primitive et facilite la levée des ambiguïtés inhérentes aux occultations.
- Remédier efficacement aux bruits.

Approche Région: L'approche région est une variante de l'approche continue, elle se caractérise par l'extraction dans l'image courante de régions appelées "Blob". Ces régions sont définies par des ensembles de pixels connexes.

Le cas le plus fréquent de cette approche est l'utilisation d'une image de référence I_0 Les régions en mouvement sont obtenues alors par différence entre l'image courante et l'image de référence. L'inconvénient majeure de cette méthode est la difficulté rencontré quant a la mise a jour de l'image de référence.

Une autre méthode utilisée, consiste a combiner l'image en cours avec la différence entre les deux dernières images successives.

I.3.2 Interprétation du mouvement:

A partir des informations obtenues par le mouvement apparent, l'interprétation consiste à inférer le mouvement 3D, deux problèmes cruciaux sont à relever:

I.3.2.1 Estimation du mouvement 3D

L'estimation du mouvement 3D consiste à estimer les paramètres effectifs du mouvement 3D à partir des propriétés géométrique 2D des primitives utilisées. Une méthode d'estimation est appréciée selon sa capacité d'estimer avec précision les paramètres du mouvement, de prendre en compte l'incertitude dans les données qui est indispensable et de gérer les anomalies imprévisibles qui peuvent surgir lors de l'application de l'algorithme de mise en correspondance temporelle. Dans ce sens, on trouve une bibliographie abondante traitant ces aspects.

I.3.2.2 Mouvement et segmentation:

Lorsqu'il existe plus d'un mouvement 3D dans une scène, il est intéressant (voir même indispensable) de séparer les entités effectuant le même mouvement en classes, ce processus est appelé la segmentation des séquences au sens du mouvement. Malheureusement cette segmentation qui s'appuie uniquement sur l'information mouvement et non fiable, plusieurs auteurs suggèrent de combiner le mouvement avec d'autres information statiques (courbure, gradient, …etc.) afin de mieux cerner le problème de segmentation des séquences.

I.3.3 Récapitulatif.

Le tableau ci-après donne une vue générale sur l'analyse de mouvements (applications, approches, classification…etc).

Applications	Codage d'images TV Véhicule Autonome Robotique Imagerie médicale Surveillance du trafic Biomédical Militaire (suivi de cibles mobiles…etc.)
Données d'entrée	Séquences monoculaires Séquences binoculaires Séquences 3D (Sonar, télémètre LASER…)
Primitives choisies	Points d'intérêts Segment de droite Courbes Contours Région
Problèmes	Détection du mouvement apparent Estimation du mouvement Reconstitution 3D de scène Segmentation de scène Interprétation de scène Suivi
Type de mouvement	Scène statique & capteur mobile Scène dynamique & capteur fixe Scène dynamique & capteur mobile
Approches	Continue (flot optique) Discontinue (discrète)

I.4 PANORAMA DES APPROCHES DE SUIVI

I.4.1 Approches basées sur la segmentation.

Les approches basées sur la segmentation effectuent d'abord une segmentation de chaque images pour extraire des primitives choisies, puis procèdent a un appariement entre ces primitives pour deux occurrences d'images successives. Les primitives peuvent être des segments de droites, des courbes, des points d'intérêts, des régions...etc., les plus utilisées sont les segments de droite a cause de leur représentation simple et détection rapide.

I.4.1.1 Exemple: Suivi des segments de droite dans les séquences d'images.

Plusieurs travaux ont traite le suivi des segments de droite, dans [Che95] et [GBV98], le principe utilisé pour le suivi de segment de droite dans les séquences vidéo repose sur deux grandes phases fortement mêlées :

- Prédiction de la position du segment
- Appariement avec le segment le plus proche dans l'espace de paramètres.

Le filtre de KALMANN permet de faire un suivi temporel, car il intègre les notions de prédiction et de correction selon les mesures réelles.

Représentation d'un segment: La représentation choisie pour un segment de droite est celle ou le segment est caractérise par: son point milieu $P_m(X_m,Y_m)$, sa longueur l et son orientation θ**.** Dans cette représentation, la longueur l et l'orientation θ sont décorrolés des autres variables, par ailleurs, le point milieu est de covariance minimale parmi tous les points du segment.

Si $\sigma_{//}$ et $\sigma_{\perp}$ représentent l'incertitude sur la position des points extrémités dans les direction respectivement parallèle et orthogonale au segment de droite, la matrice de covariance du point milieu s'exprime par

$$\Lambda = \begin{pmatrix} \sigma_x^2 & \sigma_{xy}^2 \\ \sigma_{xy}^2 & \sigma_y^2 \end{pmatrix}$$

avec :

$$2\sigma_{mxx}^2 = \sigma_{//}^2 \cos(\theta)^2 + \sigma_{\perp}^2 \sin(\theta)^2$$

$$2\sigma_{myy}^2 = \sigma_{\perp}^2 \cos(\theta)^2 + \sigma_{//}^2 \sin(\theta)^2$$

$$2\sigma_{mxy}^2 = \left(\sigma_{//}^2 - \sigma_{\perp}^2\right)\sin(\theta)\cos(\theta)$$

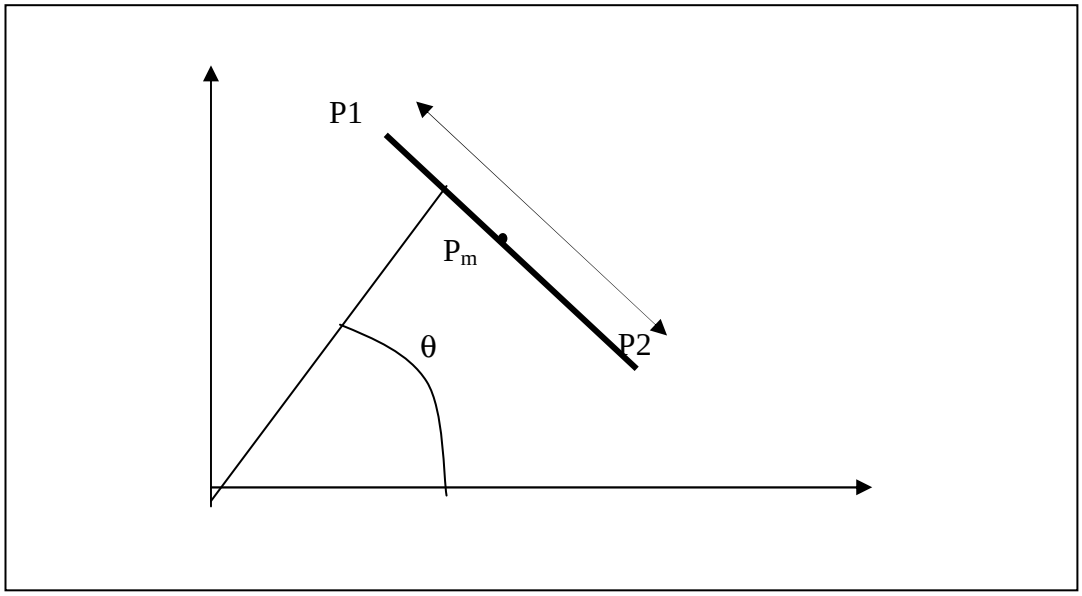


Fig I-3 Paramètres de la représentation d'un segment

Suivi de segments par filtrage: Pour suivre des segments dans les séquences d'images par filtre de Kalman, l'implantation de ces filtres dépend de la manière dont les paramètres de la représentation sont perçus.

+ Filtres scalaires (décorrélés): Si on considère que les paramètres de la représentation sont indépendants, dans ce cas quatre filtres décorrélés doivent être mis en œuvre (un filtre par composante de la représentation x_m,y_m,l,θ). On choisi un modèle de mouvement à accélération constante pour les coordonnées du point de milieu x_m,y_m et à vitesse constante pour la longueur l et l'orientation θ. Les vecteurs d'état ont par conséquence la forme :

$$X_t^{x_m} = \begin{pmatrix} x_m \\ \dot{x}_m \\ \ddot{x}_m \end{pmatrix}, X_t^{y_m} = \begin{pmatrix} y_m \\ \dot{y}_m \\ \ddot{y}_m \end{pmatrix}, X_t^{l} = \begin{pmatrix} l \\ \dot{l} \end{pmatrix}, X_t^{\theta} = \begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix}$$

Les matrices du modèle pour chacune des composantes du point de milieu s'écrivent sous la forme :

$$C = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}, \Phi = \begin{pmatrix} 1 & \Delta t & \frac{(\Delta t)^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix}$$

Φ est la matrice de dynamique (modèle choisi à accélération constante), C la matrice de passage du vecteur d'état au vecteur X_t de mesure V_t, Le modèle dévolution correspondant s'écrit $X_{t+1} = \Phi.X_t$ et le modèle de mesure $V_t = C.X_t$ car on ne mesure que la position.

Le choix d'un modèle à vitesse constante pour la longueur et l'orientation au lieu d'un modèle à paramètres constants s'explique par les problèmes d'occlusion en ce qui concerne la longueur et par les rotations qui animent les segments en 2D en ce qui concerne l'orientation.

Pour déterminer l'aire de recherche, et pour simplifier le problème on suppose que le mouvement est à vitesse constante, dans ce cas le vecteur d'état prédit à l'instant t et la matrice associée à l'aire de recherche s'expriment par :

$$X_{t+1} = \Phi_t.X_t$$

$$U_t = C_t.P_{t/t-1}C_t^T + R_t$$

Le vecteur d'état étant composé de la position et de la vitesse.

$$X = \begin{pmatrix} x \\ \dot{x} \end{pmatrix}$$

Les différentes matrices du filtre s'écrivent sous la forme :

$$R_t = \sigma_m^2 = \sigma_{//}^2 = \sigma_{\perp}^2, C = \begin{pmatrix} 1 & 0 \end{pmatrix}, P = \begin{pmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{pmatrix}$$

R_t représente la covariance ud bruit de mesure, C représente la matrice de passage du vecteur d'état au vecteur de mesure et P la martice de covariance d'erreur. On obtient l'expression finale de U_t :

$$U_t = p_{11} + \sigma_m^2$$

Cette équation définit l'aire de recherche, les variations de X_t étant comprises dans l'intervalle X_t -U_t

et X_t +U_t.

+ Filtres vectoriels (corrélés): Considérer les paramètres de représentation du segment décorrélés est une hypothèse qui peut ne pas être vérifiée. Dans la plupart des cas les coordonnées du point milieu sont corrélées.

Les différents vecteurs et matrices utilisés par ce filtre s'expriment par :

$$V = \begin{pmatrix} x \\ y \end{pmatrix}, X = \begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \\ y \\ \dot{y} \\ \ddot{y} \end{pmatrix}, C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}, \Phi = \begin{pmatrix} 1 & \Delta t & \frac{(\Delta t)^2}{2} & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{(\Delta t)^2}{2} \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Les différentes matrices d'incertitude (R matrice d'incertitude sur la mesure, Q la matrice d'incertitude sur la dynamique et P la covariance d'erreur du filtre) s'expriment sous la forme suivante :

$$R = \begin{pmatrix} \sigma_{mxx}^2 & \sigma_{mxy}^2 \\ \sigma_{mxy}^2 & \sigma_{myy}^2 \end{pmatrix}, Q = \begin{pmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_v^2 & 0 \\ 0 & 0 & \sigma_a^2 \end{pmatrix}, P = \begin{pmatrix} p_{11} & \dots & p_{16} \\ \dots & \dots & \dots \\ p_{16} & \dots & p_{66} \end{pmatrix}$$

Les coefficients de la matrice R sont ceux définis pour la covariance du segment, Les deux filtres scalaires pour la longueur et l'orientation sont définis comme dans le cas précédent.

Apres calcul, l'aire de recherche correspond a une ellipse dont l'équation est:

$$(X - X_t)^T . U_t^{-1} . (X - X_t) = 1$$

Après développement on obtient :

$$\frac{p_m+\sigma_m^2}{(p_m+\sigma_m)^2-(p_c+\sigma_c)^2}(x^2+y^2)-\frac{2.(p_c+\sigma_c^2)}{(p_m+\sigma_m)^2-(p_c+\sigma_c)^2}xy=1$$

La surface de l'éllipse est donnée par :

$$S=\pi\left((p_m+\sigma_m^2)^2-(p_c+\sigma_c^2)^2\right)$$

Suivi de segments par contraintes d'appariements: Cette méthode de suivi est complémentaire à la notion de filtrage, car l'utilisation brute d'un filtre de Kalmann ne prend pas en compte les cas d'appariements multiples, de coupure, de fusion des segments et supposent une faible disparité entre deux images consécutives. Pour rendre le suivi par filtrage plus robuste, des contraintes sont imposées entre le segment et son appariement. L'idée consiste à faire en sorte que pendant le filtrage, le vecteur d'état vérifie sans cesse ces contraintes (réinitialiser le filtre par les estimées obtenues en appliquant les contraintes).

Les contraintes imposées sont de quatre types : géométriques, de mouvement et d'intensité Les auteurs ont proposés six contraintes (d'autres contraintes peuvent être insérées) étant donné S le segment initial avec son point milieu M(x_1,y_1), S' est le segment apparié (candidat) et son point milieu M'(x_2,y_2) :

La première contrainte est que la distance entre M' et la droite engendrée par le segment S soit inférieur à un seuil k :

$$d=\frac{|ax_2+by_2+c|}{\sqrt{a^2+b^2}}\prec K_1$$

La deuxième contrainte est que la distance entre les deux points milieu M et M' soit inférieure à un seuil fixé

$$d=\sqrt{(x_1-x_2)^2+(y_2-y_2)^2}\prec K_2$$

La troisième condition est que l'angle formé par les droites engendrées par les deux segments S et S' soit inférieure à un seuil fixé : soit $P_1(x_1,y_1)$,$P_2(x_2,y_2)$ les deux points extrémités du segment S, $p'_1(x'_1,y'_1)$,$p'_2(x'_2,y'_2)$ les deux points extrémités de S', cette contrainte s'exprime par :

$$d=\frac{\left|(x'_2-x'_1).(y_2-y_1)-(y'_2-y'_1).(x_2-x_1)\right|}{\sqrt{(x'_2-x'_1)^2+(y'_2-y'_1)^2}\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}<K_3$$

La quatrième condition est que la distance entre les deux boules B de centre M et de rayon l/2 et B' de centre M' et de rayon l'/2 soit inférieure à un seuil donné

$$d = \sqrt{(x_M - x_{M'})^2 + (y_M - y_{M'})^2} - \frac{1}{2}(l + l') < K_4$$

Les quatre contraintes précédentes sont des contrainte géométrique, une cinquième contrainte concernant le mouvement du segment peut être prise en considération : soit S'' le segment apparié de S' dans l'image suivante et son point milieu M''($x_{M'}$,$y_{M''}$), Cette contrainte de mouvement impose que le flot des points milieu f_1,f_2 soit dans le même sens, étant donnée l'équation de la droite engendrée par le segment S' : ax+by+c=0, la cinquième contrainte peut être exprimée par :

$$d = (ax_M + by_M + c)(ax_{M''} + by_{M''} + c) \prec 0 \ .$$

La sixième condition introduite par les auteurs est de type intensité, les moyennes d'intensité à droite et à gauche des segments S et S' soient assez proche, soit $I_d(S)$, $I_g(S)$ les moyennes respectivement droite et gauche des intensités des voisinages rectangulaires au segment S, $I_d(S')$, $I_g(S')$ même notation pour le segment S'. Le sixième critère peut s'exprimer de la manière suivante :

$$d = |I_d(S) - I_d(S')| + |I_g(S) - I_g(S')| \prec K_5$$

avec :

$$I_g = \frac{1}{a.b}\sum Rect_{g[a,b]} I(x,y), I_d = \frac{1}{a.b}\sum Rect_{d[a,b]} I(x,y).$$

La figure suivante illustre les différentes grandeurs d à minimiser dans les contraintes.

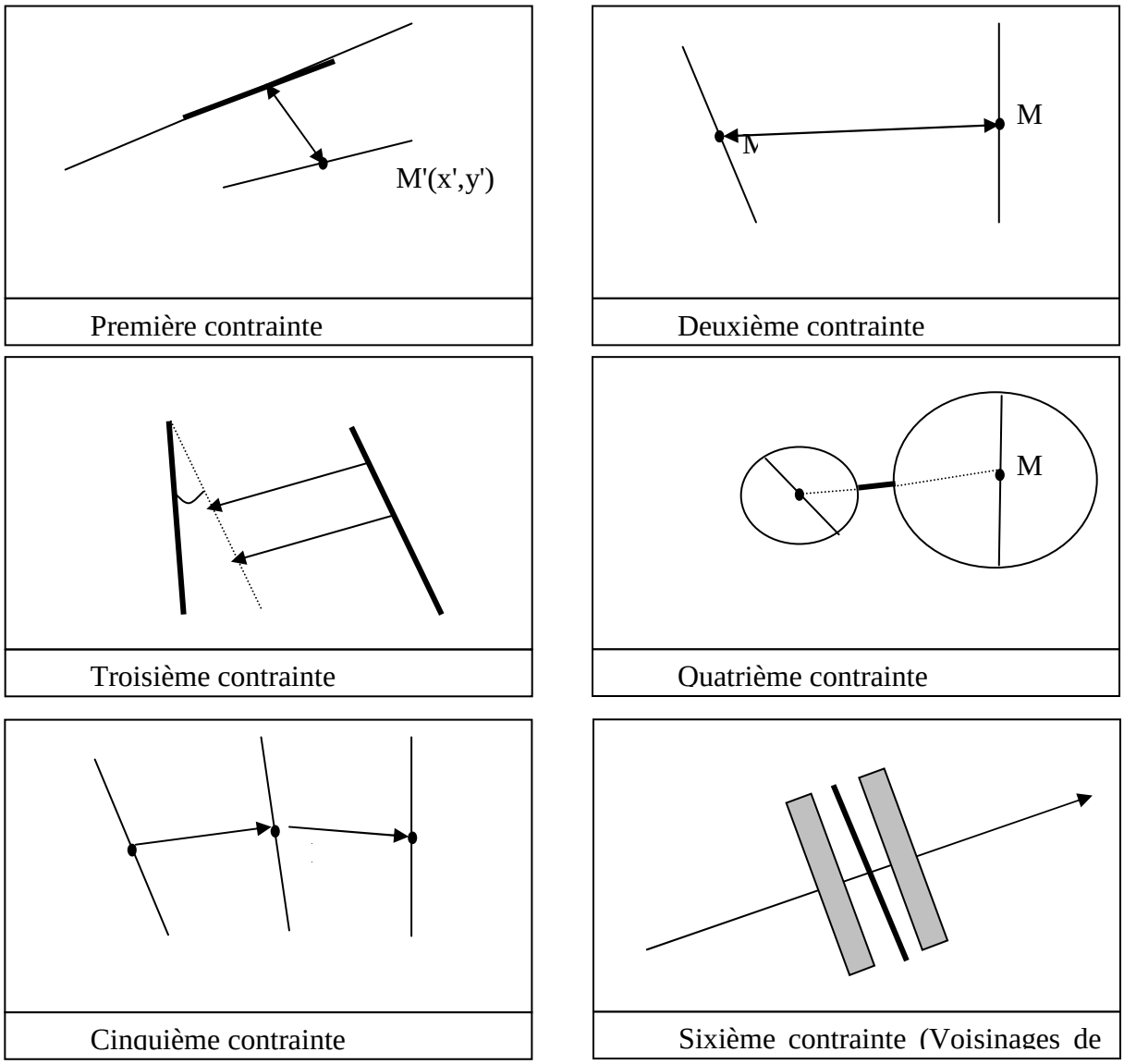


Fig I-4 Schéma illustratif des différentes grandeurs utilisées par les contraintes

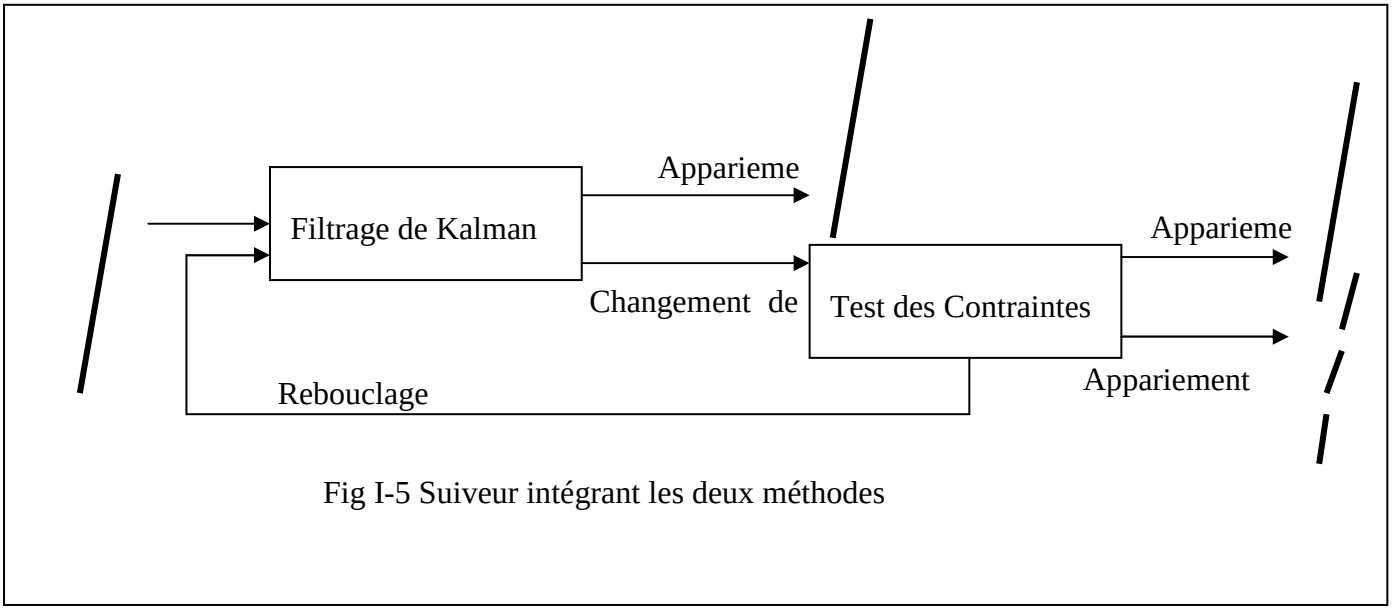


Fig I-5 Suiveur intégrant les deux méthodes

I.4.1.2 Détection des formes rigides

*** Choix d'un champ affine pour un ensemble de segments**.

Une fois les segments d'une image sont appariés complètement avec leurs estimée dans l'image suivante, et afin de déterminer les différents ensembles de segments constituant des objets effectuant un mouvement rigide, il est nécessaire de faire correspondre à un ensemble de segment une représentation unique au sens du mouvement. Ce champ peut être exprimé par :

$$\vec{\omega}(x,y) = (c_{11}x + c_{12}y + a).\vec{i} + (c_{21}x + c_{22}y + b).\vec{j}$$

Ce champ est ainsi représenté par six paramètres. Pour estimer ces derniers, introduisant un paramètre très important (notamment dans la construction des cartes de mouvement) définit par :

$$\beta(M) = \dot{d} - e(M).v$$

d représente la différence dans la distance par rapport à l'origine du segment S contenant le point M et son apparié, v représente la variation de l'orientation du segment et de son apparié et enfin e(M) est l'excentricité du point M par rapport au segment défini par :

$$e(M) = y\cos(\theta) - x\sin(\theta)\ .$$

Les six paramètres du champ affine sont déterminés par minimisation du critère suivant :

$$J = \min_{a,b,c_{11},c_{12},c_{21},c_{22}} \sum_{i=0}^{n} + \left(\left(\beta(e_1^i) - \vec{\omega}(e_1^i).N^i \right)^2 + \left(\beta(e_2^i) - \vec{\omega}(e_2^i).N^i \right)^2 \right)^2$$

e_1,e_2 représentent les deux extrémités du segment S^i**,** $\mathbf{N}^i$ représente la normale par rapport au segment. Chaque segment fournit deux résidus dans le critère J, et par conséquent, le nombre minimum de segments nécessaires pour minimiser le critère J est 3.

Pour déterminer les structures rigides dans une séquence, il est possible de tester toutes les combinaisons de trois segments dans l'image, malheureusement, cette méthode est peu robuste à cause des faux appariements, Pour cela, les auteurs ont proposés un moyen détourné pour tester la rigidité : prendre un ensemble de cinq

segments et considéré qu'il représente au maximum deux objets le tableau ci-après illustre les différentes configurations :

Objet 1	Objet 2
5	0
4	1
3	2
2	3
1	4
0	5

De cette manière, on a toujours un ensemble de 3 segments ce qui permet d'estimer les six paramètres du champ affine (prendre seulement 4 segments peut engendrer deux objets de 2 segments chacun ce qui rend l'estimation des six paramètres impossible).

Avant de tester un ensemble de segment, il faut faire un regroupement préalable des segments contenus dans l'image, ce regroupement n'est pas aléatoire, mais de la sorte que les groupes obtenus aient une forte probabilité d'être rigide. Pour faire, des critères de complémentarité sont imposés pour ne traiter que les structures qui ont une forte chance d'être rigide. Le regroupement se fait de manière itérative i.e qu'un segment S est inséré dans une structure S^p s'il vérifie certains critères :

Le premier est que la distance euclidienne entre le point de milieu du segment S à vérifier et le point milieu de n'importe quel segment de la structure S^p soit inférieur à un seuil K

$$\forall p, d_m\left(S, S^p\right) = \sqrt{\left(y_m^p - y_m\right)^2 + \left(x_m^p - x_m\right)^2} \prec k$$

Le deuxième est que les segments de S^p les plus proche de S aient des orientations différentes de celle de S (en effet si deux segments proche d'une structure possédant les mêmes orientations le déterminant de la matrice de résolution des six paramètres est nul).

Ce critère se traduit sous la forme mathématique comme suit :

Soit N et N^p les normales aux segments S et S^p ,L^p la longueur du segment S^p ,M et M^p les points milieu, ux+vy+w=0 est l'équation de la droite engendrée par S^p .

$$d_{D_{u,v,w}}(x_m,y_m)=\frac{|ux_m+vy_m+w|}{\sqrt{u^2+v^2}}<S_1, d_E(M,M_p)<\frac{L^P}{2}\Rightarrow\frac{\left|N_x.N_y^p-N_y.N_x^p\right|}{\left\|\overrightarrow{N}\right\|.\left\|\overrightarrow{N_P}\right\|}$$

Le troisième est qu'une au moins des deux extrémités du segment S soit assez proche d'une des extrémités d'un segment classé dans la structure

En utilisant ces critères, le nombre de combinaisons à tester est réduit considérablement. Le test de rigidité consiste a estimer les champs affines de toutes les combinaisons de trois segments parmi les cinq avec les deux segments restants, la structure est rigide si l'erreur est inférieure à u seuil fixé.

Finalement, une classification des structures est effectuée en comparant les champs affines entre eux, parmi les classes issues de cette étape celles ayant les mêmes paramètres de mouvement que ceux de la caméra connu à priori, cette classe est celle de la scène fixe dans la séquence.

I.4.1.3 Critiques.

L'inconvénient principal aux approches basées sur la segmentation est sûrement la lourdeur de la segmentation elle-même, en effet, il est quasiment impossible de concevoir un système de suivi temps réel en se basant sur l'extraction de primitives de types segments.

La puissance de calcul des processeurs modernes a permis d'envisager des applications utilisant la segmentation (codage et compression vidéo MPEG par exemple). En outre, les approches basées sur la segmentation ont comme grand avantage la production des indicateurs ayant un contenu sémantique ce qui est un avantage pour les traitement ultérieurs (particulièrement la reconnaissance).

I.4.2 Approches statistiques

Dans ce type d'approche, le problème du suivi du mouvement est ramené à un problème de discrimination statistiques entre l'objet en mouvement et le reste de l'image. On trouve plusieurs synonymes à ce type d'approches (Sélection d'attribut, modèles d'apparence). Elles sont utilisées souvent dans le cas ou il est difficile de qualifier l'objet en termes des primitives (pour des raisons de bruits ou la nature déformable de l'objet) La détection des objets en mouvement se fait généralement par classification basée sur des critères statistiques et particulièrement ceux de la couleurs des pixels.

I.4.2.1 Exemple: suivi par sélection d'attribut.

Plusieurs chercheurs ont utilisé l'information de la couleur (ou le niveau de gris) comme critère de classification des pixels appartenant a l'objet en mouvement et ceux appartenant a l'arrière plan (scène fixe), [CY03] ont utilisés les composant RGB comme base de sélection d'attributs adaptative (dynamique). Le problème du suivi du mouvement est ramené à un problème de discrimination locale entre deux classes Objet et arrière plan sous l'hypothèse que les attributs séparant le mieux entre objet/Scène sont les mieux adaptés pour le suivi les axes de leur méthode sont:

A- Les attributs candidats

On adopte un ensemble de épart dérivé directement des valeurs RGB du vecteur d'intensité des pixels. Cette représentation à l'avantage d'être robuste aux changements de point de vue (mouvement caméra), aux occultations et à la non rigidité. L'ensemble des attributs candidats est formé par toutes les combinaisons linéaires des valeur R,G et B du vecteur d'intensité du pixel :

$$F_1 = \{w_1 R + w_2 G + W_3 B / w_i \in [-2,-1,0,1,2]\}$$

Le nombre total brut de cet ensemble est 5^3 (125 combinaisons), en éliminant les combinaisons corrélées $k(w_i,w_j,w_k)$ (ex (-1,2,-1) et (1,-2,1)…etc) et le triplet nul (0,0,0) on obtient 49 candidats possibles dans F_1. Le choix est justifié par :

+ Calcul efficace (arithmétique simple sur des entiers)

+ Attribut mono valeur (1D) au lieu d'un espace 3D (RGB)

+ Tout type d'information représentative de l'apparence est inclus dans cet ensemble (vecteur RGB, chrominance approximative R-B, couleurs excessifs 2G-R-B…etc).

B- Sélection des attributs candidats.

Si le couple Objet/Arrière plan sont uni couleur, la variation apparente des valeurs RGB peut être ramenée à l'étude de la distribution gaussienne dans l'espace couleur, l'analyse linéaire discriminante peut être utilisée pour déterminer le sous-espace de projection qui séparera la moindre différence entre Objet et arrière plan. Cependant, on doit manipuler des objets et arrières plan qui possèdent des distributions multimodales (appartenants à plusieurs plages de valeurs) ce qui rend la résolution analytique inutilisable.

Les classes de distributions des attributs sont transformées chacune en une valeur calculée par le logarithme décimal du rapport de variance de l'attribut en question prise sur l'objet par celle prise sur l'arrière plan. Cette transformation non linéaire effectue deux tâches principales :

- Elle crée de nouveaux attributs adaptés à l'étude de la séparabilité Objet/Arrière plan

- Elle réduit les distributions multimodales en distributions unimodales et simplifie par conséquence la séparation des classes de distributions Objet/Arrière plan.

Pour cela, on utilise un rectangle de taille h*w couvrant l'objet et un autre cadrant qui représente l'arrière plan local à l'objet de taille 0.75*max(h,w). Etant donné un attribut F_i, soit $H_{obj}(i)$ et $H_{bg}(i)$ les histogrammes des valeurs F_i,pris respectivement sur l'objet et sur l'arrière plan. On définit deux probabilités discrètes empiriques p(i) et q(i) par normalisation de chaque histogramme en le divisant par le nombre d'éléments qui le composent, pour chaque attribut candidat potentiel on définit un nouvel attribut L plus adapté au processus de discrimination dont sa variance est donnée par

$$L(i) = Log\frac{Max(p(i),\varepsilon)}{Max(q(i),\varepsilon)}$$ où ε est une faible valeur utilisée pour éviter une division par 0 ou la prise du logarithme décimal d'un 0.

Cette transformation non linéaire projette les distributions prises sur l'objet en valeur positives et celles prises sur l'arrière plan en valeurs négatives constituant ainsi une image dont chaque pixel est muni d'un poids pour chaque attribut facilitant la discrimination.

Evaluation de la séparabilité des attributs :

Ici, une mesure du degré de séparabilité Objet/Arrière plan des attributs Li est calculé. Normalement on procède aux calculs on utilisant les valeurs Li, or, et pour des raisons d'efficacité on utilise les p(i) et q(i) déjà calculés on appliquant les formules déduites suivantes :

$$Var(L;p) = \sum_i p(i)L^2(i) - \left(\sum_i p(i)L(i)\right)^2$$

$$Var(L;q) = \sum_i q(i)L^2(i) - \left(\sum_i q(i)L(i)\right)^2$$

$$Var(L;(p,q)) = \frac{Var\left(L;{}^{(p+q)}\!/_{2}\right)}{Var(L;p)+Var(L;q)}$$

C- Classement des images de poids :

Les images de poids (Valeurs positives dans l'objet et négatives dans l'Arrière plan) nous donnent, dans le cas idéal, des indicateurs pour chaque pixel (1 pour pixel appartenants à l'objet et -1 pour pixels appartenants à l'arrière plan). Le suivi est réalisé par seuillage et le centre de l'objet est calculé par la méthode des moments. Dans la pratique, les distributions de l'objet et de son arrière plan sont s'entrechevauchent et la séparabilité totale Objet/Arrière plan n'est pas réalisée. Pour classifier les pixels on choisit les N meilleurs attributs réalisant la discrimination.

D- Suivi :

Le mécanisme de classification décrit précédemment est inclus dans le système de suivi. L'objet et son arrière plan étant bien définis dans la trame t-1, les N attributs mieux discriminants sont utilisés pour calculer N images de poids pour la trame t sous réserve que l'objet et l'arrière plan local gardent la même apparence de la trame précédente.

Un processus qui calcule le déplacement moyen est applique a chacune des N images de poids, ce processus mous fourni N déplacements moyen. On calcul le déplacement de l'objet par l'utilisation d'un simple estimateur médian :

$$X\wedge = \frac{\sum_{i=1}^{N} x_i}{N} \qquad Y\wedge = \frac{\sum_{i=1}^{N} y_i}{N}$$

Cet estimateur médian est utilise pour garantir une meilleure robustesse au lieu de prendre un seul déplacement L’algorithme est itère pour toutes la séquence de la vidéo, ainsi les attributs utilises pour le suivi et les classes d’objets et arrière plan évoluent conjointement a chaque instant. Cet algorithme rencontre comme tout application de vision artificielle le problème classique du bruit et des pixels mal-classes qui polluent particulièrement l’arrière plan et éventuellement provoquent l’échec du suivi. Pour remédier a ce problème on calcul la distribution de l’attribut empiriquement entre la trame T_{t-1} et T_0 on supposant que l’objet et son arrière plan sont non pollues dans la trame T_0. Cette heuristique suppose que les histogrammes des couleurs initiales restent représentatifs tout au long de la séquence.

I.4.2.2 Critiques:

Les méthodes statistiques ont l'avantage de fournir des réponses temps réel à la détection des zones de l'image en mouvement, par contre les zones détectées ne sont qu'une collection de pixels dépourvue de toute sémantique et contraint les traitements ultérieurs a refaire de la segmentation, outre qu'elles nécessitent une initialisation de départ. Si leur validité dans un domaine d'applications est justifiée, elles sont inadéquates pour d'autres champs applicatifs.

CHAPITRE II.

II. LES SYSTEMES MUTIAGENTS : CONCEPTS, ARCHITECTURES ET OUTILS

Les concepts d'IAD (Intelligence Artificielle Distribuée) et de SMA (Systèmes Multiagents) sont nés de la constatation suivante : l'intelligence peut émerger à partir de l'interaction entre plusieurs entités plus au moins intelligentes mais suffisamment autonomes. L'interaction peut engendrer un comportement plus riche que l'ensemble des comportements de ces entités prises séparément.

II.1.Introduction

Le thème des systèmes multiagents (SMA), s'il n'est pas récent, est actuellement un champ de recherche très actif. Cette discipline est à la connexion de plusieurs domaines en particulier de l'intelligence artificielle, des systèmes informatiques distribués et du génie logiciel. C'est une discipline qui s'intéresse aux comportements collectifs produits par les interactions de plusieurs entités autonomes et flexibles appelées agents, que ces interactions tournent autour de la coopération, de la concurrence ou de la coexistence entre ces agents. La description portera tout d'abord, les notions d'agents et de systèmes multiagents (SMAs), et détaille par la suite les différentes questions que soulèvent la problématique des SMAs, en particulier: les interactions et la coopération, la coordination, la planification et la communication. En deuxième lieu, on évoquera les architectures et outils actuels de conception des SMA.

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

Il ressort de cette définition, Quatre concepts l'autonomie, l'action, la perception et la communication que toute modélisation d'un SMA doit spécifier clairement.

II.2.Concepts SMA et Agent

II.2.1.Le Concept d'agent:

Il n'existe pas de Définition adoptée par la communauté scientifique sur le terme "agent". En revanche tout le monde s'accorde à dire que la notion d'autonomie est au centre de la problématique des agents. M. Ferber [Fer95] a proposé la définition suivante:

DÉFINITION : AGENT

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

Récemment, Jennings, Sycara et Wooldridge [JSW98] ont proposé la définition suivante pour un agent **(Figure 4.1)**:

<Un agent est un système informatique, situé dans un environnement, et qui agit d'une façon autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu>.

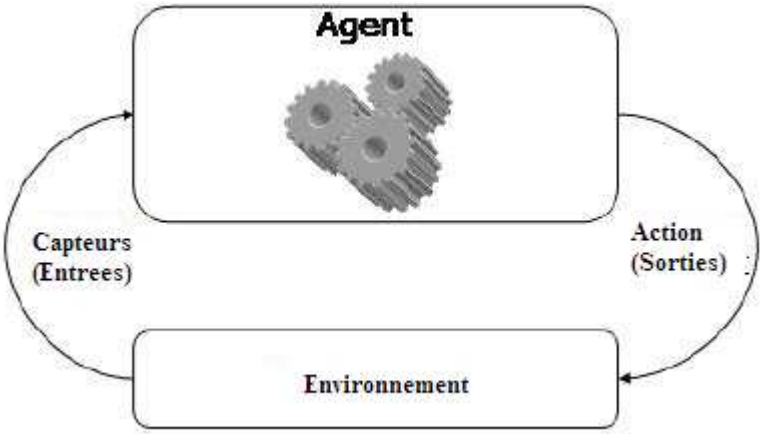


Fig. 2.1 Agent et sont environnement.

II.2.2.Système multiagents :

II.2.2.1. Définition :

Plusieurs auteurs ont fait une distinction claire entre une simple collection d'agents (Agents d'interface, Agents sur le Net) et un Système Multi-Agents où les Agents ont une capacité à interagir pour résoudre collectivement un problème.

DEFFINITION : SYSTEME MULTI-AGENTS

<On appelle système multi-agents (ou SMA), un système composé des éléments suivants:

1. un environnement E i.e., un espace disposant généralement d'une métrique.

2. un ensemble d'objets O. Ces objets sont situés i.e., pour tout objet il est possible, à un moment donné, d'associer une position dans E. Ces objets sont passifs i.e., qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.

3. un ensemble A d'agents, qui sont des objets particuliers (A O), lesquels représentent les entités actives du système.

4. un ensemble de relations R qui unissent des objets et des agents entre eux.

5. un ensemble d'opérations Op permettant aux agents de A de percevoir. produire, consommer, transformer et manipuler des objets de O.

6. des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette technique de modification, que l'on appellera les lois de l'univers [Fer95]>.

II.2.2.2. Les caractéristiques d'un SMA :

Les principales caractéristiques des SMA sont:

- Chaque agent ne dispose que d'informations incomplètes et a un champ d'action limité.
- Le contrôle du système est réparti.
- Les données manipulées sont décentralisées.
- Les traitements sont asynchrones.

II.2.2.3. Approche Voyelles (AEIO) de description d'un SMA.

Vu que la communauté SMA n'a pas encore trouvé de consensus pour définir ce qu'est un SMA. On constate que les différences entre les visions d'un SMA résident essentiellement dans l'approche de modélisation et de conception de ce dernier.

Nous adoptons une approche qui distingue quatre dimensions AEIO:
Agent, Environnement, Interaction, Organisation, il est possible d'utiliser cette approche en mettant l'accent sur n'importe laquelle des dimensions.

L'intérêt majeur de l'approche AEIO est sa vision générale des SMA. En effet, elle permet de considérer la conception d'un SMA selon la dimension qui semble la plus intéressante sans toutefois en privilégier une.

A- La dimension Environnement (E)

Représente l'univers d'appartenance des agents. Cet environnement contient l'ensemble des objets passifs pouvant être perçus ou actionnés par les agents.

B- La dimension Interaction (I)

Cette dimension est définit par FERBER comme <la mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques>.il existe trois types d'interactions avec communication: par signaux, par échanges de messages, et sophistiquées.

- Communications par signaux : Les signaux sont des informations qui circulent dans le SMA. Un exemple typique de ce type de communication est le comportement d'agents réactifs. En effet, les agents reçoivent des stimuli et répondent par des actions qui correspondent chacun à des signaux reçus et émis [Dro93 et al]. Ces signaux doivent transiter par l'intermédiaire de l'environnement. Ainsi, dans une communication par signaux, il n'y a pas d'interaction directe entre deux agents. Le rôle de chaque signal est défini au sein des agents lors de leur conception. Certains utilisent les signaux pour synchroniser les agents [Geo88].

- Communications par échanges de messages : Ces communications permettent aux agents d'échanger des informations, ou des plans d'actions. Citons par exemples les langages à acteurs qui utilisent ce type d'interaction [Hew77]. à la vue de l'évolution des techniques et des travaux actuels, les communications par échanges de messages tendent vers les communications sophistiquées [Pro02].

- Communications par actes de langages : Les communications par actes de langages sont initialement issues de travaux sur les langues naturelles, sur les dialogues entre l'homme et la machine, et sur les intentions dans les communications. L'évolution sans

cesse croissante des capacités de communication a permis d'introduire ce type de communication au sein des interactions entre entités informatiques. Cela s'est d'autant plus accéléré que l'utilisation des réseaux informatiques s'est faite de plus en plus importante. En effet les programmes fonctionnant au sein de réseaux ont besoin d'échanger des informations et, par évolution naturelle, de mettre en œuvre des processus de communication de plus en plus complexes. Les communications sophistiquées sont basées sur des langages et des protocoles.

Langage: Le langage est généralement basé sur la théorie des actes de langages développée par SEARLE [Dev06 et al]. Malheureusement, les actes de langages se limitent à une action et ne traitent pas une séquence d'interactions [Bra93]. Le nombre de langages est relativement important et peu homogène: KQML, ARCOL, ASIC, COOL ou FIPA qui tente cependant a uniformiser la notion de langage en proposant l'<Agent Communication Language ACL> [FIPA98].

C'est en 1975 que John AUSTIN se penche sur l'énonciation et sa définition comme un acte qui sert avant tout à produire des effets sur son destinataire [Aus75]. Ainsi, un acte de langage désigne l'ensemble des actions intentionnelles effectuées au cours d'une communication. Par conséquent, cette dernière est une action dans laquelle nous trouvons un locuteur qui émet un message et un allocuteur qui le reçoit. Un acte de langage est défini par John AUSTIN et ses successeurs.

C- La dimension Organisation (O):

La dimension O (pour Organisation) regroupe l'ensemble des règles et des contraintes permettant de structurer le système multi-agents.

DÉFINITION : LA DIMENSION O

<l'organisation dans sa spécification la plus simple peut être définie comme une restriction de l'autonomie (capacité des agents à prendre seul des décisions) des agents de manière à assurer à des degrés divers leur coopération pour un but donné. Un modèle organisationnel est une spécification formalisée de ces contraintes > [Fer95].

L'organisation représente l'ensemble des liens entre les agents. Ainsi que les contraintes sociales et physiques.

D- La dimension Agent :

DÉFINITION : LA DIMENSION A

<décrie l'ensemble des fonctionnalités de l'agent. Ainsi un agent est capable de raisonner, c'est-à-dire de planifier ses actions, et d'introspecter afin de raisonner sur ses propres compétences et connaissances [Les83]>.

D'autre part, un agent est capable d'interagir avec l'environnement ou avec d'autres agents. Pour cela, il met en œuvre les protocoles de communication définis dans la dimension Interaction. Un agent peut aussi intégrer les aspects organisationnels (dimension O) dans son processus de raisonnement [Sic94 et al]. Ainsi, un agent possède souvent une représentation des autres et de son environnement afin de pouvoir réaliser ses calculs de planification pour atteindre ses objectifs.

- Agent réactif VS Agen cognitif :

Un des grands noms parmi les critiques du raisonnement symbolique fut Brooks qui, par le biais de plusieurs articles, exposa son opposition au modèle symbolique et proposa une approche alternative qu'on appelle aujourd'hui IA réactive (SMA réactifs). Selon lui, le comportement intelligent devrait émerger de l'interaction entre divers comportements plus simples. Ainsi, au sein de son programme de recherche, il a développé l'architecture subsumption. Dans cette architecture, on bâtit des agents sans utiliser de représentation symbolique ni de raisonnement. Un agent est alors vu comme un ensemble de comportements accomplissant une tâche donnée. Chaque comportement est une machine à états finis qui établit une relation entre une entrée sensorielle et une action en sortie.

Typiquement, l'ensemble des comportements est représenté sous forme d'une hiérarchie dans laquelle les couches des niveaux inférieurs représentent des comportements moins abstraits et les couches des niveaux supérieurs, des comportements plus abstraits. Le développement d'un agent devient donc un processus où l'on devra expérimenter avec les nouveaux comportements. Ceci est habituellement accompli en plaçant l'agent dans son environnement et en observant les résultats.

Des reproches ont été adressés à cette approche dite "réactive", parmi lesquels, il convient de voir que:

– si les agents ne possèdent pas de modèle de leur environnement, ils doivent posséder suffisamment d'informations locales leur permettant de choisir une action acceptable.

– comme les agents basent leurs décisions sur des informations locales, il est difficile de voir comment ils pourraient tenir compte des informations non-locales.
– il est difficile de voir comment un agent purement réactif peut apprendre de son expérience et améliorer ainsi ses performances.
– le comportement global d'un agent devrait émerger des interactions entre les divers comportements qui le composent, cette émergence rend donc très difficile la tâche de construire un agent dans le but d'effectuer une tâche spécifique.
– s'il est assez simple de bâtir un agent qui comporte très peu de couches, l'exercice devient beaucoup plus compliqué lorsqu'on a besoin de plusieurs couches. Les interactions dynamiques entre les diverses couches deviennent trop complexes à comprendre.

- Architectures hybrides :

Dès le début des années 90, on savait que les systèmes réactifs pouvaient bien convenir pour certains types de problèmes et moins bien pour d'autres. De même, pour la plupart des problèmes, les solutions de l'IA classique, basées uniquement sur la planification, ne conviennent pas non plus. On commence dès lors à investiguer la possibilité de combiner les deux approches afin d'obtenir une architecture hybride [Fer92] et [Fis94]. Dans ce cas, un agent est composé de plusieurs couches, arrangées selon une hiérarchie, la plupart des architectures considérant que trois couches suffisent amplement. Ainsi, au plus bas niveau de l'architecture, on retrouve habituellement une couche purement réactive, qui prend ses décisions en se basant sur des données brutes en provenance des senseurs. La couche intermédiaire fait abstraction des donnés brutes et travaille plutôt avec une vision qui se situe au niveau des connaissances de l'environnement. Finalement, la couche supérieure se charge des aspects sociaux de l'environnement, c'est à dire du raisonnement tenant compte des autres agents.

II.3.Architectures des SMA

Il est possible de classer les architectures d'agents, dans [Wei99] on trouve :

- Les architectures basées sur la logique dans lesquels la prise de décision est le fruit d'une déduction logique.
- Les architectures réactives pour lesquelles la prise de décision est implémentée comme un lien

direct entre situation et action.

- Les architectures BDI pour lesquelles la prise de décision est le fait de manipulation de structures de données représentant les croyances, les désirs et les intentions de l'agent.
- Les architectures en couche pour lesquelles la prise de décision se fait au niveau de plusieurs couches logicielles raisonnant sur l'environnement à des niveaux d'abstraction différents.

II.3.1.Architectures Délibératives (cognitives).

L'approche classique pour construire des agents consiste à les voir comme un type particulier de systèmes à base de connaissances, c'est l'approche de l'IA symbolique.

On considère qu'un agent (ou une architecture d'agent) est délibératif s'il contient une représentation symbolique explicite du monde et que ses décisions (par exemple choix de l'action à accomplir) sont le fait d'un raisonnement logique basé sur la manipulation de ces symboles. Deux problèmes fondamentaux se posent :

– La traduction du monde réel en une description symbolique suffisamment précise et pertinente tout en restant utilisable en pratique.

– Pouvoir de raisonnement avec ces informations dans un temps suffisamment court pour que les résultats soient encore utiles.

Hormis les travaux qui ont été menés en représentation des connaissances, en raisonnement automatique, en apprentissage ou encore en planification automatique nous sommes loin d'avoir résolu ces problèmes. Malgré que ce type d'algorithmes semble crucial si l'on veut pouvoir faire évoluer les agents dans des domaines du monde réel en temps contraint, La construction de tels agents (démonstrateur de théorèmes) est, pour le moment du moins, inutilisable en pratique.

La planification des agents est un domaine actif depuis le début des années 1970 ayant contribué significativement aux avancées dans la conception d'agents artificiels. La planification peut être assimilée à la programmation automatique, il s'agit de trouver une suite d'actions qui, une fois exécutée, permettra d'atteindre le but désiré. Dans la plupart des cas, le module de planification automatique devrait être un constituant central d'un agent artificiel. Dans ce aui suit on présentera brièvement quelques architectures délibératives.

II.3.1.1. IRMA

IRMA (Intelligent Ressource-bounded Machine Architecture) [Col01] de Bratman est né du besoin d'un modèle capable de développer un raisonnement pratique (practical reasoning system - un système par lequel un agent forme des plans) et d'exhiber un comportement rationnel pour des agents aux ressources limitées. IRMA propose d'inclure différents mécanismes permettant de limiter les calculs nécessaires aux raisonnements.

Cette architecture est composée de quatre structures de données symboliques principales : une bibliothèque de plans, une représentation explicite des croyances, des désirs et des intentions. Les croyances contiennent les croyances de l'agent sur son environnement, les plans constituent la connaissance procédurale de l'agent, cette connaissance est stockée sous forme de plans partiels que l'agent peut adopter comme intentions. Les plans jouent à la fois un rôle fonctionnel dans le choix des actions mais permettent également de limiter les options qui s'offrent au système dans ses raisonnements. Les désirs représentent les états futurs idéaux de l'environnement, ils peuvent être incompatibles. Les intentions sont des plans instanciés qui permettent d'exprimer les actions que l'agent doit accomplir pour réaliser ses désirs.

II.3.1.2. AUTODRIVE

Né de la volonté de traiter les différentes tâches humaines impliquées dans la conduite d'un véhicule [Col01 et al], l'architecture AUTODRIVE permet aux agents de planifier leurs routes dans un environnement multiagents simulant des conditions de trafic.

L'architecture est centrée autour d'un planificateur qui combine des éléments traditionnels de résolution de problème (génération de plans hiérarchiques et ordonnés pour les routes) avec un processus de création dynamique de buts. Ce type de génération de plan est basé sur l'isolement des buts de haut niveau qui présentent une certaine stabilité (par exemple, le choix fixe du chemin à suivre) des sous-buts de bas niveau plus instables (par exemple les arrêts aux feux rouges).

Les agents AUTODRIVE sont capables de reconnaître les plans des autres agents et d'associer des informations dérivées de leurs observations à des informations abstraites par inférence. Cela leur permet de modifier dynamiquement leurs plans sur la base de prédictions de conflits ou d'interactions futures.

II.3.1.3. HOMER

Vere et Bickmore [Dev06 et al] estimèrent que les technologies relatives aux agents étaient suffisamment avancées pour pouvoir construire un prototype d'agent autonome avec des compétences linguistiques et capable de planifier et agir. Ils développèrent un tel agent et le nommèrent HOMER. HOMER est un robot sous-marin simulé qui évolue dans un monde en deux dimensions dont il a une connaissance partielle. Si HOMER évolue avec une connaissance partielle du monde, il peut obtenir de nouvelles informations par ses perceptions de l'environnement ou directement par les informations que lui fournit l'utilisateur. HOMER reçoit des instructions d'un utilisateur dans un sous-ensemble de l'anglais, ces instructions pouvant contenir des références temporelles modérément complexes. HOMER peut alors planifier ses actions, les exécuter, modifier ses plans afin d'accomplir les instructions qu'il a reçu L'agent enregistre tous les évènements dans une mémoire épisodique limitée lui permettant de répondre à des questions sur ses expériences passées, sur ses activités courantes, ses perceptions ainsi que sur ses intentions futures.

Les opérations d'HOMER sont centrées autour d'un planificateur temporel qu'il utilise pour la synthèse de plans en réponse aux sollicitations de l'utilisateur humain.

Les commandes intègrent généralement des contraintes de temps et le planificateur impose des conditions qui sont constamment surveillées durant les phases de planification et d'exécution. Un ensemble de modèles d'activité sous la forme de précondition/postcondition décrivant les actions, les inférences et les évènements que l'agent connaît est associé au planificateur. Le planificateur d'HOMER est capable de replanifier de manière limitée afin de gérer les buts supplémentaires qui lui sont donnés.

II.3.1.4. GRATE

GRATE (Generic Rules and Agents model Testbed Environment) est une architecture en couches dans laquelle le comportement de l'agent est guidé par ses croyances, ses désirs, ses intentions et ses intentions jointes [Col01 et al]. L'idée fondamentale dans la construction de l'architecture de GRATE est de fournir une séparation claire entre les capacités de résolution de problèmes d'un agent (qui sont dépendantes du domaine) et l'architecture de contrôle nécessaire à la résolution coopérative de problème dans un groupe d'agents. GRATE est donc conçu autour de

deux niveaux :

– le niveau coopération et contrôle qui contient les mécanismes génériques nécessaires pour la résolution coopérative de problèmes

– le niveau du domaine qui contient les capacités de résolution de l'agent propre à un domaine.

L'architecture comprend différents modules génériques et données qui forment la couche de coopération et de contrôle. Ces modules et ces données sont préprogrammés et sont adaptés par les développeurs pour les problèmes spécifiques d'un domaine. Les accès au niveau du domaine sont gérés par un module intermédiaire de contrôle. Le module de contrôle sert d'interface entre le niveau du domaine et le niveau de coopération et de contrôle, le module de gestion de situation est responsable de la gestion des activités locales (adoption de plans, transmission d'informations, sélection de tâches) et sociales (engagements dans des opérations coopératives) de l'agent et le module de coopération est responsable de la gestion de toutes les interactions entre l'agent et ses accointances.

II.3.2. Architectures non Délibératives (Réactives).

Comme nous l'avons mentionné, il y a de nombreux problèmes non résolus associés à l'IA symbolique. Ces problèmes ont conduit les chercheurs à remettre en question tout le paradigme symbolique ce qui a conduit au développement des architectures non délibératives. Ces architectures non délibératives comprenant généralement les architectures réactives, situés ou orientés comportement prennent généralement toutes les décisions de contrôle à l'exécution sur la base d'informations limitées (uniquement l'information disponible au niveau des capteurs), d'un état interne limité et avec un minimum d'inférences. On peut donc estimer que ces architectures n'incluent pas de représentation symbolique pour modéliser le monde et n'effectuent pas de raisonnement symbolique complexe.

Alors que les planificateurs traditionnels cherchent à produire des séquences d'actions optimales ou correctes, les architectures non délibératives sont conçues pour produire des actions robustes. Généralement construits à partir de mécanismes de contrôle très simples (machines à états finis, règles stimulus-réponse), de telles architectures ont été stimulées par l'hypothèse de Simon selon laquelle le comportement complexe d'un agent ne doit pas nécessairement être le produit d'une

organisation interne complexe mais pourrait être le reflet de la complexité de l'environnement dans lequel il évolue.

II.3.2.1. L'architecture subsumption

Brooks [Bdv05 et al] a présenté une architecture alternative pour le développement des agents : la subsumption architecture. Dans ses travaux ultérieurs Brooks a mis en avant plusieurs thèses :

1 – un comportement intelligent peut être obtenu sans représentations explicites telles que celles proposées par l'IA symbolique.

2 – un comportement intelligent peut être obtenu sans raisonnements abstraits explicites tels que ceux proposées par l'IA symbolique.

3 – l'intelligence est une propriété émergente d'un système complexe.

Deux idées ont guidées ses recherches :

- La vraie intelligence est située dans le monde et non dans des systèmes désincarnés tels que les systèmes experts.
- Un comportement intelligent apparaît comme le résultat d'une interaction de l'agent avec son environnement.

Pour démontrer ses propos, Brooks a construit de nombreux robots basés sur son architecture. L'architecture subsomption a une structure en couches dans laquelle, chaque couche réalise un comportement lié à l'accomplissement d'une tâche spécifique (par exemple l'évitement d'obstacles). Cette façon de procéder est souvent appelée décomposition verticale contrairement aux approches classiques dites horizontales.

Chaque couche de l'architecture subsomption est composé d'un réseau de machines à états finis (FSM) associé à des données ainsi qu'à des unités de gestion du temps. Les couches communiquent par le biais de messages de taille fixe. Les FSM d'une couche peuvent changer d'état en fonction des messages reçus ou du temps qui s'est écoulé. Les couches travaillent de manière asynchrone et en parallèle, elles n'utilisent pas de mémoire globale partagée.

Les couches de plus haut niveau subsument les rôles des couches de plus bas niveau lorsqu'elles veulent prendre le contrôle. En particulier, les couches peuvent, pour des périodes de temps programmées, supprimer les entrées et inhiber les sorties des couches de plus bas niveau ce qui aura pour effet de pousser l'agent vers la

réalisation de ses buts de plus haut niveau tout en respectant ses buts critiques de plus bas niveaux.

Il faut bien voir que ces systèmes sont extrêmement simples en terme de calcul et que pourtant Brooks a démontré que des robots équipés d'une telle architecture exhibaient des comportements évolués.

II.3.2.2. Pengi

Chapman & Agre [Dev06 et al] parvenaient à des conclusions similaires sur les difficultés relatives à l'IA symbolique. Ils sont partis de l'observation suivante: la plupart de nos activités quotidiennes ne sont que routines, dans le sens où elles ne nécessitent pas de raisonnements évolués de notre part. La plupart des tâches, une fois apprises, peuvent être accomplies de façon routinière sans grands changements. Agre proposa d'adapter cela dans une architecture d'agent. L'idée est que la plupart des décisions étant des routines, elles peuvent être encodées dans une structure de bas niveau (comme un circuit) qui serait mise à jour périodiquement pour traiter de nouveaux problèmes. Cela a donné naissance au système Pengi. Pengi est un agent autonome (un pengouin) qui évolue dans un monde simulé peuplé de prédateurs hostiles. L'activité de Pengi est basé sur la notion de routine, i.e. un motif d'interaction entre l'agent et le monde. Un agent s'engage dans une routine sans idée préconçu de ce qui va arriver, lorsque la situation change d'autres réponses deviennent disponibles, un tel agent improvise ses actions. L'architecture de l'agent est composée d'un système central et d'un système périphérique. Le système central est implémenté comme un ensemble de règles situation-action et se charge de la sélection de l'action appropriée en fonction des circonstances. Le système périphérique est responsable du traitement des perceptions et du contrôle des effecteurs.

II.3.2.3. Automates Situés

Rosenschein et Kaelbling [RK96] ont développé une architecture opérant en temps réel, basée sur la théorie des automates situés qui propose une sémantique formelle fournissant une spécification du contenu des états internes d'une machine en fonction des états externes de l'environnement dans laquelle la machine se trouve. Cette spécification peut être décrite ainsi : quand une machine est dans l'état s on peut dire qu'elle porte l'information que "si et seulement si chaque fois que la machine est dans l'état s, la proposition est vraie" dans l'environnement.

Cette théorie a conduit à une méthode de programmation pour les agents embarqués dans laquelle on considère que l'agent effectue une transduction du flux des perceptions reçus de l'environnement vers un flux d'actions qu'il exerce sur l'environnement. Le calcul est modélisé au sein de l'agent sous forme de machine à états finis, exprimé comme un circuit dont l'on peut garantir qu'il répondra dans une borne de temps constante, le circuit ne fait aucun calcul symbolique, toutes les manipulations symboliques sont effectuées à la compilation. Le but est de réduire toute l'information déclarative sous forme de circuit très simple.

Cette approche est conçue pour permettre la compilation automatique de descriptions (tâches, environnement) de haut niveau vers des mécanismes de contrôle de bas niveau. Pour accomplir cette tâche, les auteurs ont développé Gapps un langage qui facilite la programmation d'un agent en générant automatiquement les réseaux logiques appropriés et Rex, un langage de description pour générer des circuits.

II.3.2.4. Reactive Action Packages

Firby a [Dev06 et al] développé un planificateur réactif basé sur la notion de Reactive Action Packages (RAPs). Les RAPS sont des processus autonomes qui poursuivent un but jusqu'à ce qu'il soit satisfait. Si le planificateur a plusieurs buts, il aura plusieurs RAPs indépendants tentant chacun de satisfaire son but.

Le planificateur réactif est composé d'une file d'exécution de RAP, d'un interpréteur de RAP, d'un modèle du monde mis à jour constamment, et enfin d'une interface matérielle aux senseurs et aux effecteurs du robot. Les RAPs sont constitués d'un ensemble prédéfini de méthodes pour atteindre un but particulier. Ces méthodes, annotées par des contraintes d'applicabilité, sont soit des commandes primitives (actions envoyées à l'interface matérielle) ou un réseau partiellement ordonné de sous-tâches.

Les RAPs sont placés dans la file d'exécution en attendant d'être sélectionnés par l'interpréteur. Ils sont choisis sur la base de contraintes temporelles et d'ordre résidant dans le réseau de tâches des RAPs. Une fois choisi, un RAP consulte le modèle du monde et en fonction des informations sur la situation courante enverra une commande à l'interface matérielle ou placera des sous-buts pour les prochains calculs.

Les RAPs retournent dans la file d'exécution s'ils sont dans l'attente de l'exécution d'un sous-but, à ce moment là un autre RAP peut être choisi. L'exécution des RAPs est intercalée. Ce planificateur garantit une très bonne réactivité.

II.3.2.5. Plans Universels

Les plans universels de Schoppers [Dev06 et al] sont une représentation des comportements d'un agent qui spécifie les actions appropriées (réactions) pour toutes les situations qui peuvent être perçues dans un domaine particulier. Un plan universel est effectivement un plan linéaire conditionnel ou un arbre de décision, qui étant donné un état initial de l'environnement, peut associer à chaque état possible du monde une action spécifique. Les actions dans un plan universel sont sélectionnées par une classification de la situation courante à l'exécution. Pour ce faire, le plan universel doit explicitement identifier tous les états du monde pouvant être perçus durant l'exécution de l'agent.

Les plans universels sont compilés à l'avance. L'arbre de décision qui en résulte représente le but de l'agent (racine de l'arbre) et ses sous buts. Durant l'exécution, le plan compilé est interprété afin de trouver à chaque instant l'action la plus appropriée en fonction de l'état courant du monde.

II.3.2.6. Architecture en réseau

Maes [Dev06 et al] a proposé un algorithme de sélection de l'action qui représente un agent autonome comme une collection de modules de compétences. Ces modules sont proches des opérateurs des systèmes classiques de planification : ils spécifient leurs conditions d'activation ainsi que leurs effets attendus. Chaque module se voit associer un niveau d'activation.

Les modules de compétence forment un réseau utilisant trois types de liens (successeur, prédécesseur, conflit) qui indique les relations entre les modules. Les modules utilisent ces liens pour activer ou inhiber d'autres modules accumulant de l'énergie qui représente les meilleures actions à effectuer. Les modules sont sélectionnés lorsque leur niveau d'activation atteint un seuil prédéfini.

Les énergies d'activation proviennent de la situation courante observée et des buts de l'agent. Le comportement global de l'algorithme de sélection de l'action est paramétré par différents seuils : seuil d'activation d'un module, niveau d'énergie

injecté dans le réseau par les observations, niveau injecté par les buts, niveau d'énergie consommé par les buts à maintenir. La dynamique d'interaction entre les modules établit la séquence d'actions sélectionnées de manière distribuée.

Cette architecture se rapproche de l'architecture subsomption ainsi que des réseaux de neurones.

II.3.2.7. HPTS

Le modèle HPTS (Hierarchical Parallel Transition Systems) [Don01], défini par S. Donikian et É. Rutten, est basé sur une hiérarchie de modules constitués d'automates parallèles. Il a été implémenté dans le cadre de la plate-forme GASP (General Animation and Simulation Platform). Une des particularités intéressante de ce modèle est qu'un automate permet de décrire aussi bien les comportements d'un agent, que ses capteurs ou ses actions d'animation internes et externes. L'interface d'un automate est constituée d'un ensemble d'entrées, de sorties et de paramètres de contrôle, ainsi que d'une boîte aux lettres. Les entrées et les sorties transmettent des flots continus de données. Les paramètres de contrôle influencent le comportement de l'automate ; ils peuvent être modifiés par l'automate lui-même ou par une entité extérieure. La boîte aux lettres permet de recevoir des messages, provenant en particulier des sous automates et de l'automate hiérarchiquement supérieur. Des messages prédéfinis permettent de contrôler l'activité d'un automate (lancement, suspension, reprise ou terminaison) et d'indiquer le statut courant de l'automate (actif, inactif ou suspendu).

Un automate encapsule par ailleurs des variables locales, une fonction d'intégration et un ensemble de sous-automates. Si cet ensemble est vide, l'automate est un état atomique. Plusieurs sous-automates peuvent être actifs en même temps. Le rôle de la fonction d'intégration d'un automate dépend de sa constitution :

– si l'automate est un état atomique, la fonction d'intégration dépend uniquement des flots d'entrées, des variables locales et des paramètres de contrôle.

– sinon, la fonction d'intégration réalise l'arbitrage entre les propositions d'action des sous-automates actifs.

La fonction d'intégration peut être :

– une fonction classique, comme les opérations de calcul ou de comparaison sur les types de base (entiers, réels et booléens)

– une fonction de filtrage, comme la saturation ou le seuillage.

– une fonction de préemption, qui gère les actions conflictuelles selon des priorités strictes.
– un opérateur de retard, permettant de temporiser la circulation des données dans le cas de dépendances cycliques.
– une composition des fonctions précédentes.

Un langage spécifique de description d'automates HPTS a donc été introduit, ainsi qu'une interface graphique. Plusieurs exemples conséquents valident l'utilisation du modèle HPTS dans le cadre de la simulation de conducteurs autonomes.

II.3.3.Architectures Hybrides.

Suite aux expériences plus ou moins infructueuses que nous avons décrites, il fut suggéré que ni une approche purement délibérative ni une approche purement réactive n'étaient adaptées aux agents. Il fut donc proposé d'étudier des approches hybrides.

Une voie possible est de construire un agent comme la composition de deux (ou plus) sous-systèmes : un composant délibératif comprenant un modèle symbolique du monde, construisant des plans, prenant des décisions sur la base des travaux de l'IA symbolique et un composant réactif capable de réagir aux évènements de l'environnement sans effectuer de raisonnements complexes. Le composant réactif présentant souvent une forme de priorité sur le composant délibératif afin qu'il puisse fournir une réponse rapide aux changements environnementaux. Cela conduit naturellement aux architectures en couches pour lesquels se pose le problème de la mise en place d'une infrastructure de contrôle permettant de gérer les interactions entre ces différents niveaux.

II.3.3.1. PRS

Le Procedural Reasoning System (PRS) [IGR92] est une architecture générique pour représenter et raisonner sur les actions dans un domaine dynamique. L'architecture d'un module PRS contient: 1. une base de données contenant les croyances courantes du système sur le monde, 2. un ensemble de buts courants, 3. une bibliothèque de plans ou procédures, nommées Knowledge Areas (KAs), qui décrivent des séquences particulières d'actions et de tests à accomplir pour atteindre des buts donnés ou pour réagir à certaines situations et 4. une structure pour les intentions,

composée d'un ensemble partiellement ordonné de plans choisis à l'exécution.

Un interpréteur manipule ces composants, sélectionnant les plans appropriés (KAs) sur la base des croyances et des buts, ajoutant ces KAs dans la structure pour les intentions et enfin exécutant ces KAs. PRS interagit avec l'environnement par le biais de la base de données qui reflète les changements de l'environnement ainsi que par les actions qui sont exécutées. Plusieurs instances de PRS peuvent être utilisées de manière synchrone pour des applications nécessitant la coopération de plusieurs systèmes.

Le contenu de la base de données représente les croyances courantes du système, certaines croyances sont fournies initialement par l'utilisateur (généralement des faits sur des propriétés statiques du domaine), d'autres croyances sont dérivées par PRS lors de l'exécution des KAs (observations sur le monde ou conclusions dérivées par le système sur la base de ces observations). Dans PRS, les buts sont des descriptions des tâches ou comportements désirés.

Chaque KA est constitué d'un corps décrivant les étapes de la procédure, d'une condition d'invocation qui spécifie dans quelles situations le KA est utile. Il peut être représenté comme un réseau et peut être assimilé à un plan. Chaque arc du réseau est étiqueté par un but à accomplir. La condition d'invocation se décline en deux parties. La partie déclenchement est une expression logique décrivant les évènements qui doivent survenir pour que le KA soit invocable.

L'ensemble des KAs ne contient pas seulement de la connaissance procédurale sur un domaine spécifique mais peut également inclure des KAs de niveau méta, i.e. de l'information sur les manipulations des croyances, des buts et des intentions du système lui-même. Par exemple, un KA de niveau méta peut implémenter différentes méthodes pour le choix d'un KA parmi plusieurs applicables. Ces KAs de niveau méta ont accès aux données du système ainsi qu'aux propriétés des KAs.

L'interpréteur PRS se charge de faire tourner tout le système. La boucle est relativement simple : à tout moment certains buts sont posés et des évènements modifient les croyances de la base du système. Ces modifications vont alors déclenchées différents KAs. Un ou plusieurs KAs applicables vont alors être choisis et placés dans la structure d'intention. Enfin, PRS choisit une tâche de la racine de la structure de gestion des intentions et exécute une étape de cette tâche. Cela déclenche une action primitive, la formation d'un nouveau sous-but ou de nouvelles croyances. Les nouveaux buts et croyances déclenchent de nouveaux KAs et le cycle recommence.

Il est important de comprendre que chaque intention de la structure d'intentions

représente une pile de KAs invoqués. L'exécution d'un KA entraîne la formation de sous-buts qui eux même invoquent d'autres KAs ce qui forme une pile de KAs à la manière d'une pile d'appels de procédures des langages de programmation traditionnels. Lorsque le système traite plusieurs tâches, il gère ces piles dynamiques, exécutant, suspendant et relançant ces procédures à la manière d'un système d'exploitation.

PRS a été implémenté et appliquer avec succès à différentes tâches ayant des exigences temps réel dont notamment : surveillance de dysfonctionnements pour des modules d'engins spéciaux de la NASA, diagnostic, contrôle et surveillance de réseau de télécommunication, contrôle de robots mobiles, gestion du trafic aérien..

II.3.3.2. Adaptative Intelligent Systems

Un Adaptative Intelligent Systems (AIS) est un système à base de connaissances qui raisonne sur et interagit avec d'autres entités dynamiques en temps réel. Pour qu'un AIS soit fonctionnel il doit être capable de perception, d'action, de raisonnement et d'attention.

Hayes-Roth [Dev06 et al] a développé un AIS nommé GUARDIAN pour une application de surveillance de patients en soins intensifs. GUARDIAN comprend un composant cognitif, un ensemble de systèmes asynchrones d'entrée/sortie, un ensemble de canaux dynamiques d'entrée/sortie, et un cycle de raisonnement. Implémenté comme un black- board parallèle, le composant cognitif effectue des raisonnements généraux, s'engageant dans des processus de construction et de modification de plans, succession de décisions temporellement ordonnées décrivant les classes d'opérations que l'agent entend accomplir durant un certain temps. Ces opérations sont ensuite traitées par un gestionnaire d'agenda, par un Ordonnanceur et un Exécuteur d'opérations.

Les canaux dynamiques d'entrée/sortie intègrent les capteurs et effecteurs de l'agent avec le composant cognitif en implémentant des fonctions et heuristiques de sélection d'attention précalculées. Ceci permet de limiter le nombre d'opérations que doit traiter le composant cognitif.

II.3.3.3. Phoenix

Le projet Phoenix étudie les relations fonctionnelles entre les agents, les environnements dans lesquels ils évoluent et les comportements qui en résultent. Il a

une architecture temps réel utilisée pour contrôler différents agents autonomes et semi-autonomes immergés dans un environnement simulé de feu de forêt.

Les agents Phoenix sont composés de deux mécanismes parallèles presque indépendants pour la génération des actions. Le premier composant réflexe, est conçu pour générer des réactions immédiates face à différentes situations environnementales. Le second, le composant cognitif, est chargé d'agir à long terme, en effectuant des calculs plus coûteux de planification. Chaque composant est connecté indépendamment aux senseurs et aux effecteurs de l'agent.

Le composant cognitif a le contrôle final sur les actions de l'agent. Son rôle principal est d'instancier et d'exécuter des plans enregistrés. Il a également la charge de répondre aux signaux d'interruption du composant réflexe, afin de gérer les communications avec les autres agents et pour effectuer des fonctions avancées telles que la sélection de plan, la gestion et le contrôle des plans, la gestion des erreurs et la replanification. Le contrôle s'effectue par le biais de processus intégrés dans les plans qui peuvent être utilisés pour obtenir des informations sur les progrès des plans et pour générer des plans de gestion des erreurs si les résultats ne sont pas conformes aux attentes.

II.3.3.4. Touring Machines

L'architecture TouringMachines a été conçue par Ferguson [Fer92]. Elle comprend deux composants respectivement de perception et d'action qui s'interfacent directement avec l'environnement de l'agent et trois couches de contrôle intégrées dans un système de contrôle qui s'interface entre les composants couches qui sont des processus indépendants s'exécutant concurremment.

La couche réactive génère des actions en réponse aux évènements qui nécessitent une réponse trop rapide pour que les autres couches puissent intervenir. Elle est implémentée comme un ensemble de règles situation-action à la manière de l'architecture subsomption de Brooks.

La couche de planification construit des plans et choisit les actions à exécuter afin d'atteindre les buts de l'agent. Cette couche est constituée de deux composants : un planificateur et un mécanisme permettant de focaliser l'attention de l'agent.

Le planificateur gère la génération des plans et leur exécution, il utilise une bibliothèque de plans partiellement définis, ainsi qu'une carte topologique du monde,

afin de construire de plans pour accomplir les buts principaux de l'agent. L'intérêt du mécanisme de focalisation de l'attention est de limiter la quantité d'information que le planificateur doit gérer, et ainsi améliorer l'efficacité générale du système. Il fonctionne en filtrant les informations provenant de l'environnement.

Le modelling layer contient des représentations symboliques de l'état cognitif des entités évoluant dans l'environnement de l'agent. Les manipulations de ces modèles permettent d'identifier et de résoudre les conflits entre les buts, situations pour lesquels un agent ne peut plus atteindre ses buts du fait d'interférence entre eux.

Les trois couches peuvent communiquer par messages, elles sont regroupées dans un système de contrôle qui sert de médiateur entre les couches et, en particulier, s'occupe de gérer les conflits entre les actions proposées par les différentes couches en utilisant des règles de contrôle.

II.4.Les Outils de développement de SMA.

Les recherches en système multiagents ont conduit au développement de langages de programmation adaptés pour l'implantation de tels systèmes. Se doter de langages de programmation évolués permettant de tirer le meilleur parti de ce nouveau paradigme constitue l'un des enjeux fondamental du domaine dans la perspective d'une utilisation plus large de cette technologie. La gamme des langages proposés varie du purement déclaratif au purement impératif en passant par des approches hybrides et enfin les langages orientés coordination.

II.4.1.Langages Déclaratifs

Les langages déclaratifs, comme en témoigne leur nombre, représentent l'approche dominante pour la programmation des agents. Ils sont généralement caractérisés par leur nature formelle, généralement basée sur la logique. Parmi les langages déclaratifs on peut citer:

- Agent-0 et Agent-K: Basé sur le concept de programmation orientée agent - Agent-Oriented Programming (AOP) introduit d'une manière formelle par Shoham [Sho93] comme une spécialisation de la programmation orientée objet. Il pose ainsi les bases d'un nouveau paradigme de programmation permettant de décrire des agents intelligents et autonomes ayant des état mentaux tels que des croyances, des capacités, des engagements ainsi qu'une notion du temps.

Un système complet d'AOP devrait se composer :

– d'un langage formel avec une syntaxe et une sémantique claires permettant de décrire les états mentaux.

– d'un langage de programmation des agents

– d'un « agentifieur » permettant de convertir une entité neutre en agent programmable

Les actions d'un agent sont déterminées par ses choix. Ces choix sont contraints par les décisions antérieures de l'agent ainsi que par ses croyances. Shoham introduit donc deux catégories d'état mentaux : croyance et décision.

Les actions et les faits ne sont pas distingués, une action étant représentée par le fait correspondant. De ce fait, les actions sont considérées comme instantanées. Un langage temporel a été défini, par exemple having(robot; book)t signifie que le robot a un livre à l'instant t . Les croyances (beliefs) représentent les propositions que l'agent croit vrai à un moment donné. Elles peuvent porter sur le monde, sur l'état mental de l'agent ou encore sur les autres agents. Les obligations représentent les actions que l'agent s'est engagé à faire. Les décisions sont définies comme étant des obligations faites à soi-même. Les capacités représentent les actions que l'agent peut exécuter.

L'interpréteur générique d'agent exécute la boucle suivante :

1. lire les messages courants et mettre à jour l'état mental de l'agent
2. exécuter les engagements pris pour la date courante ce qui peut potentiellement induire des modifications de la base de croyances.

Dans Le langage de programmation associe Agent-0 les agents ne s'engagent à effectuer que des actions élémentaires qui ne nécessitent pas de planification. Les faits sont spécifient le contenu des actions et leurs conditions temporelles associées.

Agent-0 offre quatre types de routines de communication : (INFORM t a fact) avec t un temps, a un nom d'agent et fact un fait, (REQUEST t a action), (UNREQUEST t a action), (REFRAIN action).

Les actions conditionnelles permettent de spécifier des actions à effectuer sous certaines conditions portant sur l'état mental de l'agent. Une condition peut porter sur les croyances ou sur les obligations de l'agent. Les règles d'engagement Un agent s'oblige à effectuer une action en fonction des messages qu'il reçoit et de son état mental.

Dès lors, un programme Agent-0 contient la définition des capacités de l'agent, ses croyances initiales, la définition d'un time grain, et une séquence de règles d'engagement. A partir de ses croyances initiales, de ses règles d'engagement, et des

messages reçus, l'interpréteur de l'agent se charge de mettre à jour ses croyances et ses obligations ainsi que d'exécuter les actions qui doivent l'être à chaque pas de temps.

Le langage Agent-K resulte de l'integration de KQML (Knowledge Query & Manipulation Language) à Agent-0 [DE94].

+ ABLE: ABLE (Agent Behaviour Language) a été développé par un laboratoire de recherche de Philips [Dev06 et al]. Avec ABLE, les agents sont programmés sous forme de règles. Ces règles peuvent inclure une représentation du temps. ABLE peut être compilé en structures de bas niveaux à la manière de ce qui a été fait dans le travail sur les automates situés. Du fait de cette compilation l'exécution est très efficace. ABLE a été étendu en Real-Time ABLE pour supporter le temps réel.

+ MOZART: Mozart étend Oz pour le support des calculs distribués. Il permet le partitionnement d'un programme sur un ensemble de sites, un site étant un processus système. Les sites peuvent être lancés sur une machine locale ou sur un réseau de machines.

Mozart implémente un modèle de ressources. Dans ce modèle une ressource est vue comme une capacité restreinte à un seul site. Un programme Mozart est capable de spécifier dynamiquement les ressources dont il a besoin et de les utiliser. Mozart est un langage de programmation concurrente de haut niveau. Il implémente le réseau de manière transparente pour toutes les structures sous-jacentes de Oz. Cela offre un environnement de programmation adapté pour le développement d'agents logiciels. Il n'offre toutefois pas un support explicite des agents et de leurs états mentaux.

+ Concurrent METATEM: Concurrent METATEM [Fis93, Fis94] est un langage basé sur la notion d'objets concurrents communiquant dont les fondements logiques et le modèle de calcul diffère de l'AOP proposé par Shoham. Chaque objet exécute directement sa spécification donnée en logique temporelle et communique avec les autres objets de manière asynchrone par diffusion de messages.

+ TELESCRIPT: TELESCRIPT est le premier langage d'agent commercial, il a été développé par General Magic. La technologie TELESCRIPT propose deux concepts essentiels : places et agents. Les agents sont mobiles et peuvent communiquer entre eux. Quatre grands composants ont été développés pour supporter la technologie TELESCRIPT :

1. le langage TELESCRIPT conçu pour supporter les tâches de communication : mobilité, authentification, contrôle d'accès, ...

2. le moteur TELESCRIPT est l'interpréteur du langage TELESCRIPT qui maintient les places, contrôle l'exécution des agents et offre une interface avec les autres applications.
3. l'ensemble des protocoles TELESCRIPT, ces protocoles régissent principalement l'encodage et le décodage des agents pour permettre la mobilité des agents entre places.
4. un ensemble d'outils pour permettre le développement d'applications TELESCRIPT.

+ Le projet IMAGINE - APRIL et MAIL: APRIL (Agent PRocess Interaction Language) [MC95] et MAIL sont deux langages qui ont été développés pour faciliter le déploiement de systèmes multiagents dans le cadre du projet IMAGINE. Chacun a un rôle spécifique. APRIL a été conçu pour fournir les primitives centrales nécessaires : notamment capacités multi-tâches utilisant les processus et communication. APRIL++ est une extension orientée objet d'APRIL. Il est possible de définir des agents exhibant un comportement réactif avec APRIL par contre le système n'offre pas de buts explicites ni de mécanisme de raisonnements. APRIL est très générique mais implémenter un agent nécessite de tout reconstruire en utilisant simplement les primitives bas niveau qui sont offertes.

Au contraire, le langage MAIL propose une collection d'abstractions prédéfinies telles que les plans. APRIL a été envisagé comme le langage d'implémentation de MAIL.

+ PLACA: PLACA (PLAnning Communication Agents) [Tho95] est une extension d'Agent-0 qui vise à pallier ces limitations. Contrairement à Agent-0, PLACA se base sur la planification en supposant que tous les agents sont dotés de capacités élémentaires de planification, les agents PLACA peuvent ainsi échanger des requêtes de haut niveau sans se préoccuper de la manière dont elles vont être exécutées en pratique. Pour ce faire, PLACA ajoute deux nouveaux mécanismes aux états de l'agent.

Une liste consistante d'intentions et une liste de plans. Les intentions sont adoptées de façon très similaire aux obligations d'Agent-0. Les plans sont alors générés par un générateur de plans externes pour satisfaire les intentions de l'agent. Ce planificateur externe, qui peut être vu comme une boîte noire, a accès aux états mentaux de l'agent. Plus précisément, les nouvelles structures ajoutées dans PLACA sont les suivantes : (INTEND x) faire en sorte que x soit vrai, (ADOPT x) adopter l'intention, le plan x. (DROP x) abandonner l'intention, le plan x, (CAN-DO x), (CAN-ACHIEVE x), (PLAN-DO x), (PLAN-ACHIEVE x), (PLAN-NOT-DO x) qui peuvent être utilisés dans les conditions mentales

+ dMARS: dMARS (distributed Multi-Agent Reasoning System) [dKLW97] est une implantation en C++ de PRS. [dKLW97] propose une spécification formelle de l'architecture qui détaille précisément les différentes structures employées et les opérations qui les manipulent réalisant un pas en avant pour l'analyse théorique de PRS.

+ MyWorld: MyWorld est un langage de programmation orienté agent qui fut développé par Wooldridge et Vandekerckhove [Woo02 et al] sur la base des travaux de Shoham sur Agent-0 et sur l'architecture BDI. MyWorld illustre le fossé entre théorie et implémentation. Le système est constitué autour de quatre composants : 1. un Umpire responsable de l'exécution des agents et du contrôle de l'interface utilisateur, 2. un langage orienté agent dans lequel les agents sont programmés, 3. des contraintes portant sur l'environnement, 4. un scénario décrivant l'expérience en cours.

L'architecture d'un agent se compose de six composants : 1. un ensemble de croyances contenant les informations sur le monde. Les croyances peuvent être ajoutées de trois façons : à partir d'inférences issues des règles de croyance, à partir des perceptions sur le monde, à partir d'actions internes. 2. un ensemble d'intentions, une intention représente des états futures que l'agent veut atteindre. Les intentions sont formées d'un but, d'une motivation (une condition persistante), et d'une priorité. Le comportement d'un agent est dicté par les intentions de plus haute priorité. Les intentions sont maintenues jusqu'à ce que l'agent atteigne son but ou que l'intention ne soit plus valide car sa motivation n'est plus vraie. 3. un ensemble de règles de croyance qui définissent comment de nouvelles croyances peuvent être induites à partir des croyances courantes. Ces règles sont appliquées à chaque cycle de la boucle d'exécution des agents. 4. un ensemble de règles d'adoption d'intention qui définissent les situations pour lequel l'agent doit adopter de nouvelles intentions. 5. un ensemble de règles de stratégie qui définissent comment atteindre les intentions. 6. un ensemble d'actions que l'agent peut accomplir.

Le cycle d'exécution d'un agent comprend les cinq étapes suivantes : 1. ajouter les nouvelles croyances et les mettre à jour par l'application des règles de croyance, 2. mettre à jour les intentions en supprimant celles qui ne sont plus valides et en ajoutant celles qui deviennent applicables, 3. sélectionner l'intention la plus prioritaire et trouver la règle de stratégie applicable, 4. évaluer la stratégie pour déterminer l'action applicable, 5. effectuer l'action.

+ ALADIN: ALADIN [Ros96, RMP96] est un langage de spécification d'agents basés sur l'architecture INTERRAP et implémenté sur la base du langage Oz. Il fournit les structures nécessaires pour déployer les différentes couches du modèle INTERRAP. Les trois couches et la base de connaissances sont implémentés comme des objets Oz qui sont exécutés dans des threads différents qui s'exécutent en parallèle.

+ AgentSpeak(L), AgentSpeak(XL) et AgentSpeak(F): AgentSpeak(L) [Rao96], proposé en 1996 par Rao, tente de réconcilier théorie et pratique en se reposant sur PRS et dMARS. AgentSpeak(L) est un langage de programmation basé sur une restriction de la logique du premier ordre avec des évènements et des actions. Le comportement de l'agent est dicté par un programme écrit en AgentSpeak(L). AgentSpeak(XL) propose des structures de haut niveau pour la génération automatique des fonctions de sélection d'intentions. Il introduit la notion d'actions internes ne modifiant pas l'environnement et pouvant donc être exécutés instantanément que ce soit, Dans [BFVW03] on propose une approche alternative, introduisant AgentSpeak(F) qui est une version restreinte à état fini d'AgentSpeak(L) dont le but est de faciliter l'application d'un modèle de vérification aux systèmes AgentSpeak(L).

+ Viva: Viva [Wag96] est un langage de programmation orienté agent qui suit le paradigme AOP introduit par Shoham combiné à des concepts issus de Prolog et de SQL. La spécification d'un agent Viva comprend trois parties : la définition d'un schéma, la définition d'un comportement et l'initialisation. Le schéma détermine le langage de la base de connaissance, le comportement est constitué d'un ensemble de règles d'action et de réaction, l'initialisation assigne à l'agent ses croyances et ses tâchesinitiales.

Un agent Viva est composé de quatre éléments : 1. un état mental incluant ses connaissances, ses tâches et ses intentions courantes, 2. une file d'évènements contenant les messages des autres agents et ses perceptions, 3. un ensemble de règles d'action comprenant les différents types d'action disponibles, 4. un ensemble de règles de réaction correspond au comportement réactif et communicatif de l'agent.

+ RETICULAR AGENT DEFINITION LANGUAGE (AKA AGENTBUILDER): AgentBuilder est un environnement commercial pour le développement d'agents. Il est basé sur Reticular Agent Definition Language (RADL) qui est un langage de programmation orienté agent. RADL est inspiré d'Agent-0 et de PLACA. Il est basé sur quatre états mentaux [Dev06 et al] : croyances, capacités, engagements et intentions. Les communications entre agents se font par échanges de messages KQML. Les

engagements définissent explicitement les actions à accomplir tandis que les intentions définissent seulement un état à atteindre et la suite d'actions à effectuer est à la discrétion de l'agent.

+ ZEUS: L'ensemble d'outils Zeus a été conçu pour faciliter le développement rapide d'applications multiagents. L'objectif était de développer des agents capables de délibération, dirigés par les buts, rationnels et versatiles [Dev06 et al]. Les agents adoptent des buts multiples et raisonnent sur leur environnement en utilisant des faits définis par le biais d'ontologies. Un gestionnaire de messages prend en charge les messages entrants, un moteur de coordination gère les buts de l'agent, une base d'accointances fournit un modèle social utilisé par le moteur de coordination, un planificateur et un Ordonnanceur planifient les tâches de l'agents sur la base des décisions prises par le moteur de coordination, une base d'ontologies gère la définition de chaque fait, le contrôleur d'exécution s'occupe de contrôler l'exécution des tâches prévues par le planificateur.

II.4.2.Langages Impératifs

Les approches purement impératives sont moins fréquentes, principalement du fait que les concepts relatifs aux agents sont fortement déclaratifs par nature. Toutefois, de nombreux programmeurs utilisent encore des langages impératifs classiques pour mettre en oeuvre des systèmes multi-agents et les notions attachés aux agents sont implémentées de manière ad-hoc.

+ JACK: JACK [Win05] est développé par une société commerciale nommé Agent Oriented Software. L'idées derrière est qu'il préférable d'introduire les agents comme une extension des objets plutôt que comme un nouveau paradigme révolutionnaire. JACK propose d'étendre le langage Java tant dans sa syntaxe que dans sa sémantique. C'est est un sur-ensemble de Java et en ce sens tous les outils et les bibliothèques offertes par Java restent accessibles en l'état. Syntaxiquement parlant, JACK étant Java de trois façons : 1. ajout de nouveaux types de haut niveau pour déclarer des agents, des ensembles de croyance, des vues, des évènements, des plans et des capacités 2. chaque type de haut niveau est défini en utilisant des déclarations précédés de # qui définissent les nouvelles entités et leurs relations, 3. dans le corps des plans, il propose une série de primitives préfixées par @, telles que l'envoi d'un évènement (@post) ou encore l'attente d'une condition (@wait _f or).

Les agents sont les entités de base de JACK, ils sont spécifiés en définissant les évènements qu'ils peuvent gérer et envoyer, leurs données dont notamment leur base de croyance ainsi que les plans et les capacités qu'ils peuvent utiliser. Une base de croyances est une petite base de données relationnelle enregistrée en mémoire. Il est simple de définir une base de croyances ainsi que définir des requêtes travaillant dessus. Les bases de croyances peuvent également poster des évènements par exemple lorsqu'elles sont modifiées. Les vues sont des bases de croyances virtuelles calculées sur la base d'autres bases de croyances.

Un évènement est un changement qui intervient dans le temps et qui appelle une réponse. Les évènements sont utilisés pour modéliser les messages reçus, l'adoption de nouveaux buts ainsi que des informations provenant de l'environnement. Un plan est une « recette » qui gère un type d'évènement. Les plans spécifient les évènements qu'ils sont à même de gérer, une condition sur le contexte qui décrit les situations dans lesquels le plan peut être utilisé et un corps qui inclut du code Java ainsi que du code JACK et qui est exécuté par le système.

JACK peut être étendu par le biais de plugins, Il est facile d'étendre et de remplacer l'infrastructure de communication utilisée par JACK.

II.4.3.Langages Hybrides

De nombreux langages combinent des aspects déclaratifs et impératifs. Il s'agit généralement de langages déclaratifs par nature qui fournissent en même temps des constructions impératives ou encore qui offre des facilités pour utiliser du code implémenté dans divers langages de programmation impératifs.

+ IMPACT: IMPACT [ESP99, DZ05] a été commencé par V.S. Subrahmanian en 1997. Le projet IMPACT vise à développer une plateforme multi-agents capable de gérer des données hétérogènes et distribuées, de faciliter l'intégration de code hérité et d'être utilisé dans des applications à large échelle. L'un des points principaux d'IMPACT est l'idée d'agentification, i.e. la transformation de code hérité en agents qui communiquent et agissent. Une méthodologie permettant de transformer du code arbitraire en agent a été développée.

Afin d'agentifier du code hérité, IMPACT introduit la notion de programme d'agent écrit avec un langage d'appels de code. Un appel de code peut être vu comme une encapsulation du

code hérité qui représente sous forme logique les conditions et résultats produits par ce code. Ils sont utilisés dans les clauses qui forment le programme d'agent, déterminant les contraintes sur les actions que peuvent effectuer les agents.

GO !: Go ! [Dev06 et al] est un langage de programmation orienté agent qui présente des constructions déclaratives (fonctions et relations) et des constructions impératives (procédure d'actions). Il est basé sur APRIL étendu avec des mécanismes de représentation des connaissances inspirés de la programmation logique. Chaque agent est composé de plusieurs threads qui communiquent directement avec les threads des autres agents et également entre eux par le biais d'objets partagés ce qui permet de coordonner les activités au sein d'un agent. Go ! est fortement typé ce qui réduit les erreurs de programmation. Les comportements des agents sont décrits par des règles de réaction.

Jason: Jason [BHV05] est un interpréteur, implémenté en Java par R. Bordini et J. Hübner, pour une version étendue d'AgentSpeak(L) qui, comme nous l'avons vu, est un langage de programmation orienté agent bien adapté à l'implémentation de systèmes de planification réactifs (reactive planning systems) selon l'architecture BDI. Dans le cadre de ce travail, AgentSpeak a été étendu en un langage de programmation pratique et élégant reposant sur une base formelle rigoureuse et son interpréteur Jason a été implémenté. Du fait de sa formalisation, Jason devrait pouvoir bénéficier des travaux qui ont été menés sur la vérification formelle des agents AgentSpeak(L). Il est disponible en licence LGPL à l'adresse suivante : http://jason.sourceforge.net. Il implémente la sémantique opérationnelle d'AgentSpeak et des extensions qui ont été proposées dont notamment la communication basée sur les actes de langage ainsi que les travaux sur les échanges de plans.

+ JADE: JADE (Java Agent Development framework) [BBCP05] est un environnement logiciel pour développer des systèmes multi-agents en Java conformes aux spécifications FIPA. Nous ne détaillerons pas ici les aspects plateformes, notons toutefois que JADE est léger et versatile et peut être déployé sur un grand nombre de systèmes ainsi que sur des téléphones portables. L'architecture permet aux agents de s'engager dans des actions concurrentes. Une bibliothèque de comportements est fournie et peut être étendue. Les comportements sont implémentés en Java en redéfinissant des méthodes de classe. JADE ne présente pas de support explicite d'états mentaux avancés ni de constructions proches de celles des agents BDI. Une version en license LGPL est disponible sur :

+ Jadex: Jadex [PBL05] est un environnement logiciel pour la création d'agents dirigés par des buts suivant le modèle BDI. Vu de l'extérieur un agent est une boite noire qui reçoit et envoie des messages (figure 4.12). De même que pour les autres systèmes inspirés de PRS, toutes sortes d'évènements (messages, évènement but) servent d'entrées aux mécanismes internes de réaction et de délibération qui aiguillent les évènements vers des plans issus d'une bibliothèque de plans. Dans Jadex, le mécanisme de réaction et de délibération est le seul composant global d'un agent. Tous les autres composants sont groupés en modules réutilisables appelés capacités.
La version courante de Jadex est en licence LGPL et peut être téléchargée sur http://jadex.sourceforge.net/.

II.4.4.Langages Orientés Coordination

+ COOL: (COOrdination Language) [BF95, BF97] est un langage utilisant KQML qui permet de décrire les protocoles d'interaction inter-agents. Il est né de l'idée que les problèmes de coordination peuvent être traités par une représentation explicite des connaissances portant sur les processus d'interaction entre les agents. Ces connaissances permettent de séparer les capacités de l'agent liées à ses interactions sociales de celles liées à la résolution de problèmes individuels.
La conception de COOL repose sur un certain nombre d'hypothèses et d'idées de base. Tout d'abord, les agents autonomes ont leurs propres plans qui leur permettent d'atteindre leurs buts. Les plans des agents représentent de manière explicite les interactions avec les autres agents, ces interactions se font par échanges de messages. Les agents ne peuvent prédire le comportement exact des autres agents, mais ils peuvent déterminer des classes de comportements attendus. De ce fait, les plans des agents sont conditionnels aux actions et réactions possibles des autres agents. Les plans des agents peuvent être incomplets et la connaissance pour les corriger peut être accessible que durant l'exécution. Les agents doivent donc être capables d'étendre et de modifier leurs plans pendant l'exécution.
Une activité de coordination est modélisée par une conversation entre deux ou plusieurs agents, conversation spécifiée par une machine à états finis (automate). Les états de l'automate représentent l'état dans lequel se trouve la conversation. Il y a un état initial et plusieurs états de terminaison. Les messages échangés sont des actes de langage

(KQML étendu). Un ensemble de règles de conversation définit les messages que reçoit un agent dans un état donné, les actions locales à effectuer, les messages à envoyer et les changements d'état. Un ensemble de règles de recouvrement d'erreurs spécifie comment traiter une incompatibilité entre l'état de la conversation et les messages reçus. Un ensemble de règles de continuation décrit comment les agents acceptent les requêtes de nouvelle conversation ou choisissent de poursuivre dans une conversation existante. Les classes de conversation spécifient les états, les règles de conversation, les règles d'erreur spécifiques à un type de conversation.

Les règles contiennent des variables locales ainsi qu'un environnement de conversation persistant qui permet de définir des variables accessibles d'une règle à l'autre. Chaque agent a plusieurs classes de conversation qu'il peut utiliser pour communiquer avec les autres agents. Les conversations courantes sont des instances de ces classes de conversation que l'agent crée lorsqu'il s'engage dans une conversation. Les modèles de conversation de ce type sont donc décrits du point de vu d'un participant, les autres agents ont leur propre modèle ce qui pose des difficultés pour la vérification et la validation des conversations.

Du fait de délais dans la transmission des messages, de messages perdus, ou de messages non attendus, il se peut qu'un message reçu dans une conversation ne puisse être traité par aucune règle. Les agents peuvent alors utiliser des structures de conversations plus élaborées ou effectuer des traitements spécifiés par les règles d'erreur : ignorer le message, initier une conversation avec l'expéditeur pour clarifier la situation, changer l'état de la conversation ou encore rapporter une erreur.

Les messages transmis contiennent un identifiant de conversation qui permet de les orienter dans les bonnes conversations courantes des agents ainsi que de créer de nouvelles conversations.

COOL inclut un mécanisme permettant de gérer des conversations multiples et de maintenir des dépendances entre les conversations (par exemple une conversation qui est en attente de la fin d'une autre conversation). Il est possible de construire des arbres de conversations et de faire circuler les messages d'un niveau à l'autre en y adjoignant éventuellement des annotations le long de leur parcours. Normalement les conversations s'exécutent en parallèle mais il est possible d'interrompre une conversation jusqu'à ce que d'autres conversations atteignent certains états. Des éléments de la théorie de la décision ont également été intégrés afin de construire des plans prenant explicitement en compte l'incertitude liée à l'environnement, l'utilité des

actions et garantissant d'obtenir certaines classes de comportements optimaux pour les agents.

COOL offre également un sous-système qui permet de faire intervenir un agent humain qui pourra intervenir dans certaines conditions, cette intervention sera ensuite enregistré pour servir ultérieurement. Il propose une interface graphique permettant de visualiser et éditer les conversations.

II.4.5.Synthèse

On constate que la plupart des recherches sur les langages de programmation orientés agent sont basées sur des approches déclaratives dont la plupart se reposent fortement sur la logique. Les langages purement impératifs sont peu nombreux ce qui semble peu étonnant dans le sens où ils paraissent inadaptés pour exprimer les abstractions de haut niveau associées aux concepts des agents. Il semble toutefois qu'un effort partagé soit consenti dans le sens d'une meilleure intégration de code écrit dans les langages de programmation impératifs traditionnels. On constate également qu'il est fait un usage abondant d'interpréteurs plutôt que de compilateurs pour les agents. Cela s'explique par la structure sous-jacente des architectures sur lesquels reposent les langages de programmation et permet de conférer plus de flexibilité dans le comportement des agents qui peut être modifié dynamiquement durant l'exécution.

CHAPITRE III.

III LA VISION COGNITIVE.

La vision cognitive, un domaine en pleine croissance, consiste à munir les algorithmes classiques de la vision artificielle par des comportements de raisonnement et de prise de décision, la nature même des problèmes posés par la vision artificielle nécessite le raisonnement et la prise de décision.

III.1 Vision Cognitive.

La vision cognitive est un créneau de recherche récent qui consiste a adopter le paradigme de l'IAD (intelligence artificielle distribuée) et particulièrement les SMA pour la résolution des problèmes de la vision par ordinateur, en effet les algorithmes classiques de vision par ordinateur, aussi compliqués soient-ils, nécessitant un module supplémentaire de prise de décision (rappelons que les problèmes de vision par ordinateur sont des problèmes mal poses selon Hadamard). La vision cognitif consiste a doter ces algorithmes de facultés cognitive, de capacités d'apprentissage, d'adaptation de plusieurs solutions et de la capacité de développer de nouvelles stratégies d'analyse et d'interprétation.

Les fondements de la vision cognitive incluent : L'architecture, le représentation, la mémoire, l'apprentissage, la reconnaissance, le raisonnement, la prise de décision et la communication. Une solution classique consiste a mettre ces algorithmes de vision classiques au service d'une autre couche cognitive dont le rôle est de prendre des décisions et arbitrer en cas d'ambiguïté (ce qui est très fréquent), cette solution présente l'inconvénient fondamentale de retarder la prise de décision et laisser éventuellement la couche inférieure de dérouter sur des solutions erronées, La vision Cognitive est une alternative intéressante dans le sens ou on construit des systèmes dont l'aspect cognitif « Intelligent » est présent dans tous les composants et omniprésent a tout moment,ce qui permet par conséquent une prise de décision au moments opportuns.

L'émergence des systèmes Multiagents décrits précédemment a ouvert à la vision cognitive une nouvelle perspective très prometteuse quant a la conception de systèmes fortement autonomes.

III.2 Exemple d'un système multiagents de suivi de mouvement

L'exemple qu'on va développer est le travail de B.Jung & P.Petta. [JP05] Ce travail illustre bien la faisabilité d'un suivi cognitif , leur point de départ était une implantation du fameux jeux "Shell Game" qui consiste a une camera statique, un ensemble de coupes identiques de couleur noir, une balle orange et une main humaine qui manipule la balle et la dissimule chaque fois sous une coupe puis change la position des coupes, Le système doit répondre a tout moment aux questions "ou est la balle", "Qu'est ce qui cache la balle" et "Quelle est la trajectoires de la main, la balle et les coupes".

La solution classique consiste a implanter en premier lieu trois détecteurs d'objets bases sur la (sélection d'attributs, distribution des couleurs, segmentation, calcul des centres de gravites des zones détectées et calcul des rectangles limites des objets détectes), Ces détecteurs sont utilises par des trackers qui considèrent que les objets détectes dans une trame sont les mêmes dans toute la séquence tant que la distance entre leur centres de gravite d'une trame a une autre reste inférieure a un seuil donne, les informations fournies par ces trackers sont utilisées par un algorithme de raisonnement qui maintient une structure arborescente d'occlusion. Cet arbre enregistre les objets caches par d'autres, pour chaque objet cache, on lui associe virtuellement la même position et le même mouvement que l'objet qui l'a cache. Les nouveaux objets détectes sont supposes en premier lieu comme objets "réapparus" et sont recherches dans l'arborescence, si un appariement est trouve l'objet est détache dans l'arbre de l'objet qui le cache, de cette manière un raisonnement stable sur l'état de la balle est effectue.
La conception d'un système multiagents consiste a définir clairement l'ensemble des agents (rôle et responsabilités), leur environnement, et surtout leur interaction (police de communication) qui constitue le point névralgique du SMA, Dans ce qui suit on va détaille la solution par agent proposée par les auteurs.

III.2.1 Spécification des agents.

Le système est organise en trois types d'agents : un agent serveur d'image qui est connecte a une camera ou extrait les images a partir d'une séquence sauvegardée (temps réel ou post-production), un ensemble (trois) agents détecteurs, et un ensemble d'agents objet :

Les agents détecteurs : Un agent détecteur est responsable de la détection d'un type donne d'objet dans les images provenant de l'agent serveur d'image. Il dispatche les informations concernant les objets détectes aux agents objet du même type, Ces derniers qui satisfaisaient des critères d'adjacence (centre de gravite) répondent a l'agent détecteur (en lui transmettons le paramètre d'écart du centre de gravite) par un message spécifique, ce dernier règle le conflit en répondant a l'agent qui optimise le mieux le critère d'adjacence, en cas d'objet non identifie, l'agent détecteur instancie une agent objet en l'initialisant par la zone détectée.
Entre autre, ces agents détecteurs sont responsable de la coordination des deux phases suivi et raisonnement, (débute et termine chacune des deux phases).

Les agents Objet : Un agent objet représente un objet détecté dans la scène et associe a un agent détecteur, l'agent objet est responsable de la correspondance entre une zone détectée par l'agent détecteur et une zone maintenu localement concernant la plus récente région qui le représente dans la scène, La distance entre le centre de gravite de cette zone et le centre de gravite de la zone transmise par l'agent détecteur pour identification sert de mesure de cohérence, si elle est inférieure a un seuil donne l'agent objet envoi un message a l'agent détecteur l'informant qu'il est candidat pour la correspondance.

Les agents objet gèrent également les événements de disparition et de réapparition par les opérations de lier/Libéré entre eux-mêmes et les objets qui les cachent.

Une fois deux agents objets sont lies, un contrat est établit entre eux, l'objet au premier plan (visible) propage les informations de son déplacement a l'objet a l'arrière plan (occulté) et informe son agent détecteur (de l'objet caché) de sa région d'intérêt qui correspond a un rectangle englobant l'objet occulté a la position de sa disparition, de cette manière la réapparition est détectée et aucun agent objet n'est instancié. Dans une approche classique, un objet perdu de la scène pendant un certain temps est éliminé de l'arbre, sa réapparition provoquera une batterie de calculs inutiles.

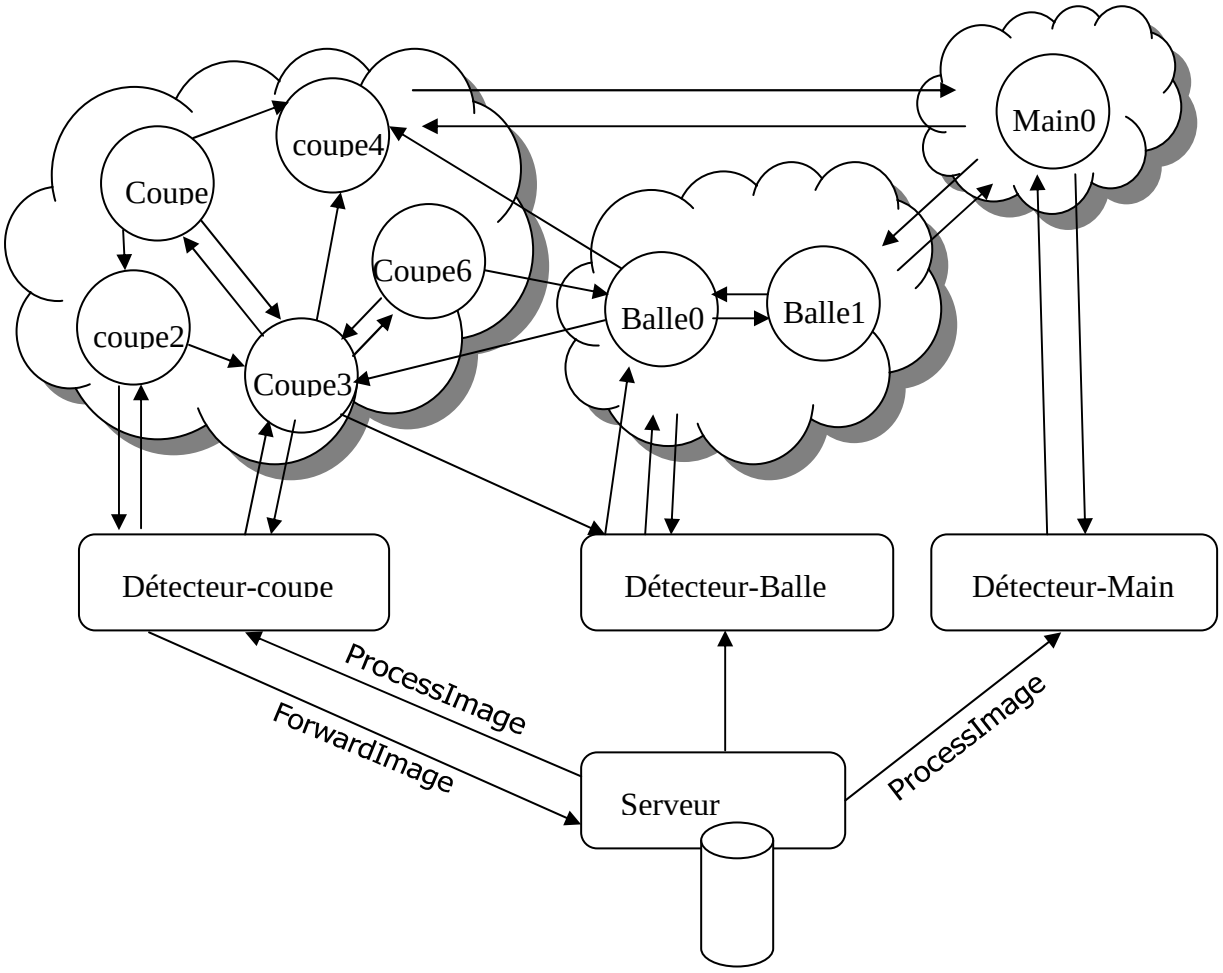


Fig 3.1 Architecture du système

III.2.2 Spécification du schéma de communication.

Le but de la coordination entre les agents [Lesser] est d'assurer un comportement cohérent d'un système bâtit a partir d'agents autonomes qui exécutent des taches indépendantes i.e se consenter pour travailler sur les mêmes sous problèmes, la spécification du schéma de communication est une tache très délicate dans la conception d'un SMA, elle consiste a détecter les interdépendances inhérentes aux agents pour la réalisation d'un but commun.

Dans ce cadre de travail, la communication est articulée autour des deux phases "Suivi" et "Raisonnement" :

La phase suivi débute avec l'agent Serveur d'image qui envoi le Message "*ProcessImage*" aux (trois) agents détecteur présents dans le système, dans le cas ou u agent détecteur n'identifie aucune région d'intérêt (selon ses connaissances par ex : l'agent détecteur de coupes ne cherche que les régions présentant un motif couleur noir) il envoi le message "*NoBlobFound*" a tous les agents objet qui lui y sont attachés, le cas échéant, il leur envoi un message "*DetectedBlobs*", les agents objet qui recevront ce message répondrons tous par le message "*BlobSelection*" comportant la position de la zone qu'ils représentent et une mesure confidentielle (L'écart entre le centre de gravité de la zone détectée et leur zone interne, si cet écart est supérieur a un seuil donné il transmettent -1 pour indiquer que la zone ne les intéressent pas), l'agent détecteur, une fois toutes les réponses sont présentes, lève l'ambiguïté et envoi le message "*ConfirmBlob*" a l'agent objet le mieux disant. Dans le cas ou tous les agents objet répondent par le paramètre -1, l'agent détecteur instancie un nouvel agent objet en l'initialisant par la zone détectée.

Il est possible que l'agent détecteur identifie plusieurs régions, dans l'itération suivante, il n'envoi le message "*DetectedBlobs*" qu'aux agents n'ayant pas fait objet d'un message "*ConfirmBlob*" auparavant.

Une fois que tous les agents détecteurs achèvent leurs itérations avec toutes les zones détectées, la phase de raisonnement commence.

Cette phase commence par l'envoi du message "*StartReasoning"* de l'agent détecteur a tous les agents objets qu'ils gèrent, chaque objet réapparu envoi le message "*RemoveLink*" a l'agent objet qui l'a caché, ce dernier confirme l'opération par retour d'un message "*LinkRemoved*", l'étape suivante consiste a ce que tous les agents objet restant qui cachent d'autres objets envoile message "*MoveSubObject*" a l'agent objet occulté (pour que ce dernier se déplace avec l'objet qui le cache), ce dernier répercute ce message éventuellement aux agents objets qui les cachent a son tour et ainsi de suite. l'étape suivante consiste a identifier les agents objet qui cachent tous les objets disparus de la scène,pour cela, chacun des agents objet disparus diffuse le message "*LookForHider*" a tous les agents détecteur qui envoient le message "*CanHide*" a tous les agents objet qu'ils gèrent et qui sont assez proche de la zone de disparition de l'objet initiateur du message "*LookForHider*", les agents objets envoient le message "*HiderFound*" ou "*HiderNotFound*" selon une mesure interne d'écart entre les centres

de gravité par rapport a un seuil donné, en cas ou plusieurs agents objet envoient "*HiderFound*", l'agent objet disparu de la scène prend le mieux disant et lui envoi le message "*SetLink*" ce message est confirmé par le récepteur en retournant le message "*LinkSet"* , les agents objet envoient enfin le message "*Update Area*" a leurs agents détecteur, qui terminent la phase de raisonnement en envoyant le message "*ForwardImage*" a l'agent serveur afin de récupérer l'image suivante.

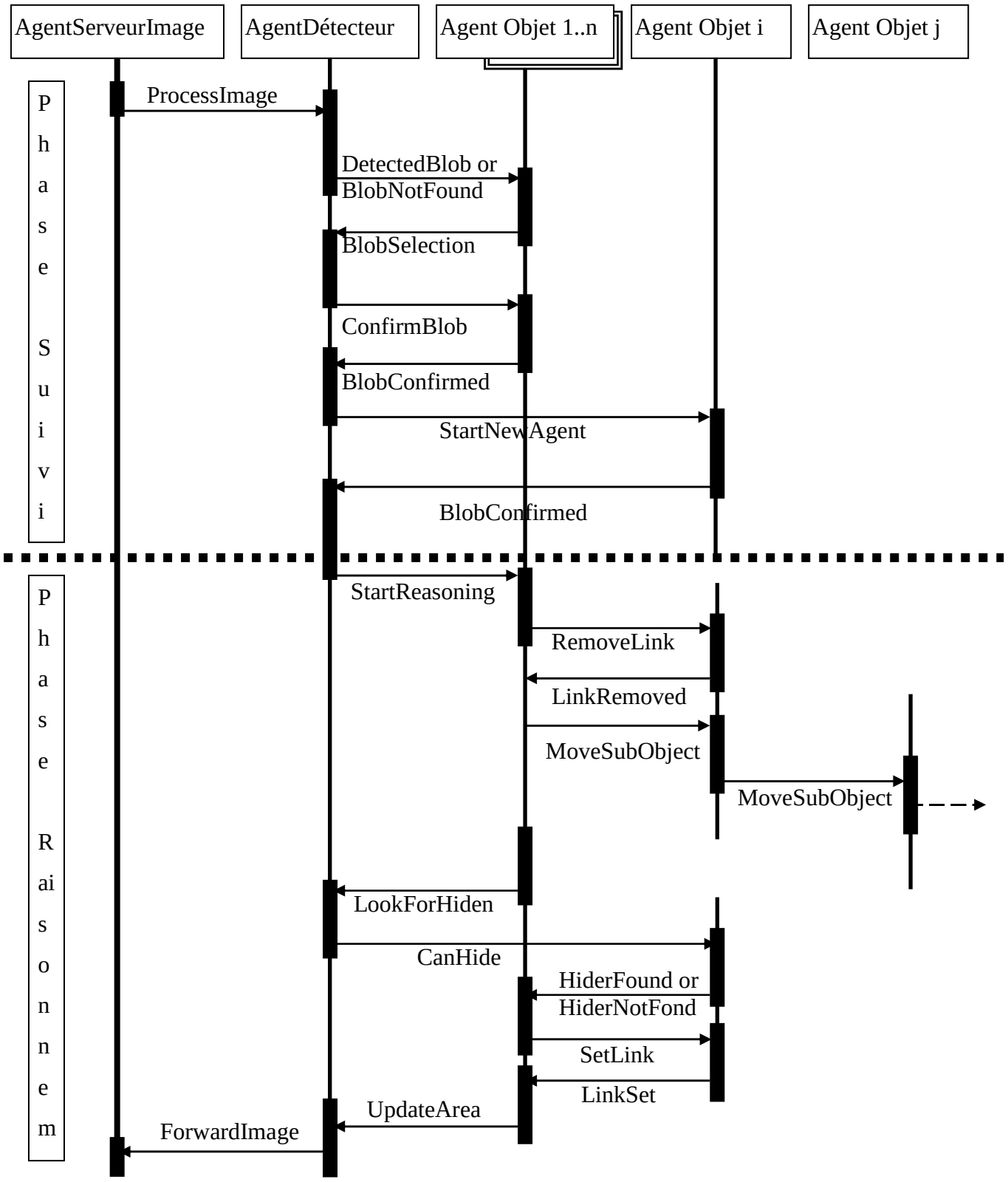


Fig. 3.2 Conversassions inter-agents

III.2.3 Résultats.

Les résultats obtenus par les auteurs sont trois fois plus rapides que la solution classique (9.1 s contre 3.0 s) implantée par un code généré par MATLAB, une réalisation classique en dehors du MATLAB (code en C/C++) est plus rapide, ces résultats ne signifient pas forcément que la solution par agent est plus rapide autant qu'elle montre la viabilité et la meilleure structuration de telles approches.

III.3 Exemple 2: Vision active distribuée.

Dans [UM02] Un système temps réel de vision coopérative distribuée en 3D (VCD) constitué d'un ensemble d'agent de vision active (AVA) communicants est proposé, chaque AVA représente une machine dans un réseau connectée a une camera active. Ce type de system a plusieurs avantages: espace d'observation élargi, robustesse, flexibilité de l'organisation et la compensation des défaillances.

Un groupe d'AVAs appelé agence prend en charge le suivi d'un seul objet ou 1. Chaque AVA possède une camera fixe et 2. Les paramètres externes de la camera (les coordonnées 3D de la camera) sont calibrés.

La détection des zones en mouvements s'effectue par différence avec l'image arrière plan générée antérieurement. Le système est conçu de la manière suivante:

1. Tous les AVAs effectuent leur recherche d'une manière autonome.
2. un AVA (dénoté par **AVA_i)** qui détecte un objet, le considère comme cible et diffuse les lignes 3D (dénoté par L_i). Une agence est formée.
3. Chaque AVA cherche l'objet sur la zone Li après réception du message de diffusion. Si un AVA (dénote par **AVA_j)** détecte une région en différence sur L_i, il répond à AVAi avec les lignes 3D (dénoté par L_j). Pour rendre cette méthode applicable au sequence réelle, la différenciation avec l'arrière plan robuste au changement d'illumination. Ce sujet est traite par plusieurs travaux.
4. AVA_i calcule la distance entre L_i e L_j, si elle est majore par un seuil s. l'objet détecté par AVA_i et AVA_j sont considérés comme le même objet. Le point moyen entre L_i et L_j est considéré comme la position 3D de l'objet.
5. AVA_i diffuse la position 3D a tous les AVAs de l'agence**.**
6. Répétition l'identification de l'objet permet a tous les AVAs de suivre l'objet cible.
7. Quand tous les AVAs échouent dans le processus d'identification de l'objet décrit précédemment, ils recommencent la recherche pour un autre objet.

AVA joue le rôle de maître de l'agence et les autres sont appelées les travailleurs. L'autorité du maître de l'agence est transférée dynamiquement à un AVA travailleur qui détecte le premier une région suspecte.

CHAPITRE IV.

IV UN SYSTEME MULTIAGENTS POUR LE SUIVI

Notre travail porte sur la conception d'un système multi-agent pour le suivi, le cadre applicatif concerne les séquences vidéo où un arrière plan est connu au préalable (cas de la vidéosurveillance, la surveillance des trafics routiers, …etc). Les objectifs attendus de notre système sont : 1- détecter les objets en mouvement, 2- fournir des coordonnées dans le référentiel image de ces objets, 3- gérer les cas d'occultation. Le système qu'on a mis en œuvre répond d'une manière exacte aux objectifs 1 et 2, et relativement (saufs dans certains cas qu'on va détailler) à l'objectif 3.

IV.1 Cadre applicatif.

On se place dans le cadre des applications de suivi du mouvement avec arrière plan fixe (cas de la télésurveillance, surveillance de trafic routier,…etc). La détection peut être réaliser donc, par différence de l'image courante I_t avec l'arrière plan fixe qu'on notera BG, dans les séquences réelles deux problèmes peuvent produire des régions en mouvement erronées: le bruit et le changement d'illumination (changement d'éclairage de la scène). Tandis que le premier peut être contourné plus au moins par utilisation d'un filtre de lissage (un filtre médian par exemple), le deuxième requiert l'utilisation d'algorithmes plus sophistiqués (utilisation des rapports des composantes RGB pour les images couleurs par exemple).

En premier lieu, on ne va considerer que le cas d'une camera unique et fixe, par la suite on étudiera la possibilité d'extension a plusieurs cameras.

IV.2 Conception du SMA

Pour concevoir le SMA, on a adopté l'approche voyelle (AEIO) qui consiste à spécifier chacune des dimensions du SMA: A (Agent), E (Environnement), I (Interaction) et O (Organisation).

IV.2.1 Définition des agents.

On a choisi l'implantation de deux agents permanents (AgentServeur AS et AgentDetecteur AD) et un vecteur dynamique d'agents objets AgentObjet (AO) initialement vide.

IV.2.1.1 L'agent Serveur AS.

L'AS est chargé de : 1- Initialiser le système de suivi avec l'arrière plan, 2- l'acquisition d'une occurrence d'image I a partir d'une camera fixe (dans notre implantation à partir d'une base d'images pré-stockée), la transmettre à l'AD, 3- mettre fin au processus de suivi. Malgré ce comportement élémentaire, on a choisi d'implanter l'AS pour une future extension où des comportements plus complexes peuvent être greffés (migration dans le cas de plusieurs caméras ou robots mobiles, traitement préalables sur les images …etc.).

IV.2.1.2 L'agent Detecteur AD.

L'AD est chargé de :

1- la détection d'objet en mouvement (sous forme de zones d'images encadrées par un rectangle),

2- l'instanciation des AO en cas de détection d'objets nouveau dans la scène,

3- le maintien d'une structure indicatrice sur l'état des AO (présent, perdu),

4- l'appariement de l'ensemble des zones détectées avec l'ensemble des AO présents à l'instant t.

- **Détection des objets en mouvement:**

A la réception d'un message indiquant le traitement d'une occurrence d'image It par l'agent détecteur, ce dernier détecte en premier lieu les zones en mouvement par rapport a l'arrière plan BG.

A) Calcul de l'image de différence Id=It-BG, Cette image de différence est constituée de pixels dont l'intensité est nulle sauf pour les zones en mouvement.

B) Agrégation des pixels d'intensité non nulle de Id en zone, ceci est réalisé par l'étiquetage de chaque pixels par un indice de zone d'appartenance, deux pixels avec le même indice appartient a la même zone, l'étiquetage est fait on procédant a un regroupement itératif des pixels.

C) Création d'un vecteur dont chaque élément représente les coordonnées du rectangle encadrant une des zones détectées précédemment, ceci est fait par le choix des Min et Max des groupes de pixels ayant le même indice de zone.

D) Elimination des zones dont la taille est inférieure a un seuil donne, ceci est justifié pour palier aux bruits éventuels.

A la fin de cette étape l'AD dispose d'un vecteur contenant les coordonnées des rectangles encadrant les zones en mouvement paré pour l'étape d'appariement avec les objets détectés dans l'image précédente.

- **Appariement:**

Dans cette étape, l'AD tente de trouver une correspondance entre les zones détectées dans l'image courante avec les objets détectés dans l'image précédente, les

coordonnées de ces derniers sont maintenues par les AO présents dans e système a cet instant, pour faire, on a adopté l'hypothèse du faible déplacement ou on suppose que les objets effectuent un mouvement relativement faible (i.e inférieur a un seuil fixé raisonnablement) entre deux images successives de la séquence, pour chaque zone figurant dans le vecteur, l'AD transmis a tous les agents objets présents les coordonnées du rectangle correspondant, les agents objets répondent par un paramètre représentant l'écart entre cette zone et leur position, l'AD choisi celui qui minimisant le mieux ce paramètre.

IV.2.1.3 L'Agent Objet AO.

L'AO est un agent qui représente un objet détecté dans la scène, il est chargé de :

1- maintenir une structure qui représente les caractéristiques courantes de l'objet (dans notre implantation on a restreint cette information aux coordonnées du rectangle encadrant l'objet et à son état : Visible, Perdu, Réapparu, d'autres informations peuvent être envisagées comme la distribution des valeurs du vecteur RGB de l'objet et ce dans le but d'effectuer des appariements plus fines),

2- de fournir à l'AD une réponse sous forme d'un paramètre de sélection qui permettra à ce dernier l'appariement avec une zone détectée de l'image.

IV.2.2 Spécification de l'environnement

L'environnement dans notre applicatif est constitué de l'ensemble des agents et de la séquence d'images, comme il est mentionné ; l'ensemble des agents est constitué de deux agents AS et AD et un ensemble dynamique d'AO.

IV.2.3 Interaction inter-agents

Vu sa simplicité de mise en œuvre, le modèle de communication choisi est l'envoi de message.

Au début l'AS envoi un message « DemarreSuivi » à l'AD avec une image qui représente l'arrière plan, ce dernier, une fois les initialisations nécessaires effectué répond par le message « PretPourProchaine », l'AS transmis la première image de la

séquence via le message « TraiteImage » adressé à l'AD, ce dernier effectue une segmentation de la différence (image en cours et arrière plan) en régions contiguës, puis encadre chaque région avec un rectangle qui représente les extrémités de l'objet augmentées de 2 pixels, ensuite il instancie des AO au nombre des régions détectées et avec les coordonnées du cadrant correspondant puis renvoi le message « PretPourProchaine » à l'AS.

Au cours de chaque itération t, l'AS envoi le message « TraiteImage » à l'AD avec l'image suivante, l'AD effectue une segmentation de l'image différence décrite précédemment.

Pour chaque zone détectée, l'AD procède de la manière suivante :

Construction d'un message «DonneEcart » avec les coordonnées du cadran de ladite zone, ce message est diffusé à tous les agents objets présents dans le système à l'instant t, les agents Objets qui reçoivent ce message, calculent la différence entre leurs centres de gravités et le centre de gravité de la zone transmise, et doivent répondre par le message « Ecart ».

L'AD, une fois toutes les réponses collectées, sélectionne le minimum déplacement parmi les réponses reçues, trois cas de figure à traiter :

1- Toutes les réponses sont supérieure à un seuil donné : dans ce cas l'AD constate qu'il s'agit d'un nouvel objet dans la scène et instancie par conséquence un nouvel AO avec paramètres les coordonnées de la zone en cours.

2- Un seul objet offre le minimum déplacement : dans ce cas l'AD envoi le message « ActualisePosition » à l'agent concerné, ce dernier actualise sa position.

3- Plusieurs Objets offrent le même écart : le premier est choisi, l'AD lui envoi le message « ActualisePosition » pour que ce dernier mette à jour sa position.

Ce comportement s'itère pour toutes les zones, après que toutes les zones soient traitées, il se peut que certains objets ne s'apparient avec aucune zone, pour ces objets, l'AD considère qu'il sont « perdus » et leurs envoie les message « EtatPerdu », ces derniers mettent à jours leur état à « Perdu »

Enfin, et lorsque la séquence est épuisée, l'AS envoi le message "StopSuivi" a l'AD ce qui provoque l'arret du processus de tracking.

IV.2.4 Organisation

Cette dimension exprime, comme il a été mentionné au chapitre précédent, l'ensemble des règles et des contraintes permettant de structurer le système multiagents.

Dans notre cas, l'AS est un simple agent réactif qui démarre le système, fourni à la demande l'image suivante de la séquence et met fin au processus de tracking. Le modèle choisi de l'organisation entre l'AD et les AO est le modèle de réseaux contractuels (appel d'offre, établissement de contrat, résiliation), pour chaque zone détectée, l'AD lance un appel d'offre aux différents AO existant, l'AD établi un contrat avec l'agent le mieux disant (minimisant l'écart de déplacement), le contrat consiste a réajuster les coordonnées de l'agent objet. Une fois effectuer, l'AD met fin au contrat.

IV.3 Schéma fonctionnel du système.

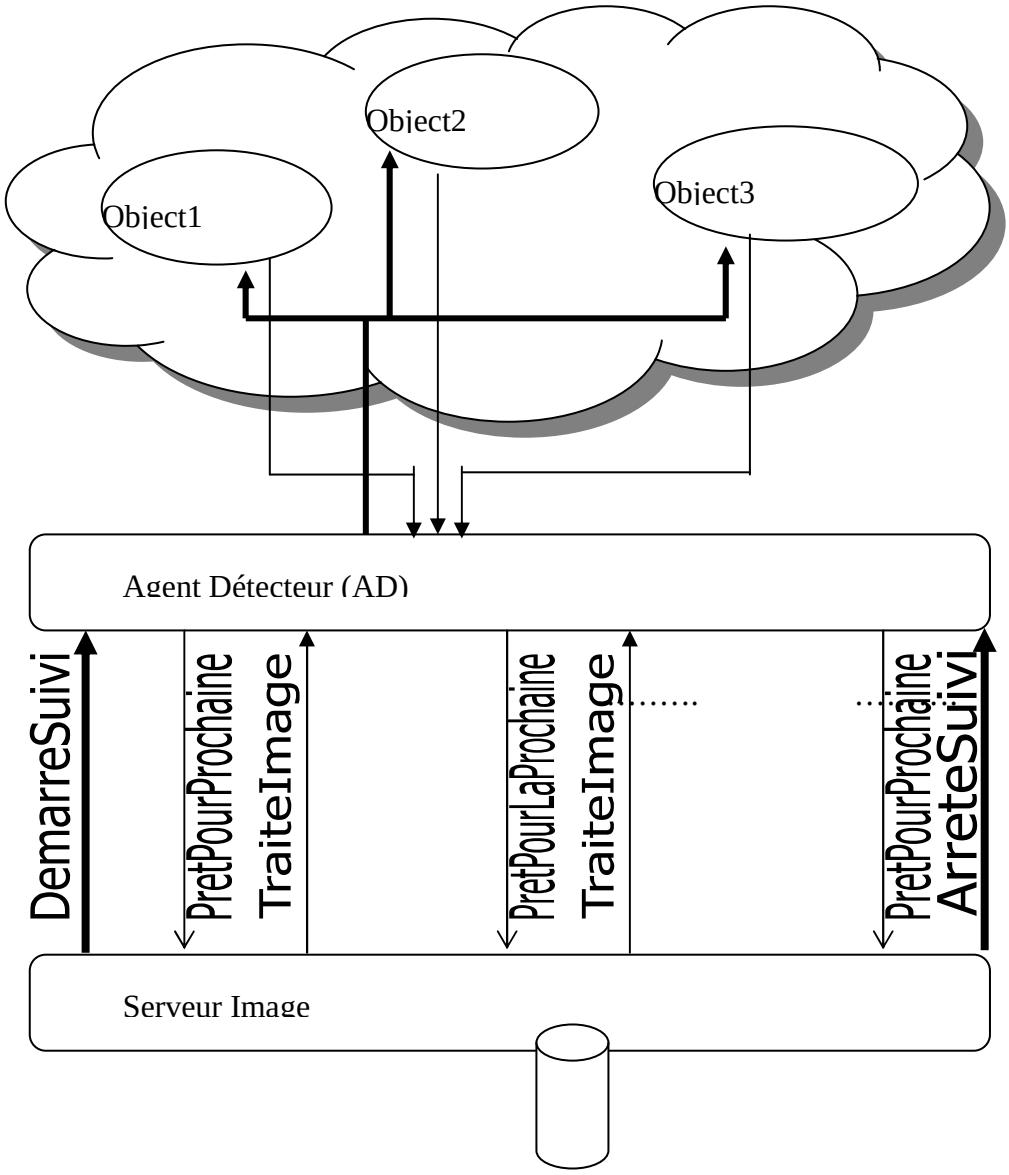


Fig 4.1: Architecture du système proposé

Ce schéma explique bien le fonctionnement du système.

La figure suivante, extraite de l'agent Sniffer de la plate de forme Jade utilisée dans notre système, illustre l'interaction des agents dans notre système (Traitement d'une image): au début,l'AS envoi a l'AD l'image en cours l'AD détecte trois zones,

pour la première, il envoi un message aux AO existant (Objcet1, Object2 et Object3), les trois AO envoient leurs offres, l'AO Object2 est choisi, pour la deuxième zone le message n'est envoyé qu'aux AO non encor appariés (Object1 et Object3), ces deux répondent par leurs offres et c'est l'AO Object1 qui est choisi, enfin, la troisième zone est appariée avec l'AO Object3.

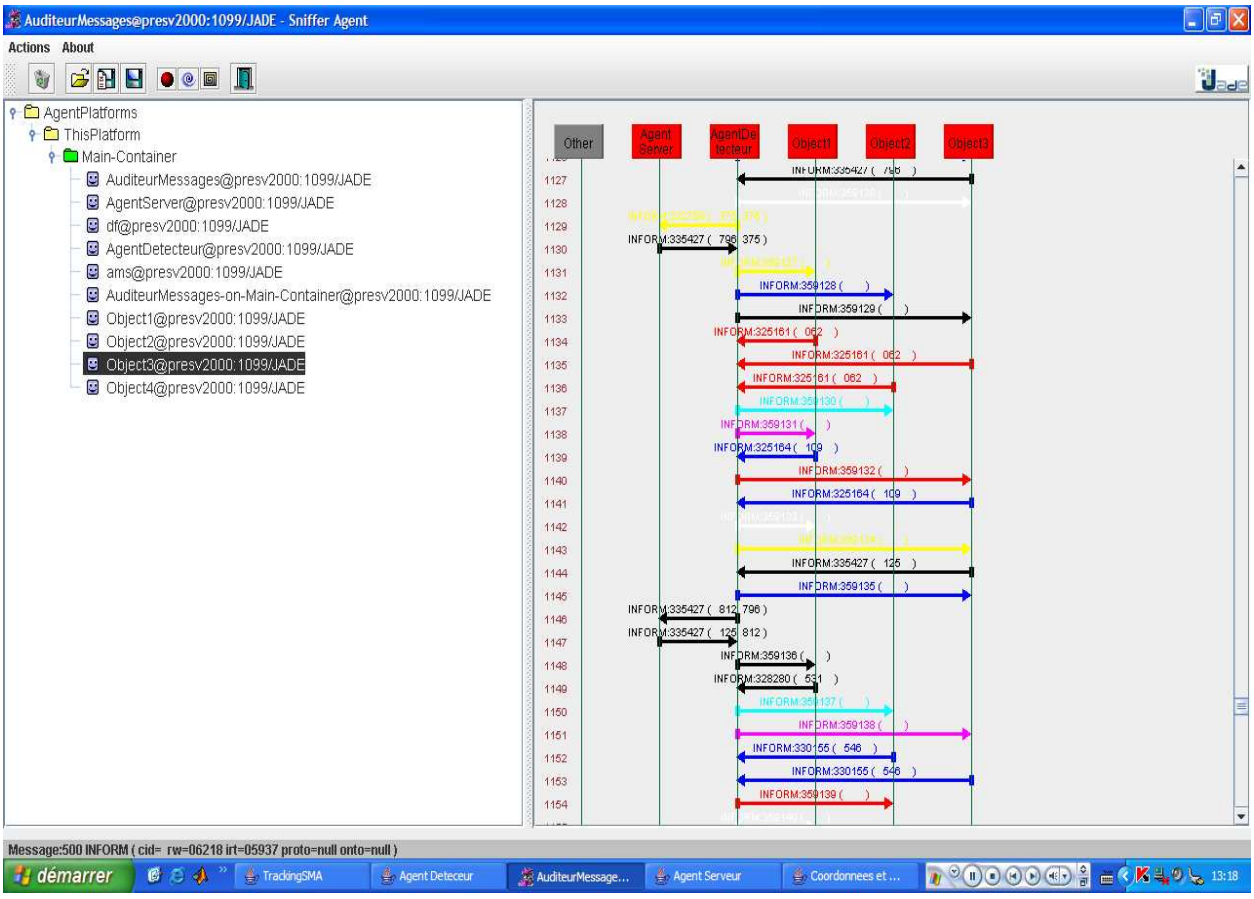


Fig. 4.2 : Echange de messages entre agents.

IV.4 Expérimentations.

IV.4.1 Implantation.

Notre système de suivi est écrit en JAVA. Utilisant la plate de forme de développement des systèmes multiagents JADE^{TM} conformément aux spécifications FIPA [FIPA98], cette plate de forme, qui est un logiciel libre distribué par TILab en Open Source avec une licence LGPL, se présente sous forme d'une multitude de package facilitant le développement des SMA. Ces packages offrent les outils nécessaires pour décrire des

agents et gérer leur cycle de vie, ajouter des comportements à ces derniers, publication des compétences des agents grâce a la notion de pages jaunes introduite par FIPA, de faire communiquer les agents via les ACLMessage, définir des sémantiques de messages grâce aux ontologies qui peuvent être intégrées aux ACLMessage, migrer les agents d'un site a un autre ...etc.,

Les principaux packages sont:

- jade.core /implémente le Kernel du système JADE, il inclus la classe d'agent Agent qui doit être héritée par les nouvelle classes Agents de l'utilisateur, il inclut une hiérarchie de classe comportement (Behaviour Class dans le sous package jade.core.behaviours) . Les comportements représente les tâches a effectuer par les agents. L'utilisateur oit définir une ou plusieurs classes d'objet Behaviuors et mes ajouter à la classe d'agent qu'il a défini.
- jade.lang.acl : ce sous package implémente la communication entre agent conformément au standard ACL (Agent Commincation Language) de FIPA.
- jade.content : contient un ensemble de classes qui supporte les ontologies (définition des contenus de langages) définies par les utilisateurs. Dans le package particulier jade.content.lang.sl on trouve l'analyseur et l'encodeur des messages ACL SL codec2.
- jade.domain contient toutes les classes Java qui représentent l'AMS (Agent Management entities) défini par le standard FIPA ,
- jade.gui contient un ensemble générique de classes utiles à crées des interfaces graphiques pour afficher et éditer les identificateurs des agents, leur description, les messages ACL … etc.
- jade.proto : ce package contient des classes pour modéliser les protocoles d'interactions entre agents (i.e. fipa-request, fipa-query, fipa-contract-net, fipa-subscribe … etc).
- jade.wrapper permet l'utilisation de JADE comme étant une bibliothèque pour les applications JAVA en leur permettant de lancer des agents JADE et des conteneurs d'agent.

JADE est livrée avec un certain nombre d'outils d'aide pour l'administration et le contrôle des aents lancés, RMA (Remote Management Agent) constitue une interface

graphique qui permet de visualiser les différents conteneurs et agents présents sur la plate de forme, entre autre elle permet de lancer un certain nombre d'agent spécifique permettant le test et le contrôle des autres agents comme le *DummyAgent* qui est un outil permettant le débogage du comportement des agents faces aux messages, il permet d'envoyer un message ACL à tout agent présent sur la plate de forme en spécifiant son contenu et éventuellement l'ontologie qu'il utilise, le *Sniffer* permet de visualiser graphiquement les messages ACL échangés entre les agents,

JADE est une plate de forme distribuée, elle peut s'exécuter sur plusieurs machines hôtes, chaque instance exécutée de JADE est appelé container (conteneur) qui peut héberger plusieurs agents, l'ensemble des containers actifs sur une machine est appelée plate de forme qui doit inclure obligatoirement un conteneur principal (main container) les autres sont des conteneurs fils, les conteneurs exécuter sur d'autres machines doivent s'enregistrer auprès du conteneur principal (en fournissant son adresse IP et son port qui est 1099 par défaut). Si un autre conteneur principal est démarré dans une machine quelconque JADE considère qu'il s'agit d'une autre plate de forme, la figure suivante illustre ce fonctionnement :

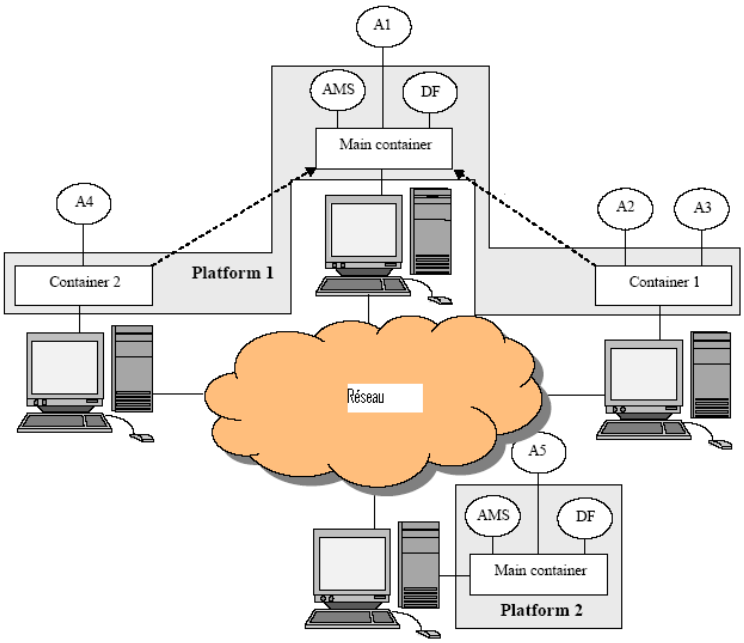


Fig 4.3 conteneurs et plates de forme JADE

Les agents A1, A2, A3, A4 appartient à la première plate de forme, A5 appartient a la deuxième plate de forme.

Utilisation :

Pour intégrer JADE dans une application Java, il existe deux manières de procéder:

1- extension de la classe Jade.core.Agent, dans ce cas l'application est une classe dérivée de Jade.code.Agent, pour l'exécuter, il faut lancer au préalable la plate de forme Jade par la ligne de commande :

Java Jade.Boot –gui (-gui pour lancer en même temps le RMA), dans l'IDE de développement, il est nécessaire de déclarer la classe Jade.Boot comme classe principale de l'application (main).

2- Intégration du Runtime JADE dans l'application (comme dans notre implantation), dans ce cas cette dernière doit prendre en charge le lancement du Kernel de JADE la suite d'instructions suivante montre comment intégrer le Runtime JADE dans une application JAVA:

```
import jade.core.Runtime;
import jade.core.Profile;
import jade.wrapper.*;
...
// créer une instance du Runtime JADE
Runtime rt = Runtime.instance();
Profile p = new ProfileImpl();
cc=rt.createMainContainer(p);
```

Les agents de notre système, sont tous dérives de la classe Agent du package jade.core.

L'agent serveur AS possède un seul comportement dérivé de la classe OneShotBehaviour (une seule exécution), dans lequel, l'AS démarre le processus de suivi (initialisations nécessaires), envoi en boucle les images de la séquence a l'AD, et a l'épuisement de la séquence, met fin au processus.

Le comportement de l'AD est implanté sous forme de machine a états finis, dérivé de la classe CompositeBehaviour qui enchaîne l'exécution deux objets de type OneShotBehaviour et CyclicBehaviour (Exécution en boucle dans une file d'attente),

Les AOs comportent deux comportements dont le premier ajuste la position de la zone de l'objet représenté dans la séquence a la demande de l'AD et Le deuxième effectue en premier lieu la détection des zones en mouvement, puis interroge les AOs pour chaque zone détectée pour apparier lesdites zones.

Le système comporte en plus, deux agents utilitaires, le RMA qui permet de visualiser les agents présents dans le système et le sniffer "Auditeur de message" qui permet de visualiser l'interaction entre les agents (messages échangés).

IV.4.2 Résultats Expérimentaux.

Le système qu'on a mis en œuvre est testé sur deux …. Séquences synthétiques:

- La première représente une balle de ping-pong rebondissant sur une table. Pour ce cas (un seul objet en mouvement), notre système a detecté correctement la balle en fournissant ses coordonnées tout au long de la séquence, les images suivantes représentent quelques occurrences de la séquence et le résultat du suivi:

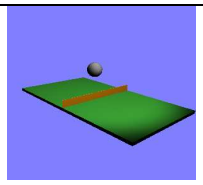
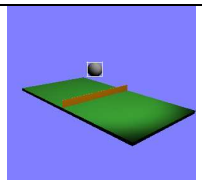
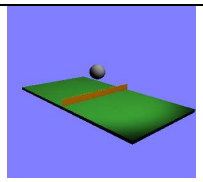
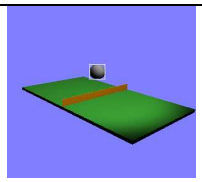
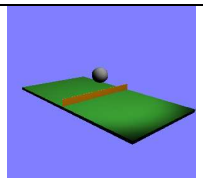
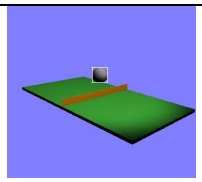
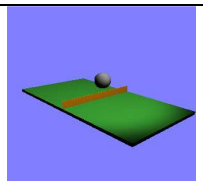
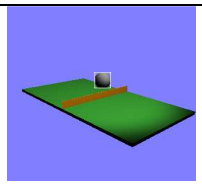
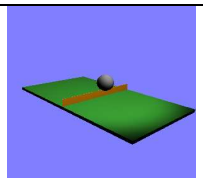
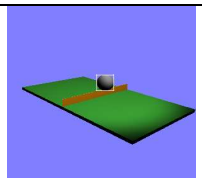
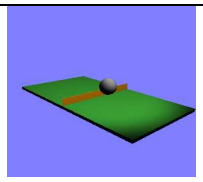
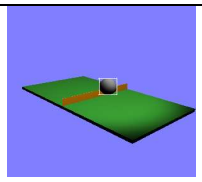
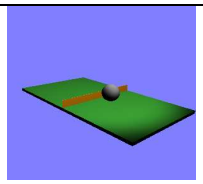
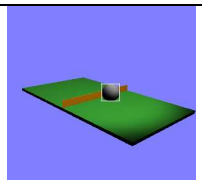
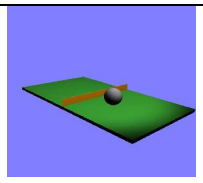
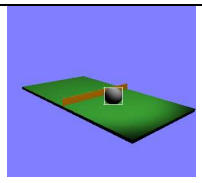
N-séquence	Image	Suivi	Observations
99			- Détection correcte
100			- Détection correcte
101			- Détection correcte
102			- Détection correcte
103			- Détection correcte
104			- Détection correcte
105			- Détection correcte
106			- Détection correcte

Fig 4.4: 8 images de la première séquence. Un seul objet en mouvement.

- Dans la deuxième séquence, on a voulu testé notre système sur le maximum des cas possibles (apparition, occultation complète, occultation partielle, disparition et réapparition), pour cela, on a choisi une séquence composé de trois avions de chasse évoluant dans un ciel synthétique (séquence générée par l'environnement 3DSMAX), les trois avions apparaissent l'une après l'autre, puis un des trois appareils disparaisse, tandis que l'un des deux autres occulte le troisième puis se détachent et enfin le troisième appareil (disparu a gauche) réapparu en haut de l'image:

N-séquence	Image	Suivi	Observations
029			- Entrée partielle du premier appareil: détection correcte
032			- Détection correcte
034			- premier appareil en complet dans la séquence: détection correcte
040			- Entrée partiel du deuxième appareil: Détection correcte
041			- Entrée partielle du troisième appareil: Détection correcte
044			- Détection correcte
047			- troisième appareil en complet: Détection correcte
068			- Les trois appareils en plein ciel: Détection correcte

Fig.4.5 Images de la 2eme séquence: Trois objets en mouvement.

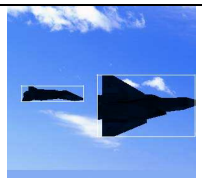
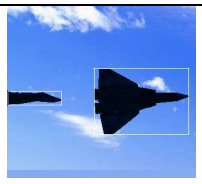
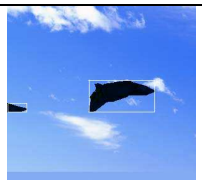
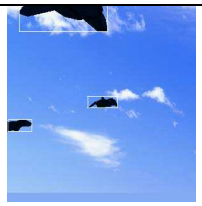
N-séquence	Image	Suivi	Observations
0118			- Collision de deux appareils et disparition partielle d'un troisième, deux objets sont détecté
0119			- Idem
0120			- Idem
0132			- Troisième appareil disparu a gauche, réapparition de l'appareil caché.
0136			- les deux appareils totalement disjoints.
0139			- Appareil 2 caché partiellement
0146			- Idem
0148			- Idem

Fig.4.6 Images de la 2eme séquence: Cas de disparition et d'occultation.

N-séquence	Image	Suivi	Observations
0163			- Deux appareils dont un apparu partiellement
0164			- Idem
0166			- Apparition du troisième appareil en haut: vu par le système comme un nouvel objet.
0168			- Idem
0170			- Apparition d'une partie du même objet indépendamment, Détection incorrecte
0171			- correction du faux appariement, le faux objet de l'image 0170 est considère comme perdu
0172			- Détection correcte

0177			- Détection correcte

Fig. 4.7 Images de la 2eme séquence: Cas des réapparitions

On peut considérer que le système est relativement fiable aux occultations et aux disparitions/Réapparition.

IV.4.3 Les Performances.

L'objectif des mesures de performances qu'on a expérimenté était de tester les temps de réponse relatifs de quelques scénarios, ces temps de réponse ne constituent en aucun cas un critère de comparaison avec d'autres approches vu la spécificité du cas traité d'une part, et d'autre part, ces mesures ne représentent pas exclusivement le temps de réponse pour le suivi proprement dit, ils représentent l'écart du temps écoulé entre le déclenchement du suivi pour une image et son achèvement pour la même image y compris les routines de manipulation de l'interface graphique.

La première mesure prise caractérise les temps de réponse pour chaque image de la première séquence (celle de la balle), la balle était d'une taille relativement grande, les images de la séquence sont de type couleur RGB 24bit/Pixel et de taille 300X300 pixels, la moyenne des réponses par image est de 356,235 millisecondes, le graphique ci après représente la distribution des temps de réponse sur la séquence.

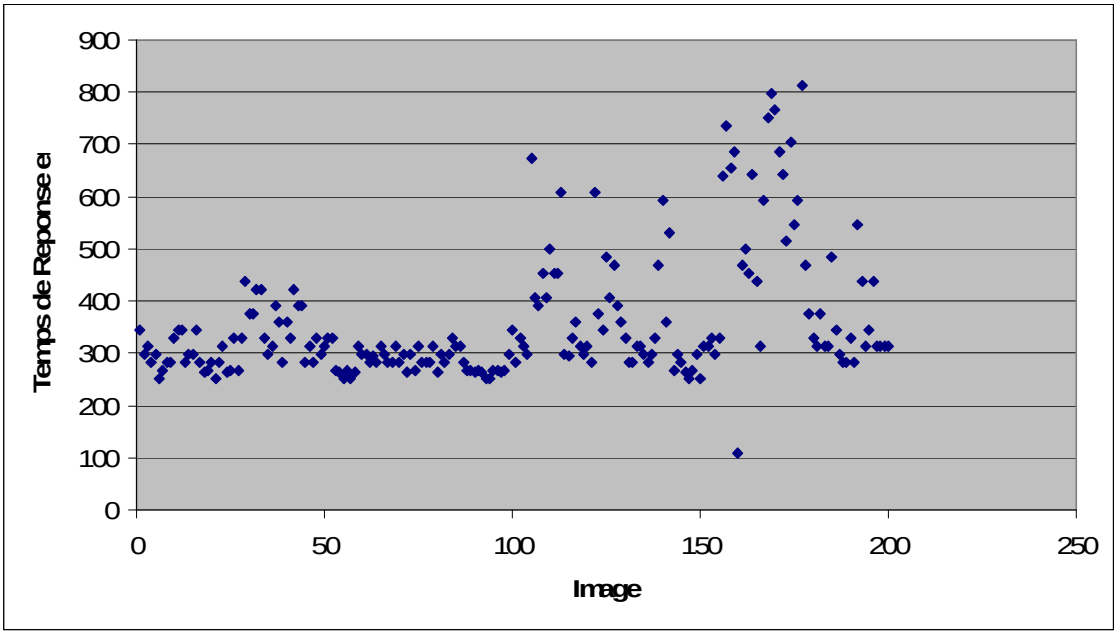


Fig 4.8 Distribution des temps de réponse sur la séquence

Première séquence

La deuxième mesure portait sur la même séquence, en changeant seulement la taille de la balle (environ ¼ de la balle initiale) le temps de réponse était de 284.45 ms:

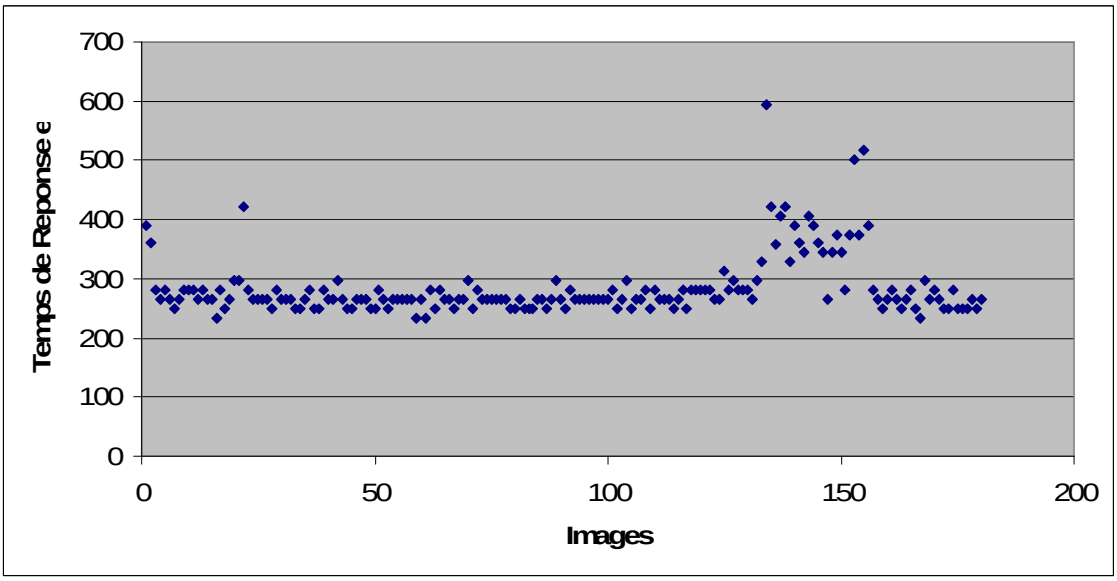


Fig4.9. Distribution des temps de réponse sur la séquence.

Première séquence avec une petite balle.

La troisième prise de mesures s'est effectuée sur la deuxième séquence (celle des trois appareils), les images utilisées sont de type couleur RGB en 24bit/pixel et de taille 300X300 pixels, les trois appareils sont de faible granularité sur la majorité des images de la séquence (de petite taille), le temps de réponse moyen par image est de 287.505 ms.

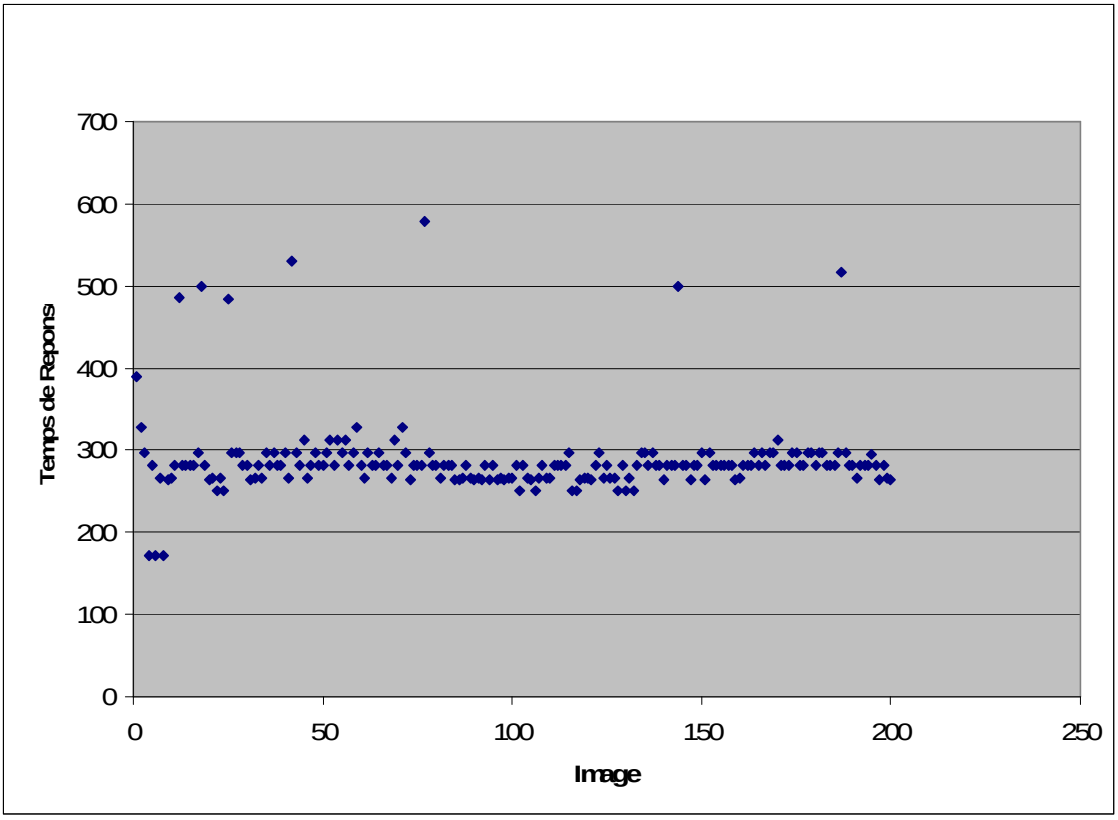


Fig.4.10 Distribution des temps de réponse sur la séquence.
Deuxième séquence.

IV.5 Conclusion.

L'objectif escompté de ce travail est l'implantation d'un SMA pour le suivi d'objet dans la séquence d'images. Le système qu'on a mis en œuvre satisfait du point de vue fonctionnement a notre objectif (une correcte détection, un suivi robuste aux occultations, un temps de réponse relativement acceptable).

La conclusion qu'on a tiré de l'expérimentation et de la mesure des performances est que le temps de réponse du système est tributaire de la taille des objets à détecter, du type et de la taille des images traitées et d'un degré moins du nombre d'objets à détecter. Ceci s'explique par la gourmandise des algorithmes de segmentation en temps machine. Et par conséquent, a notre avis, il est impératif de rechercher des solutions (éventuellement matérielles) adéquates pour optimiser le processus de segmentation des images.

CONCLUSION.

Bilan et perspectives.

Par ce travail, nous avons essayé de proposer une architecture multiagents pour le suivi d'objet dans les séquences d'images, l'expérimentation nous a permis de confirmer la validité d'une telle approche.

- Approche Agent VS approche classiques.

Les avantages de l'approche agent pour le suivi peuvent être résume en:

1- **Une meilleure structuration**: il est clair que le paradigme agent (séparation en entités autonomes) offre une meilleure structuration systémique, le cycle de maintenance et l'évolution d'un système de suivi est extrêmement flexible dans la mesure ou le changement d'un comportement d'un agent peut se faire sans affecter la logique globale du système.

2- **Flexibilité:** Au niveau du comportement interne de l'agent, la séparation des différentes taches de suivi (détection, appariement, prise de décision sur l'état des objets) en plusieurs comportements dynamiques distincts permet la possibilité de l'intégration aisée de plusieurs techniques de suivi.

3- **Exécution distribuée:** L'architecture qu'on a développé peut fonctionner naturellement sur un ensemble de machines (PC, PAD, téléphone mobile ...etc.) interconnectée. Ce qui permet d'envisager des applications temps réel.

L'inconvénient majeur d'une approche agent par rapport a une approche classique, peut être la perte de temps induite par l'interaction inter-agent (échange de messages dans notre cas). Cet inconvénient est compensé dans le cas d'une exécution parallèle ou distribuée

- Perspectives.

Une première extension envisageable (aisément) de notre système est le traitement du cas de plusieurs cameras mobiles dans un espace 3D, dans ce cas on aura plusieurs agents serveurs (un par camera) et éventuellement un ou plusieurs Agents détecteurs, le seul point a spécifier sera la coordination entre les différents agents détecteurs.

Une autre extension peut consister à la reconnaissance des objets détectés (au lieu de fournir uniquement des coordonnées d'une zone en mouvement). Ceci permettra de générer automatiquement des rapports portant sur la sémantique de la vidéo.

Bibliographie

[Aus75]	John Langshaw Austin. How to Do Things with Words. Harvard Univ. 2nd edition.1975
[BBCP05]	Fabio Bellifemine, Federico Bergenti, Giovanni Caire, and Agostino Poggi. JADE - A Java Agent Development Framework, chapter 5. Bordini et al. [BDDFS05], 2005.
[BF95]	M. Barbuceanu and M. S. Fox. Cool : A language for describing coordination in multiagent systems. (ICMAS-95), pages 17–24, San Francisco, USA, 1995.
[BF97]	Mihai Barbuceanu and Mark S. Fox. Integrating communicative action, conversations and decision theory to coordinate agents. Proceedings of the First International Conference on Autonomous Agents (Agents'97), CA, USA, 1997. ACM Press.
[BFVW03]	Rafael H. Bordini, Michael Fisher, Willem Visser, and Michael Wooldridge. Verifiable multi-agent programs. In PROMAS, 2003.
[BHV05]	R. H. Bordini, J.F. Hübner, and R. Vieira. Jason and the golden fleece of agent-oriented programming, 2005
[Bra93]	Brassac. Théorie des actes de langage et intelligence. PRC-IA, Montpellier, 1993
[Che95]	B Giai Checca"Analyse du mouvement et de la structure au sein des séquences d'images monocculaires", 1995
[Col01]	R.W. Collier. Agent Factory : A Framework for the Engineering of Agent-Oriented Applications. PhD thesis, University College Dublin, 2001.
[CY03]	Robert T.Collins, Yanxi Liu 2003: On-Line selection of discriminative tracking features, IEEE, 9 Conference on Computer Vision, 2003.
[DE94]	W. H. E. Davies and P. Edwards. Agent-K : An Integration of AOP and KQML. CIKM'94 Workshop on Intelligent Agents, USA, 1994.
[Dev06]	B. Deveze, Etat de l'art, Conception et Implementation d'agents intelligents, 2006.
[dKLW97]	Mark d'Inverno, David Kinny, Michael Luck, and Michael Wooldridge. A formal specification of dMARS. In Agent Theories, Architectures, and Languages, 1997.
[Don01]	Stéphane Donikian. Hpts : a behaviour modelling language for autonomous agents. In AGENTS '01 : Proceedings of the fifth international conference on Autonomous agents, USA, 2001. ACM Press
[Dro93]	Alexis Drogoul "De la simulation multi-agents à la résolution collective de problèmes" frflag2 Drogoul A. PhD Thesis, University Paris 6, November 1993.
[DZ05]	Jürgen Dix and Yingqian Zhang. Impact : A multi-agent framework with declarative semantics. Springer, 2005.
[ESP99]	T. Eiter, V. Subrahmanian, and G. Pick. Heterogeneous active agents i : Semantics, 1999.
[Fer92]	I. A. Ferguson. TouringMachines : An Architecture for Dynamic, Rational, Mobile Agents. PhD thesis, UK, 1992.
[Fer95]	Jacques Ferber. Les systèmes multi-agents, vers une intelligence collective. InterEditions, Paris (France), 1995.
[FIPA98]	Foundation for Intelligent Physical Agents. http://www.fipa.org.

[Fis93]	Michael Fisher. Concurrent METATEM - a language for modelling reactive systems. In Parallel Architectures and Languages Europe, 1993
[Fis94]	M. Fisher. A survey of concurrent METATEM – the language and its applications. Springer : Germany, 1994.
[GBV98]	B.Giai-Cecca, P.Bouthemey, T.Vieville:,"Detection of moving object ", IRISA 1998.
[Geo88]	GeorGeff "A theory of Action for MultiAgent Planning;" 1988.
[Hew77]	Hewitt « Viewing control structures as patterns of passing messages », *Artificial Intelligence* , 1977
[HM95]	Adu Horaud et Olivir Monga :" Vision Artificielle , Outils Fondamentaux ", Edition Hermes 1995.
[IGR92]	Francois F. Ingrand, Michael P. Georgeff, and Anand S. Rao. An architecture for real-time
[JP05]	Bernhard Jung & Paolo Petta "Agent Encapsulation in a cognitive vision MAS" Austrian Research Institute for Artificial Intelligence (OFAI) & Dept. of Med. Cybernetics and AI, Centre for Brain Research, Med. Univ. of Vienna, 2005.
[JSW98]	Jennings, Sycara, Wooldridge, Roadmap of Agent Research and Development. 1998.
[MC95]	Frank G. McCabe and Keith L. Clark. April – agent process interaction language.
[PBL05]	Alexander Pokahr, Lars Braubach, and Winfried Lamersdorf. JADEX - A BDI Reasoning Engine, 2005
[Pro02]	Conversation Management in MAS. Edition Hermès 2002.
[Rao96]	Anand S. Rao. AgentSpeak(L) : BDI agents speak out in a logical computable language. The Netherlands, 1996.
[RK96]	S. J. Rosenschein and L. P. Kaelbling. A situated view of representation and control. Computational Theories of Interaction and Agency, MIT 1996.
[RMP96]	Michael Rosinus, Jörg P. Müller, and Markus Pischel. An agent specification language, 1996.
[RN03]	Stuart Russell and Peter Norvig. Artificial Intelligence : A Modern Approach. Prentice-Hall, 2003.
[Ros96]	Michael Rosinus. Aladin - a language for designing interrap agents, 1996.
[Sho93]	Yoav Shoham. Agent-oriented programming. Artif. Intell., 1993.
[Sic95]	Sichman. On Social Reasoning in Multi-Agent Systems. PRC-IA 1995.
[Tho95]	S. Rebecca Thomas. The placa agent programming language. In ECAI-94 : Springer 1995
[UM02]	Norimichi Ukita Takashi Matsuyama "Incremental Observable- Area Modeling for Cooperative Tracking" Department of Intelligence Science and Technology Kyoto University 2000.
[Wag96]	G. Wagner. Viva knowledge-based agent programming, 1996.
[Wei99]	Gerhard Weiss, editor. Multiagent systems : a modern approach to distributed artificial intelligence. MIT Press, 1999.
[Win05]	Michael Winikoff. JACK Intelligent Agents : an Industrial Strength Platform, 2005.
[WJ95]	Michael Wooldridge and N. R. Jennings. Intelligent agents Theory and practice, 1995.
[Woo02]	Michael Wooldridge. Introduction to MultiAgent Systems. 2002.