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

UNIVESITE MOHAMED KHEIDER BISKRA
FACULTE DES SCIENCES ET SCIENCES DE L'INGENIEUR
DEPARTEMENT D'ELECTRONIQUE

MEMOIRE

Présenté pour l'obtention de diplôme de Magistère en

ELECTRONIQUE

Option

Architecture Des Systèmes

THEME

DETECTION D'OBJETS MOBILES PAR LES MODELES DE MARKOV CACHES (*MMCs*)

PAR

AOURAGH SALIMA

Soutenu devant le jury

Président	**: N.E**	**SENGOUGA**	**Prof.**	**Univ de Biskra**
Rapporteur	**: A.A**	**DEBILOU**	**C.C.**	**Univ de Biskra**
Examinateurs	**: M**	**BEDDA**	**Prof.**	**Univ de Annaba**
	Z.E	**BAARIR**	**C.C.**	**Univ de Biskra**

2005/2006

Introduction Générale

1 : Introduction

Ce travail présenté ici, s'inscrit dans le cadre général d'analyse du mouvement. On s'intéresse plus précisément aux applications dédiées à la surveillance.

Il est élaboré au sein du laboratoire de recherche « *LESIA* ; Les systèmes expert, l'imagerie et leurs applications dans l'ingénierie », dont les travaux de recherche sont centrés sur les activités traitant l'image fixe tel que la segmentation d'images à résonance magnétique « *IRM ;* Images à Résonance Magnétique. Aussi, on s'intéresse à l'aspect mobile de l'image ; autrement dit les séquences d'images et leur applications et traitement, tel que la compression et récemment la détection d'objet mobile.

Une autre motivation de notre groupe de recherche est l'implantation de ces algorithmes sur des processeurs spécialisés. Cela constitue le coté pratique de ce laboratoire.

Actuellement, le traitement vidéo est devenu un outil indispensable pour de nombreuses applications précisément celles liées à la vision par ordinateur et la robotique. Le traitement basé sur la notion dite 'vision' remplace de plus en plus des techniques, longtemps utilisées. A titre d'exemples, citons les travaux liés à l'analyse routière où le traitement vidéo remplace l'ancienne technique connue par la boucle inductive. Ces détecteurs vidéo sont adoptés à

cause de leur installation, opération, et maintenance simples [18], d'une part. D'autre part, ils permettent d'observer, surveiller et d'analyser des champs larges.

Une séquence vidéo est plus riche en informations visuelles qu'une image fixe. Cela se justifie par son habilité de « *capturer* » le « *mouvement* » [27]. Tandis que l'image fixe donne un flash ou '*snapshot*' de la scène. La séquence enregistre donc, la dynamique dans cette dernière. Cette dynamique (mouvement) présente un support important pour la vision humaine. Elle permet de détecter les objets mobiles dans la scène même s'ils sont inaperçus lorsqu'ils sont fixes.

La détection est également importante pour le traitement vidéo, aussi que pour la compression. Elle fournie les informations concernant la relation spatio-temporelle entre les objets dans l'image. Ces informations peuvent être exploitées dans des applications diverses tel que la télésurveillance, le suivi d'un objet bien défini, le traitement routier, ….etc. Ainsi, les propriétés de l'image (intensité ou couleur, à titre d'exemple) présentent une corrélation élevée dans le sens du mouvement. C'est-à-dire, elles ne changent pas si elles sont suivies. Cette caractéristique est exploitée pour enlever les redondances, cas de la compression, à noté comme exemple [27].

L'objectif de notre travail est de détecter l'objet mobile en identifiant les points de l'image comme étant mobiles ou fixes, en vue d'une surveillance. Dans ce contexte, on considère le changement d'intensité dans les images planes (*2D*), connu par *« mouvement apparent* ». Cela revient à classer les pixels en deux catégories : ceux fixes, appartenant au '*Background'*, et ceux faisant partie d'un objet mobile, connus par '*Forground'*.

Ainsi, la phase « *Détection »* est un constituant indispensable dans une chaîne de traitement de mouvement. Cependant, les méthodes acquises pour celle-ci présentent une diversité croissante; allant du seuillage, passant par la modélisation à base de différentes distributions et finissant par les Modèles de Markov Cachés *MMCs*.

Ces derniers sont utilisés dans de vastes applications ; comme la reconnaissance des gestes, la modélisation d'illumination et la détection, où ces modèles offrent une discrimination '*Forground-Background'* souple. De plus, une extension, en vue de la détection des ombres est possible. Notons seulement que la détection est référée comme étant la partie de traitement de bas niveau, mais appartenant à un processus global dit de haut niveau, et traitant une tache bien définie : identification du trajectoire d'un objet mobile ou reconnaissance de gestes ; cités à titre d'exemple.

Le travail réalisé au cours de ce mémoire concerne l'étude de la détection d'un objet mobile dans une séquence d'images, basée sur une approche Markovienne, en particulier par les chaînes de Markov cachées. Nous exploitons ces modèles afin de mettre en évidence leur performance dans ce domaine.

L'implantation de ces algorithmes de détection exige des dispositifs de traitement performants, dans le but d'une exécution en temps réel. Par conséquent, des processeurs spécialisés '*DSPs*' de '*TEXAS INSTRUMENTS*' appartenant à la famille des '*TMS320C6000*' et adaptés aux traitements d'images, peuvent être choisis pour représenter la partie traitement dans la chaîne complète de détection.

2 : Organisation du mémoire

Ce manuscrit est constitué de cinq chapitres organisés comme suit :

- Après une introduction situant le problème de la détection d'objet mobile, le premier chapitre donne un bref état de l'art concernant ce thème, en présentant une vue générale sur les méthodes utilisées. nous montrons la déférence entre ces procédures. Notons que celle-ci est dictée par le but souhaité des applications dont la détection présente une étape préliminaire. Par la suite, la problématique dans sa généralité, est présentée, et la démarche globale de traitement est définie. Ainsi, les Modèles de Markov Cachés sont adoptés, et constituent alors le support théorique de notre application.
- Le second chapitre donne une étude théorique de ces modèles. Il commence par introduire les chaînes de Markov, monte par la suite aux chaînes de Markov cachées, dans un objectif final de présenter les Modèles de Markov Cachés *MMCs*. Par conséquent, leurs éléments sont définis, et leurs problèmes associés sont présentés. Le chapitre se termine par une énumération assez complète des solutions usuelles et disponibles pour ces derniers.
- A cause de la complexité d'évaluation des probabilités associées à ce modèle *MMC*, le chapitre troisième présente les transformations utilisées, ainsi que les algorithmes existants, et dédiés soit à l'estimation des paramètres du modèle, soit à l'estimation des états cachées, ou classification. L'application de ces

procédures est montrée par des exemples divers de segmentation d'images, situés à la fin de ce chapitre.

- La détection d'objet mobile, l'estimation des paramètres à base d'un *MMC*, ainsi que la définition et choix d'algorithmes à implanter sur la chaîne de traitement sont l'objectif fixé pour ce quatrième chapitre. Pour la détection, on parle du mode des marginales a posteriori *MPM*, utilisé au niveau de l'estimation des informations cachées. Quant au second, celui de l'estimation des paramètres, on choisira l'algorithme itératif, se basant sur l'espérance conditionnelle, et connu par l'algorithme '*ICE ;* '*Iterated Conditional estimation'*. Terminant ce chapitre par une mise en évidence de l'association de ces deux algorithmes, montrée par des exemples de simulation de la détection d'objet mobile, qui sont divers, riches et réalisés sous *MATLAB*, ainsi qu'un certain nombre de traitement complémentaires, dans l'objectif de valider l'algorithme choisi.
- Le cinquième chapitre développe le dispositif expérimental constituant la chaîne de traitement d'images, son installation et son utilisation.

Ce manuscrit s'achève par une conclusion générale résumant la démarche présentée, suivie de quelques perspectives et extensions possibles de ce travail

Etat de l'art

1.1 : Introduction

Nous présentons dans ce premier chapitre, quelques éléments d'état de l'art couvrant plus au moins, la procédure de détection. Ainsi, nous essayons de toucher aux déférents constituants de cette dernière, à savoir la méthode utilisée, le support de calcul considéré et autres.

1.2 : Détection

La détection des régions de changement dans des images d'une même scène ; prises à des instants déférents, a un intérêt essentiel dans de nombreuses applications appartenant à des disciplines diverses [23].

Le spectre de son application est large : on peut citer la vidéo-surveillance, l'interaction homme-machine, le guidage des robots par la vision et le diagnostique médical.

La diversité des applications a permet l'élaboration d'un grand nombre d'algorithmes. Ces derniers varient d'une part, selon le support de calcul utilisé : intensité ou couleur, pixel ou région …,. D'autre part, ils peuvent être d'origine déterministe ou stochastique. Aussi, on

peut choisir entre déférentes méthodes de comparaison : soit entre trames, soit par rapport à une image référence, ou même une double comparaison. Autrement dit, l'environnement à traiter, le but à atteindre et la performance espérée, dictent un choix approprié d'algorithmes et de techniques.

Le problème en question, est comme suit : On dispose d'une séquence d'images acquise avec une caméra stationnaire, et on souhaite identifier l'ensemble des pixels appartenant à l'objet mobile. Ces pixels là font, toujours partie de l'ensemble total de ceux ayant subis un changement entre l'image actuelle et celles qui la précèdent. Ces derniers constituent ce qu'on appelle « *Masque* », ou *'Change Mask'*.

Or ce masque peut être le résultat d'une combinaison de certains facteurs. Citons, à titre d'exemple :

- Mouvement de l'objet par rapport à l'arrière plan '*Background '*.
- Changement de la forme de l'objet.
- Changement d'apparence de l'objet.

Un point essentiel est que ce masque doit être signifiant. C'est-à-dire, il ne doit pas contenir des formes de changement nuisantes, telles que :

- ✓ Le mouvement de la caméra.
- ✓ Le bruit du capteur et d'acquisition.
- ✓ Variation d'illumination.

Notons tout de même que l'adjectif « *signifiant* », varie selon l'application et le problème traité. Aussi, rappelons que dans cet état de l'art, on s'intéresse guerre à l'application traitée, mais seulement à la méthode sélectionnée au niveau de la « *Détection* ». Malgrés que l'application délimite plus au moins, la méthode de détection choisie, ainsi que la performance attendue.

La détection donc, est une étape préliminaire dans les applications liées à la vision. Dans la littérature, la notion « *Détection* » est parfois exprimée par « *Soustraction de l'arrière plan* » ou *'Background substraction'*. Cela revient à effectuer une comparaison de l'image d'entrée à une autre référence appelée '*Background'*. Ainsi, une partie d'algorithmes de

détection exploitent cette notion. D'autres se basent sur une différence temporelle. Les premiers sont bénéfiques, lorsque le *Background* est connue d'avance, et ne change pas significativement dans le temps. Si des changements sont prévus, une étape d'adaptation et maintenance est nécessaire. Elle peut être de plus en plus sophistiquée pour répondre aux déférents changements (conditions d'illumination, mouvements des branches d'une arbre, …). Tandis qu'une différence temporelle peut s'adapter rapidement aux changements d'illumination par exemple.

En résumé, trois types de différence peuvent être utilisés :

- Une première catégorie ; celle qui détecte les changements par rapport à un '*Background'*, en comparant l'image actuelle à une image référence vide (sans objet mobile), et acquise précédemment. Généralement, ces systèmes demandent une adaptation [3, 14, 16] [21, 46] (surtout dans des séquences naturelles).

En effet, les deux expressions : '*Background substraction'* et '*Background modeling'*, sont interchangeables. Or réellement, ils sont distincts.

Modéliser le *Background :* est le processus de création d'un modèle, dans lequel, on intègre les changements dynamiques qu'il subi, en but de sa maintenance dans le temps. Alors que la soustraction du '*Background*' définisse le processus de comparaison de l'image actuelle au modèle du '*Background*', déjà construit.

Les deux expressions sont généralement utilisées en conjoint [4], comme le montre la figure **Fig (1.1)** suivante:

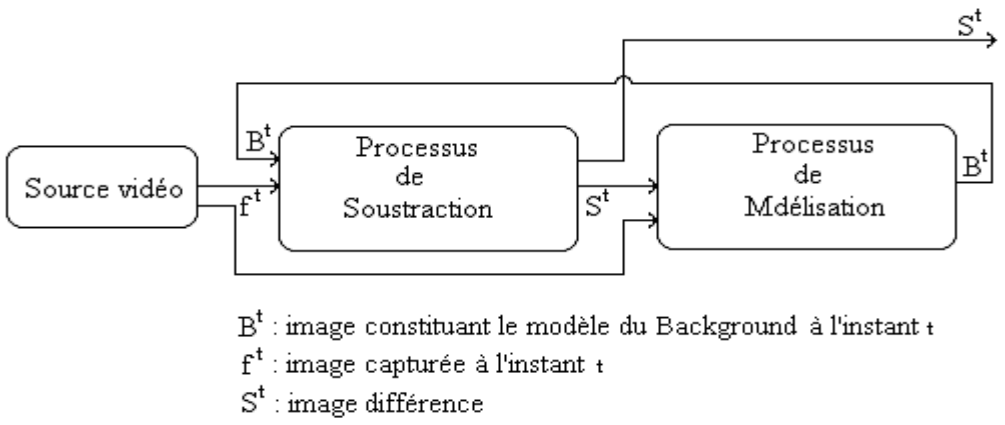


Fig (1.1) : ***Relation entre soustraction et modélisation du'Background'***

La forme la plus simple de cette image de référence, est la moyenne temporelle [16]. Mais, celle-la a besoin d'une étape d'apprentissage effectuée sur une scène vide. En addition, elle ne répond pas aux changements d'illumination. Une adaptation continue du modèle est conseillée.

Généralement, la modélisation du '*Background*' opère au niveau pixel. Elle utilise parfois des distributions [1,16]. L'intensité du pixel est alors, modélisée par un mélange de distributions Gaussiennes [1]. Leurs paramètres et leurs nombres sont mis à jours, à l'aide d'équations récursives. Tandis que Rowden et ses collègues dans [16], présentent un algorithme du même genre, ayant en plus la possibilité de séparer entre l'objet mobile et son ombre. Ils exploitent l'information chrominance.

Donc, un algorithme appartenant à ce groupe doit être capable de :

- ✓ Déterminer le nombre de distributions.
- ✓ Estimer les paramètres de chacune.
- ✓ Mettre à jours le '*Background*'.

- Une deuxième catégorie : celle de la différence temporelle, nécessitant au moins deux images [14, 20, 24], ou parfois trois [7], et donc nommée par « différence double ». Remarquons dans ce cas, qu'un changement temporel détecté ne détermine pas exactement l'objet mobile. La présence de ce dernier cause l'apparition de trois régions :

- ➢ Région du '*Background*' découvert.
- ➢ Région du '*Background*' couvert.
- ➢ Région de glissement de l'objet sur lui-même. Celle-ci est difficile à distinguer si l'objet est de nature uniforme. Cela exige une étape supplémentaire nommée localisation de cet objet mobile sur l'image plane. Une des solutions disponibles est l'approche markovienne ; avec les champs aléatoires de Markov '*MRF*[1]'. Cette méthode est décrite dans [25]. Des distributions gaussiennes sont utilisées au niveau de la détection, et de la localisation. A chaque niveau, une fonction de coût est construite, puis minimisée. La classification est faite par le critère *MAP « Maximum A Posteriori »*. Il est de même dans [13], sauf qu'au premier stade, les gaussiennes sont remplacées par des distributions de Laplace.

[1] : Markov Random Feilds

Les méthodes se défèrent non seulement dans la comparaison, mais par le support de calcul. L'un considère le pixel [11, 8]. L'autre opère sur des régions [47]. Un dernier s'intéresse aux contours [9]. Duric et ses collègues [6], intègrent la notion de « *contour* » à côté de la « *couleur* » pour modéliser le '*Background*'. Le modèle global est divisé aux sous modèles appropriés aux trois canaux couleurs. Le changement dans l'un des canaux représente un indice de mouvement. Le calcul du contour est élaboré via l'opérateur de *Sobel*. Cette stratégie est appliquée pour détecter des individus dans des scènes de l'extérieur.

Il faut signaler que la couleur est fréquemment utilisée pour distinguer entre l'objet et son ombre. La détection fausse (comme étant objet) de ce dernier, dans certaines applications, tel que la reconnaissance, amène à un traitement supplémentaire. Une technique connue est la décomposition de la couleur en deux composantes : luminance et chrominance. Une autre association couleur-contour se trouve dans [9].

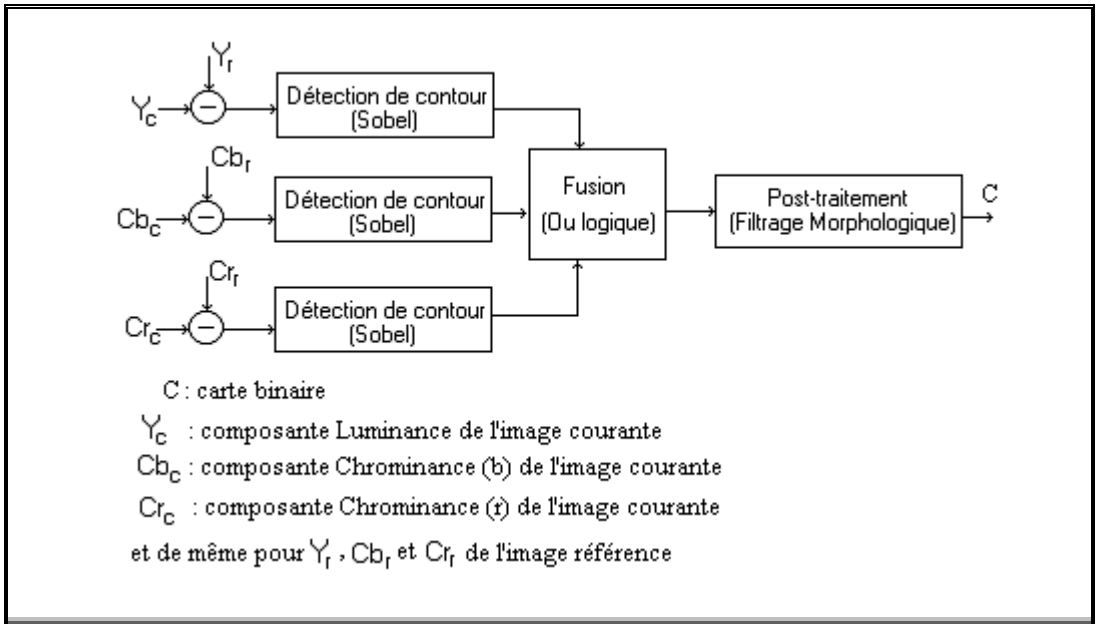


Fig (1.2): ***Processus de détection du mouvement dans [9]***

Les premiers détecteurs à base de contours, effectuent un calcul sur l'image référence et un autre sur l'image considérée. Puis dressent une comparaison entre les deux cartes contour par opérateurs logiques (un exemple : le ou exclusif). Une contrainte majeure est sa sensibilité envers les changements d'illumination. Une amélioration est de faire une soustraction en premier, suivie de la détection de contours. Or, les contours à base d'intensité sont de nature ouverte ; d'où un '*post-processing*', en vu de leur fermeture. Ainsi dans [9], les

contours sont définis par l'opérateur de *Sobel* au niveau de chaque canal (Y, Cb, Cr), suivi d'une fusion et d'un post-traitement. Cette finalisation est constituée d'un filtrage morphologique, qui a pour objectif de remplir les contours, donnant ainsi la carte binaire de la détection. Cette idée est illustrée par la figure précédente **Fig (1.2)**.

Une troisième catégorie, parfois utilisée, est celle qualifiée par hybride. Ici, on combine entre les deux types de différence. L'idée est exploitée dans [15], dans l'objectif de déterminer un seuil automatique. Cela est montrée par la figure suivante **Fig (1.3)** .

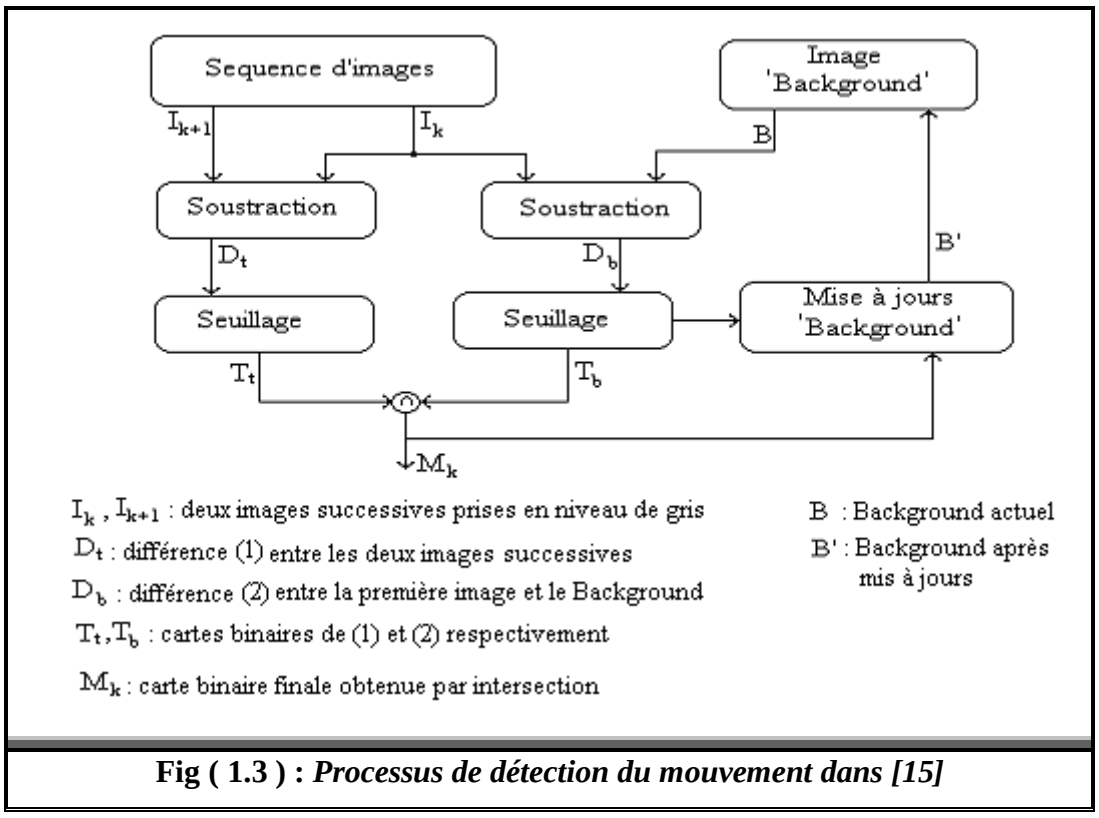


Fig (1.3) : ***Processus de détection du mouvement dans [15]***

L'étape de classification des pixels, comme mobiles ou '*Background*' s'effectue, elle aussi selon déférentes techniques. La plus ancienne ; celle d'origine intuitif est le seuillage ; parfois appelé « *binarisation* ». La valeur de ce seuil est critique. Elle dépend évidement des bruits, des changements d'illumination, aussi bien de la différence naturelle à détecter (objet mobile). Des changements non importants seront détectés si la barre est trop basse. Ceux primordiales seront négligés si le seuil est trop élevé. Donc, sa sélection appropriée pour chaque pixel est indispensable.

Dans les méthodes stochastiques basées sur les champs de Markov ; dont lesquelles l'information voisinage est introduite, ou celles basées sur les champs de Gibbs, on cherche à minimiser une fonction de coût, par l'intermédiaire d'une fonction d'énergie maximisée [4,13, 24]. Généralement, on choisi entre les algorithmes itératifs de relaxation déterministe, tel que l'algorithme *HCF* pour '*Highest Confidence First* ', et celui nommé *ICM* pour '*Iterated Conditional Mode'*, ou ceux de relaxation stochastiques tel que le « *recuit simulé* ». Ce dernier est coûteux en temps de calcul.

Un point à ne pas oublier est que, avant l'étape de classification, les modèles construits nécessitent toujours une estimation de paramètres ; paramètres des gaussiennes par exemple ou de régularisation pour les *MRFs*. Le critère de maximum de vraisemblance *MV,* est utilisé dans [13]. Tandis que Bowden dans [16] utilise en ligne, l'algorithme *EM* , à dire '*expectation maximisation'*.

Notons qu'un algorithme *EM* en ligne représente une amélioration d'un autre hors-ligne ; dans ce dernier, on a besoin d'acquérir d'avance, une séquence dite d'apprentissage ; '*training'*. Les méthodes récentes utilisent parfois l'algorithme *ICE ; 'Iterative Conditional Estimation'*. Il est basé sur une moyenne conditionnelle et à nature statistique.

Un dernier modèle, récemment introduit (2000), modélisant les régions '*Background'* et '*Forground*' utilise les Modèles de Markov cachés *MMCs* de premier ordre et à une seule dimension. Ces modèles sont basés sur les Chaînes de Markov Cachées.

Son principe est d'estimer les états cachés à travers les valeurs observées, dans un cadre probabiliste [2, 11, 26]. Cette méthode est essentiellement composée de deux étapes. Une pour l'apprentissage dans laquelle, l'algorithme *EM* est généralement utilisé. Une autre pour la classification, ou on maximise la probabilité choisie, en vue d'un choix d'états optimal, selon un critère choisi. Un critère de « Maximum A Posteriori *MAP »*, ou *MPM*, à dire « Mode de la Marginale à Posteriori » peut être utilisé. Un algorithme associé aux *MMCs* est celui de '*Viterbi'*, utilisé dans [13].

Un modèle *MMC* est capable de s'adapter aux changements globaux du '*Background'* [4], tel que les changements d'illumination. Il est aussi extensible. En ajoutant un troisième état, il devient capable de détecter les ombres [11, 26].

En fin, les *MMCs* offrent au modèle, la possibilité de changer de structure ; cas décrit dans [11], pour s'adapter à la nature non stationnaire des phénomènes réels. Des algorithmes

dites de fusion d'états, existent et répondent aux problèmes liés à la modification de cette topologie.

Signalons à la fin, que tous les algorithmes précédents tentent à s'exécuter en temps réel, dans le but de répondre aux exigences naturelles de traitement.

1.3 : Problématique

Une séquence d'images est un support riche d'informations. Son traitement, dans le but d'extraire l'information utile, nécessite énormément de calculs complexes.

Elle peut être traitée en tant que fonctions déterministes. Comme, elle peut être considérée comme une réalisation de processus stochastiques. Les outils mathématiques utilisés reposent sur la théorie des systèmes linéaires, des mathématiques discrètes et des processus stochastiques.

L'extraction d'objet mobile à partir de scènes dynamiques est l'étape préliminaire nécessaire dans de nombreuses applications, telles que la compression, la surveillance, …,etc. Un taux de compression élevé est le résultat d'une bonne détection. Déférentes approches ont été proposées au niveau de la détection. Citons à titre d'exemple, le seuillage, la modélisation par déférentes densités, l'approche floue, …, etc.

Notre objectif est de développer un algorithme permettant la détection en temps réel d'un objet mobile dans une séquence d'images, dans le but d'améliorer les procédés de détection et de compression d'images. Comme support théorique, on s'appuie sur les modèles basés sur les « *Chaînes de Markov Cachées CMCs »*, appelés aussi *« Modèles de Markov cachés MMCs »*. Ces derniers permettent une utilisation souple et extensible.

Le traitement de l'information en temps réel, est devenu d'une exigence extrême. Pour cela des circuits performants spécifiques à l'imagerie sont introduits tel que : les *FPGAs*[2], et les *DSPs*[3]., en vue d'une implantation de ces algorithmes souple et en temps réel.

L'algorithme de détection ainsi choisi, si les conditions matérielles et logicielles sont favorables, sera par la suite implanté dans une chaîne de traitement d'images piloté par un *DSP* de la famille TMS320C6000 de TEXAS INSTRUMENTS.

[2] : Feild Programmable Gate Array
[3] : Digital Signal Processor

Les chaînes de Markov cachées

2.1 : Introduction

Les modèles de Markov cachées *MMCs* sont des modèles statistiques, riches et largement utilisés en traitement du signal. Ils sont développés par *Andrew Markov* (étudiant de *Chebyshev*), et ils sont premièrement, orientés vers des objectifs linguistiques dans des travaux de littérature Russe [29]. Ceux sont des outils efficaces en modélisation des données séquentielles ou '*time-series Data*' [31]. Utilisés par la suite dans des problèmes de reconnaissance de la parole par *Baker*, leur théorie de base est introduite par *Baum* et ses collègues à la fin des années soixante.

Actuellement, ces modèles sont, de plus en plus adoptés en reconnaissance automatique de la parole, pour l'analyse des séquences *ADN* [4], et dans des problèmes liés à l'écriture et le traitement de texte.

Aussi, leur utilisation en visionique est vaste. Ils sont implémentés en segmentation d'images, reconnaissance des visages, interprétation des gestes, ainsi que la modélisation de l'arrière plan et récemment en traitement vidéo.

[4] : l'acide nucléique appelé : Acide Désoxyribonucléique

Ce chapitre constitue le support théorique de base de notre étude. Nous allons donc, donner une vue générale de ces modèles, les éléments particuliers intervenant dans leur construction, les problèmes liés à leur utilisation et les solutions usuelles de ces derniers.

2.2 : Chaînes de Markov et extension aux chaînes cachées

2.2.1 : Chaîne de Markov

Supposons $X = (X_1 X_2 ... X_T)$ une séquence de variables aléatoires prenant leurs valeurs dans un ensemble fini $\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$ appelé l'espace d'état.

On définie un processus aléatoire, qui à chaque instant, visite l'un des K états possibles. A l'instant suivant, ce système (processus) change d'état, avec possibilité de rester à l'état actuel, suivant un ensemble de probabilités ; **Fig (2.1)** .

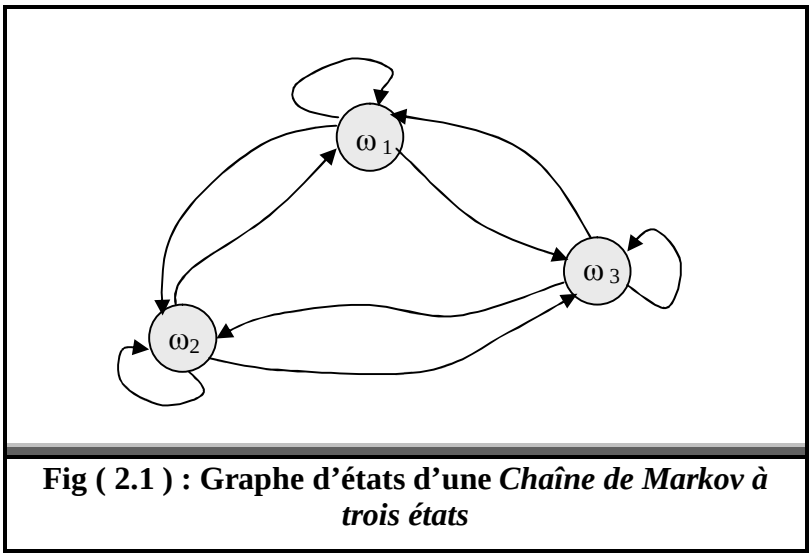


Fig (2.1) : Graphe d'états d'une *Chaîne de Markov à trois états*

Une description probabiliste complète du processus, nécessite la spécification de l'état actuel (à l'instant t), ainsi que tous les états précédents [29, 31].

Pour un processus Markovien du premier ordre, cette spécification est arrondie à l'état actuel et son précédent adjacent. Cela est produit par l'équation suivante :

$$P(X_t = \omega_j / X_{t-1} = \omega_i, ..., X_1 = \omega_1) = P(X_t = \omega_j / X_{t-1} = \omega_i) \quad (2.1)$$

Les probabilités conditionnelles $P(X_t = \omega_j / X_{t-1} = \omega_i)$ sont dites des probabilités de **Transition**, et sont notées a_{ij}^t . a_{ij}^t ne dépendra pas du temps si elle est la même à deux temps

déférents t et t'. Dans ce cas, les a_{ij}^{t} seront notées tout simplement les a_{ij} ; constituant ainsi une matrice carrée d'ordre K ; appelée **Matrice de Transition**, et nommée A. Signalons que K est le nombre d'états possibles, dans lesquels peut évoluer le processus. On écrit :

$$a_{ij}=P\left(X_t=\omega_j / X_{t-1}=\omega_i\right) \quad \text{(2.2)}$$

et la matrice de transition sera:

$$A=\left[a_{ij}\right]=\left[P\left(X_t=\omega_j / X_{t-1}=\omega_i\right)\right] \quad ,1\leq i,j\leq K \quad \text{(2.3)}$$

Les éléments a_{ij} vérifient les propriétés stochastiques suivantes :

$$\begin{aligned} &1/ \quad 0 \leq a_{ij} \leq 1 \qquad \forall i, j \\ &2/ \quad \sum_{j=1}^{K} a_{ij}=1 \qquad i=1,...,K \end{aligned} \quad \text{(2.4)}$$

Désignons par $P(t)$, le vecteur stochastique défini par $P\left(X_t=\omega_i\right) \quad avec: i=1,...,K$, ainsi $P(1)$ sera le vecteur des distributions initiales (à l'instant t=1). On note ce vecteur par π, tel que chacun de ses éléments est donné par:

$$\pi_i=P\left(X_1=\omega_i\right) \quad \text{(2.5)}$$

De même, les π_i ; éléments de π, vérifient les propriétés stochastiques précédentes :

$$\begin{aligned} &1/ \quad 0 \leq \pi_i \leq 1 \quad \forall i \\ &2/ \quad \sum_{i=1}^{K} \pi_i=1 \quad \forall i \end{aligned} \quad \text{(2.6)}$$

Notons donc, qu'une chaîne de Markov est entièrement définie par sa matrice de transition A et ses distributions initiales, données par le vecteur π.

Cela en effet, peut être résumée par la définition ci-après :

Définition :

Chaîne de Markov

Une ***chaîne de Markov*** est un processus stochastique $X = (X_t)_{t=1}^{T}$ tel que :

$\forall t, \quad P(X_t / X_{t-1i}, X_{t-2}..., X_1) = P(X_t / X_{t-1})$.

$\pi_i = P(X_1 = \omega_i)$

De plus, si $P(X_t = \omega_j / X_{t-1} = \omega_i) = a_{ij}, \forall t,$ si ces probabilités appelées ***probabilités de transition*** sont indépendantes de t, alors la chaîne est dite homogène.

2.2.2 : Graphe d'indépendance d'une chaîne de Markov

La chaîne de Markov appartient à la famille des signaux pouvant être représentés par des graphes ; où les cercles sont étiquetés par les états possibles, et les arcs désignent les transitions valables ente ces états. Le graphe d'indépendance d'une chaîne de Markov est donné par la **Fig (2.2)** :

Fig (2.2) : ***Graphe d'indépendance d'une chaîne de Markov***

La figure ci-dessus représente un processus Markovien observable. Ses sorties donnent directement les états, dont chacun est attaché à un évènement physique observable [31].

La probabilité jointe d'une séquence d'état $X = (X_1 X_2 ... X_T)$ est facilement calculée pour une chaîne de Markov. Elle est donnée par un produit de probabilités comme suit :

$$P(X_1, X_2, ..., X_T) = P(X_1) \cdot P(X_2 / X_1) \cdot P(X_3 / X_1, X_2) \cdot ... \cdot P(X_T / X_1, X_2, ..., X_{T-1}) .. \quad (2.7)$$

En intégrant la propriété de Markov donnée par la formule (2.1), l'équation (2.7) devient :

$$P(X_1, X_2, ..., X_T) = P(X_1) \cdot P(X_2 / X_1) \cdot P(X_3 / X_2) \cdot ... \cdot P(X_T / X_{T-1}) \ldots\ldots\ldots\ldots \quad (2.8)$$

Cette dernière est condensée donnant l'écriture suivante:

$$P(X_1, X_2, ..., X_T) = \pi_{X_1} \cdot \prod_{t=1}^{T-1} a_{X_t X_{t+1}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.9)$$

Nous citons un petit exemple qui met en valeur la notion de la Chaîne de Markov, et permet de mesurer la probabilité d'une séquence d'états bien définie. Remarquons que cette mesure de probabilité est évaluée par un simple produit.

Exemple

Utilisation de chaîne de Markov dans une prédiction météo :

Assumons que chaque jour, le temps est dans l'un des trois états suivants :

- Etat ω_1 : pluvieux.
- Etat ω_2 : nuageux.
- Etat ω_3 : ensoleillé.

On donne la matrice de transition A par:

$$A = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

et le vecteur initial π par : $\pi = [0\ 0\ 1]$.

La question posée est :

Sachant qu'aujourd'hui est ensoleillé, et avec un tel modèle, quelle est la probabilité que le temps suit, dans les sept (07) jours qui suivent, la séquence d'observation O donnée par :

$O = (\omega_3, \omega_3, \omega_3, \omega_1, \omega_1, \omega_3, \omega_2, \omega_3)$ pour $t = 1,2,...,8$.

Une telle probabilité est exprimée par $P(O/\text{mod}\grave{e}le)$ tel que :

$P(O/\text{mod}\grave{e}le) = P(\omega_3,\omega_3,\omega_3,\omega_1,\omega_1,\omega_3,\omega_2,\omega_3/\text{mod}\grave{e}le)$ avec $P(O/\text{mod}\grave{e}le)$ dénote la probabilité de la séquence d'observations O sachant ce modèle.

En utilisant la propriété de la chaîne de Markov marquée par (2.8), la formule précédente

$$\begin{aligned} P(O/\text{mod}\grave{e}le) &= P(\omega_3)\cdot P(\omega_3/\omega_3)\cdot P(\omega_3/\omega_3)\cdot P(\omega_1/\omega_3)\cdot P(\omega_1/\omega_1)\cdot P(\omega_3/\omega_1) \\ &\quad \cdot P(\omega_2/\omega_3)\cdot P(\omega_3/\omega_2) \end{aligned}$$

devient:

$$\begin{aligned} &= \pi_3 \cdot a_{33}\cdot a_{33}\cdot a_{31}\cdot a_{11}\cdot a_{13}\cdot a_{32}\cdot a_{23} \\ &= (1)\cdot(0.8)\cdot(0.8)\cdot(0.1)\cdot(0.4)\cdot(0.3)\cdot(0.1)\cdot(0.2) \\ &= 1.536\cdot 10^{-4} \end{aligned}$$

2.2.3 : Chaîne de Markov cachée

Maintenant, on ajoute des observations ; on considère que X est un processus caché, et on souhaite déterminer ses états par l'intermédiaire du vecteur d'observations Y .

Donc, un modèle de Markov caché; noté *MMC* est un processus stochastique double [31], avec :

- ✓ Un premier processus X caché, constitué d'un ensemble d'états connectés entre elles par des probabilités de transition.
- ✓ Un deuxième processus Y observable, constitué d'un ensemble de sorties; dites observations. Chaque sortie peut être émise par n'importe quelle état caché, conformément à une fonction densité de probabilité de sortie.

On peut annoncer la définition ci après :

Définition :

Chaîne de Markov cachée

Une ***chaîne de Markov cachée 'CMC'*** est un processus stochastique double (X,Y) [35], tel que :

- $X = (X_t)_{t=1}^{T}$, soit une chaîne de Markov homogène. Elle est définie par sa distribution initiale π et sa matrice de transition A.

- $Y = (Y_t)_{t=1}^{T}$, soit un processus observable et indépendant conditionnellement à X ; tel que : $P(Y/X) = \prod_t P(X_t/Y_t)$.

2.2.4 : Graphe d'indépendance d'une chaîne de Markov cachée

Graphiquement, on représente une chaîne de Markov cachée par la figure **Fig (2.3)** . On dit qu'on a une dépendance ponctuelle de *Y* relativement à X .

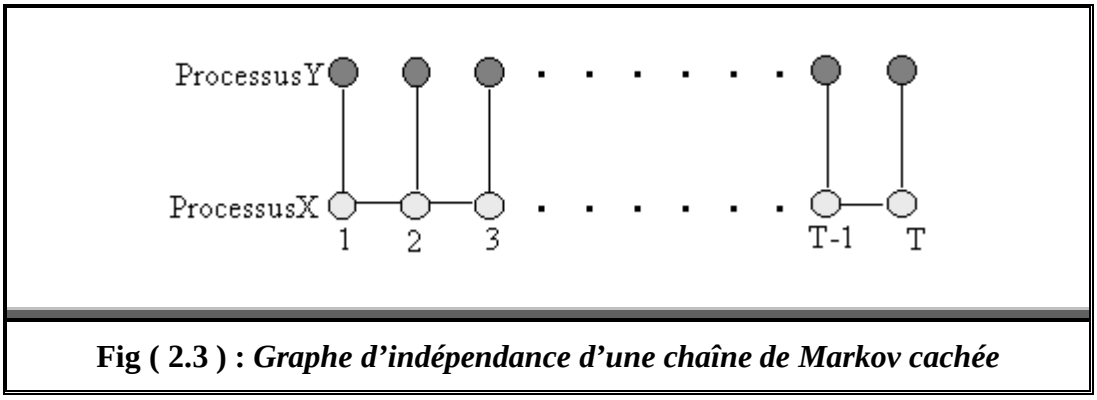


Fig (2.3) : ***Graphe d'indépendance d'une chaîne de Markov cachée***

2.3 : Lois d'observation

Plusieurs approches ont été successivement proposées dans le choix des lois d'observation rattachées aux états cachées du modèle. Si l'espace d'observation est déscret et fini, chaque distribution $b_j(.)$ est entièrement définie par la reconnaissance de la probabilité d'émission de chacun des points de l'espace. $\{y_1, y_2, ..., y_M\}$. La valeur de M, donne la taille de cet espace d'observation. L'ensemble des lois d'observation est $B = \{b_j(.)\}_{1 \le j \le K}$, et entièrement caractérisé par le vecteur $\{b_j(m.); 1 \le j \le K; 1 \le m \le M\}$.

Chaque paramètre $b_j(m)$ détermine la probabilité que le point y_m soit émis par la distribution $b_j(.)$, l'ensemble des probabilités $b_j(m)$ vérifiant, pour chaque état ω_j du modèle la contrainte :

$$\forall j : \quad \sum_{1 \le m \le M} b_j(m) = 1$$

Lorsque l'espace d'observation est continu, plusieurs choix sont possibles pour les fonctions densités de probabilité, et sont en général basés sur des gaussiennes. L'ensemble des lois d'observation $F = \{f_j(.)\}_{1 \le j \le K}$ est alors caractérisé par la connaissances des K densités de probabilité $f_1, f_2, ..., f_K$ associées aux K classes considérées.

2.4 : Eléments d'une chaîne de Markov cachée

A fin d'utiliser une chaîne de Markov cachée, il est indispensable de spécifier les caractéristiques ci-après. On parle d'éléments de cette chaîne :

- ✓ K : Nombre d'états dans le modèle. Chaque état a une signification physique déterminée, selon l'application prise. On dénote les états individuels par $\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$. L'état, à l'instant t est noté X_t.
- ✓ M : Nombre de symboles observés. Ils correspondent aux sorties physiques du système modélisé. On dénote les symboles individuels par $\{y_1, y_2, ..., y_M\}$. Celui, à l'instant t est noté Y_t. La structure, elle aussi, est imposée par le phénomène étudié.
- ✓ $A=[a_{ij}]$: matrice de transition, avec:

$$a_{ij} = P(X_t = \omega_j / X_{t-1} = \omega_i), \quad 1 \le i, j \le K .$$

- ✓ $\pi = [\pi_i]$: probabilités initiales de transition, avec :

$$\pi_i = P(X_1 = \omega_i), \quad 1 \le i \le K .$$

- ✓ $B = \{b_j(y_t)\}$: probabilités d'émission, tel que :

$b_j(y_t)$: est la probabilité d'émettre le symbole y_t à l'instant t, sachant que l'état à ce moment est $X_t = \omega_j$. On écrit :

$$b_j(y_t) = P(Y_t = y_t / X_t = \omega_j)$$

- ✓ T : Longueur de la séquence d'observations Y ; avec Y_t est le symbole observé à l'instant t.

Donc, en utilisant une notation compacte, un *MMC* est noté par $\lambda = (\pi, A, B)$ désignant les paramètres complets d'un modèle de Markov caché.

2.5 : Types principaux des modèles *MMCs*

A toute chaîne de Markov est associé un graphe d'états, représentatif des possibilités d'évolution séquentielle du processus modélisé. Sa topologie dépend directement des positions relatives des zéros dans la matrice *A* de transition. La structure d'un *MMC* diffère donc, selon le phénomène à étudier. Parmi celles les plus utilisées, on cite deux types importants : « *MMC ergodique* » et « *MMC gauche-droit* » [31,33], tout deux présentés par la figure : **Fig (2.4)** .

Celui ergodique de **Fig (2.4.a)**, est un cas spécifique ; où le modèle est dit complètement connecté, et toutes les possibilités d'évolution sont permises a priori. Autrement dit, tous les a_{ij} sont positifs, et la matrice *A* est pleine. Cela permet d'atteindre n'importe quel état en passant par n'importe quel chemin.

Tandis que un *MMC* à matrice *A* triangulaire supérieure appelé « *gauche-droit* » ou '*Left-Right*' est illustré dans **Fig (2.4.b)**. Celui-ci est particulièrement adapté à la modélisation d'évolution temporelle de processus (parole, gestes). Ce type se dote de la propriété d'accroissance de l'indice d'état relativement au temps (pas de retour arrière). Il est notamment choisi pour modéliser les signaux dont les caractéristiques changent avec le temps. La propriété fondamentale de ce genre est :

$$a_{ij} = 0\,,\ \text{si}\ \ i < j, \qquad \text{et}\ \ \pi_i = \begin{cases} 0\,, & i \neq 1 \\ 1\,, & i = 1 \end{cases}$$

Il revient de souligner que le graphe d'états d'un *MMC* n'est pas à confondre avec le graphe d'indépendance, lequel représente le modèle « déroulé » dans le temps, en soulignant graphiquement la dépendance ponctuelle des observations relativement aux états.

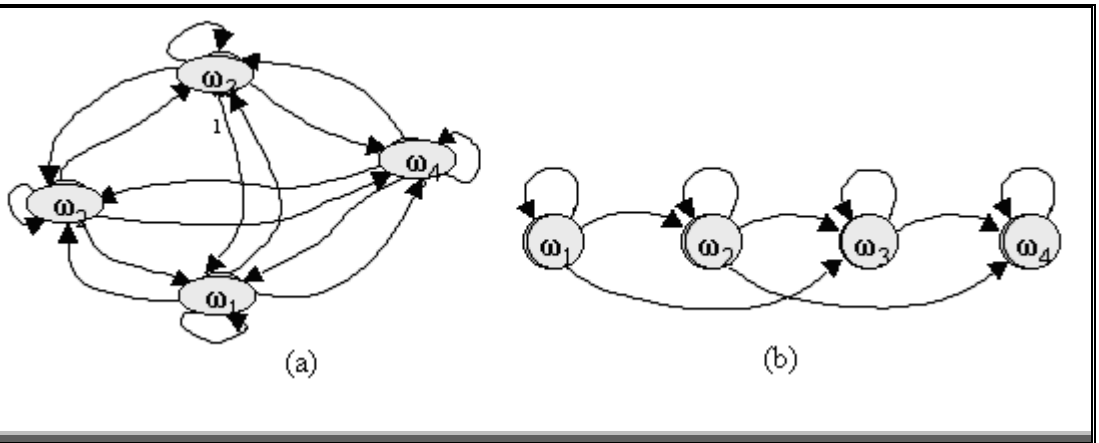


Fig (2.4) : ***Graphes d'état de deux types des modèles MMCs***
(a) Modèle ergodique, (b) Modèle gauche-droit

2.6 : Les trois problèmes liés aux *MMCS*

Etant donné un *MMC*, la plus part des applications sont réduites à résoudre les trois problèmes essentiels liés à une chaîne de Markov cachée [29, 30, 31, 32]. Autrement dit, pour une modélisation à base de chaînes de Markov cachées, il faut répondre aux trois questions suivantes :

Soit λ un *MMC* donné, et soit $Y = (Y_1, Y_2, ..., Y_T)$ une séquence d'observation donnée :

2.6.1 : Evaluation

Le problème d'évaluation est celui permettant de répondre à la question suivante:

Calculer $P(Y/\lambda)$*?*

Où $P(Y/\lambda)$ est la probabilité d'une séquence d'observations Y sachant le modèle λ. C'est le problème nommé aussi, par *'Scoring'*. Par exemple, si on dispose de plusieurs modèles compétitifs, cette démarche permet de choisir le meilleur modèle générant cette séquence d'observations Y.

2.6.2 : Optimisation

Dans ce cas la, on se tente de déterminer la démarche convenable, amenant à la réponse de la question posée ainsi:

Comment choisir une séquence d'état $X = (X_1, X_2, ..., X_T)$*, qui est optimale ?*

On cherche, donc à décoder les états qui correspondent de mieux à la séquence d'observations Y ; c'est pour quoi on utilise souvent le nom : *'Decoding'*.

Alors, cette partie est la section dans laquelle on découvre les états cachées d'un *MMC* ; ou bien, trouver la séquence d'état *correcte*. Notons que cette dernière n'existe pas ; c'est plutôt une *estimation meilleure*. C'est pourquoi en pratique, on se sert des critères d'optimisation ; d'où le nom ***optimisation*** donné à ce problème.

2.6.3 : Apprentissage

Ce problème s'occupe d'ajuster les paramètres du modèle λ, dans l'objectif de maximiser la probabilité d'observation annoncée au premier problème; donc :

Trouver $\hat{\lambda}$ *? tel que :* $\hat{\lambda} = \arg\max_{\lambda} P(Y/\lambda)$.

Cela signifie d'adapter les paramètres (π, A, B) du modèle, pour mieux décrire l'application. La séquence d'observation Y utilisée, dans ce cas est appelée ***séquence d'apprentissage*** ; notée parfois $Y_{Training}$.

Ce dernier problème est crucial, puisqu'il permette de mieux adapter le modèle choisi au phénomène considéré.

2.7 : Solutions des trois problèmes

Dans ce paragraphe, nous allons donner, en bref, les solutions usuelles des trois problèmes ci-dessus selon [31].

2.7.1 : Premier problème : évaluation

On veut trouver la probabilité d'une séquence d'observations Y sachant le modèle λ . C'est déterminer la mesure de vraisemblance $P(Y/\lambda)$.

Rappelons que :

$$P(Y_t / X_t) = b_{X_t}(Y_t) \quad \text{(2.10)}$$

et :

$$P(Y, X/\lambda) = P(Y/X, \lambda) \cdot P(X/\lambda) \quad \text{(2.11)}$$

Tel que $P(Y, X/\lambda)$: est la probabilité jointe de Y et X sachant ce modèle λ .

La démarche suivie pour mesurer la probabilité $P(Y/\lambda)$ est d'énumérer toutes les séquences possibles de longueur T.

Mais pour le moment, considérons $X = (X_1, X_2, ..., X_T)$, une séquence d'état fixe. Selon la propriété d'indépendance conditionnelle des observations, la probabilité $P(Y/X, \lambda)$ est donnée par :

$$P(Y/X, \lambda) = \prod_{t=1}^{T} P(Y_t / X_t, \lambda) \quad \text{(2.12)}$$

Accordée à l'équation (2.10), la formule (2.12) devient comme suit :

$$P(Y/X,\lambda)=\prod_{t=1}^{T} b_{X_t}(Y_t) \qquad (2.13)$$

Ou encore :

$$P(Y/X,\lambda)=b_{X_1}(Y_1)\cdot b_{X_2}(Y_2)\cdot ....\cdot b_{X_{T-1}}(Y_{T-1})\cdot b_{X_T}(Y_T) \qquad (2.14)$$

d'une part.

D'autre part, rappelons la formule (2.9):

$$P(X/\lambda)=\pi_{X_1}\cdot\prod_{t=1}^{T-1} a_{X_t X_{t+1}}$$

La probabilité d'une séquence d'observations Y sachant λ est donnée donc, par la somme de toutes les probabilités jointes, associées aux combinaisons d'état possibles, donnant chacune une séquence X d'états de même longueur T. On écrit :

$$P(Y/\lambda)=\sum_{\text{sur tous les X}} P(Y/X,\lambda)\cdot P(X/\lambda) \qquad (2.15)$$

En substituant (2.9) et (2.14) dans (2.15), on obtient la mesure de vraisemblance comme suit :

$$P(Y/\lambda)=\sum_{\text{sur tous les X}} \pi_{X_1}\cdot b_{X_1}(Y_1)\cdot a_{X_1X_2}\cdot b_{X_2}(Y_2)\cdot a_{X_2X_3}\cdot ....\cdot b_{X_{T-1}}(Y_{T-1})\cdot a_{X_{T-1}X_T}\cdot b_{X_T}(Y_T).. \qquad (2.16)$$

remarque ↑ :

Le calcul précédant de $P(Y/\lambda)$, fait appel à un nombre de multiplications de l'ordre de $(2\cdot T-1)\cdot K^T$, associé à (K^T-1) additions [31]. Il semble être difficile à effectuer même pour des petites valeurs de K et T.

A titre d'exemple, pour un nombre d'états de $K=2$, et une séquence d'observations de $T=100$ de longueur, le nombre total d'opérations est de l'ordre de $(2\cdot 100\cdot 2^{100})=100\cdot 2^{101}$

Cet accès de calcul peut être résolu en utilisant la procédure nommée *'Forward-Backward'*. Elle est basée sur le détermination des deux probabilités *'Forward'* ; $\alpha_t(i)$, et *'Backward'* ; $\beta_t(i)$, dont les spécifications et méthodes de calcul seront données au chapitre suivant.

Notons maintenant que :

$$\alpha_t(i) = P(Y_1, Y_2, ..., Y_t, X_t = \omega_i / \lambda) \quad (2.17)$$

C'est la probabilité de la séquence d'observations partielle jusqu'à l'instant t avec le système est dans la classe ω_i à ce dernier moment t, sachant le modèle λ.

Et :

$$\beta_t(i) = P(Y_{t+1}, Y_{t+2}, ..., Y_T / X_t = \omega_i, \lambda) \quad (2.18)$$

Cette quantité représente la probabilité de la séquence d'observations partielle débutant à l'instant $t+1$ et allant jusqu'à l'instant final, sachant qu'au moment t, le système est dans la classe ω_i, et aussi sachant le modèle λ.

2.7.2 : Deuxième problème : *'decoding'*

Contrairement au premier problème, ou on peut donner une réponse exacte et unique, la solution de celui-ci est non spécifique. La diversité des critères conduit à un choix large de méthodes. Rappelons qu'on cherche la séquence d'état optimale, associée à un vecteur d'observations donnée [29, 31].

Parmi les politiques d'optimisation, l'un s'intéresse à chaque état seul. Il désigne, donc l'état attendu par optimisation individuelle. Cette démarche est basée sur le calcul de la probabilité $\xi_t(i)$; la probabilité d'être à l'état ω_i à l'instant t sachant l'observation complète Y et le modèle λ. La probabilité en question est donnée par :

$$\xi_t(i) = P(X_t = \omega_i / Y, \lambda) \quad (2.19)$$

De même, cette probabilité sera exprimée ultérieurement à l'aide des probabilités *'Forward'* et *'Backward'* ; $\alpha_t(i)$ et $\beta_t(i)$ respectivement.

La séquence d'état optimale est alors déterminée, par l'accumulation des états optimaux. Chaque état vérifie cette caractéristique tout seul et indépendamment des autres. On ecrit:

$$X_t\big|_{optimal} = \arg\max_{1\le i\le K} \xi_t(i), \quad 1 \le t \le T \qquad (2.20)$$

Un autre néglige les états isolés et exploite la notion « *chemin* ». Il s'intéresse donc, aux chemins partiaux optimaux. Un des premiers dits algorithmes, utilisé avec les *MMCs*, est l'algorithme de ***Viterbi***. Il détermine ce chemin optimal progressivement, tout en conservant l'optimalité partielle [29, 31, 35]. Pour utiliser ce dernier, une nouvelle probabilité à introduire, est $\delta_t(i)$ avec :

$$\delta_t(i) = \max_{X_1, X_2, ..., X_{t-1}} P(X_1, X_2, ..., X_t = \omega_i, Y_1, Y_2, ..., Y_t / \lambda) \qquad (2.21)$$

Cette mesure représente la probabilité jointe maximale correspondante aux sous séquences d'observations et d'état ; $(Y_1, Y_2, ..., Y_t)$ et $(X_1, X_2, ..., X_{t-1})$ respectivement, et avec le processus se trouvant actuellement à l'état ω_i.

Par induction, on trouve:

$$\delta_{t+1}(j) = \left[\max_{X_1, X_2, ..., X_t} \delta_t(i) \cdot a_{ij}\right] \cdot b_{X_{t+1}=\omega_j}(Y_{t+1}) \qquad (2.22)$$

Une fois de plus, on fait une accumulation d'états vérifiant la maximisation, en se servant d'un accumulateur γ (tableau), avec :

$$\gamma_t(j) = \arg\max_{1\le i\le K}\left[\delta_{t-1}(i) \cdot a_{ij}\right], \quad \begin{cases} 2 \le t \le T \\ 1 \le j \le K \end{cases} \qquad (2.23)$$

L'algorithme complet de *Viterbi* est donné ultérieurement au chapitre suivant.

2.7.3 : Troisième problème : 'training'

Ce problème est classé, comme étant, le plus difficile des trois. Sa solution vise à déterminer une méthode capable d'ajuster les paramètres de λ ; c'est maximiser $P(Y_{Training} / \lambda)$ au sens du maximum de vraisemblance [31], dont les paramètres estimés $\hat{\lambda}$ sont donnés comme suit:

$$\hat{\lambda} = \arg\max_{\lambda} P(Y_{Training} / \lambda) \quad \text{(2.24)}$$

Dans la littérature, il n'y a pas de méthodes analytiques spécifiques pour maximiser la probabilité précédente. On se sert donc, des procédures itératives. Un algorithme répondu, dans ce domaine est celui de *Baum-Welch*, connu aussi par algorithme *'Forward_Backward'*. D'autres utilisent son équivalent, dit *algorithme EM* ; pour *'Expectation-Modification'*.

Pour appliquer ces procédures itératives, on a besoin de définir une nouvelle probabilité, nomée $\psi_t(i, j)$, et mesurant le fait d'être à l'état ω_i à l'instant t, et à l'état ω_j à l'instant suivant, sachant la séquence d'observations Y et le modèle λ. On exprime celle ci par :

$$\psi_t(i, j) = P(X_t = \omega_i, X_{t+1} = \omega_j / Y, \lambda) \quad \text{(2.25)}$$

Cette nouvelle quantité est écrite en fonction des deux fameuses probabilités *'Forward'* et *'Backward'*. Par la suite, on déduit tous les paramètres nécessaires à la mise à jours du modèle (voir chapitre suivant).

remarque ↑ :

Notons seulement que cette algorithme démarre d'un *MMC* initial connu, dont les paramètres affectent sa performance attendue. Une bonne initialisation, alors est toujours demandée.

2.8 : Conclusion

Les Modèles de Markov cachés *MMCs*, précisément les Chaînes de Markov Cachées, sont donc des outils de modélisation très utiles. Leur choix en vue de résoudre un problème bien défini nécessite une spécification complète de leurs éléments ; à savoir: la matrice de transition, les distributions initiales et les densités d'émission des observations.

Ces modèles peuvent être classés comme des méthodes probabilistes, d'une part. D'autre part, ils appartiennent à la famille des modèles pouvant être représentés graphiquement à l'aide d'arcs et de cercles.

Pour la modélisation d'un phénomène quelconque par ces outils là, il est donc, indispensable de répondre aux trois problèmes essentiels associés à ces modèles, dont nous avons brièvement, nommé et donné les solutions usuelles utilisées. Nous avons aussi, défini les diverses probabilités intermédiaires nécessaires à l'intégration des algorithmes destinés à établir ces solutions.

Ainsi, le chapitre suivant couvre plus largement, les procédures et les algorithmes dédiés à la résolution de ces problèmes, et l'exploitation de ces *MMCs* dans des domaines variés. Par la suite, nous verrons leur intégration en vue de la détection d'objet mobile.

Modélisation par les Modèles de Markov Cachés

3.1 : Introduction

Les modélisations basées sur les Modèles de Markov cachées *MMCs*, permettent d'aborder de nombreux problèmes, liés au traitement du signal et de l'image [37]. Ils offrent un cadre probabiliste d'analyse, et donnent la possibilité d'exprimer des corrélations statistiques entre les états du système. Celles-ci se présentent successivement, donnant ainsi une séquence faisant partie d'un processus observé.

On considère donc, qu'on a accès à une version bruitée du signal (image) modélisé par cette chaîne. Alors, le problème général est celui d'estimation de la réalisation non observable de la chaîne par l'intermédiaire de cette observation.

Le mot « bruit », quant à lui, modélise la variabilité naturelle entre les classes considérées, ainsi que les perturbations (bruit d'acquisition, mouvement de la caméra,…). Il est injecté par les lois des observations conditionnelles aux états.

Nous rappelons, dans ce chapitre, les définitions relatives aux diverses lois de probabilités liées à l'utilisation d'une Chaîne de Markov Cachée et évoquées au chapitre

précédant, ainsi que les modes de leurs calculs ; motivations nécessaires pour aborder les problèmes de l'estimation des paramètres et de classification, d'où ceux de la détection.

3.2 : Chaîne de Markov cachée et détection

Soit I, un ensemble fini correspondant à une image de N pixels (avec $N = L \times L$, et L est le nombre des lignes et aussi des colonnes). On considère $X = (X_t)_{t \in T}$ et $Y = (Y_t)_{t \in T}$, deux processus aléatoires tel que :

- Y : représente l'image observée définie par la différence entre l'image actuelle et l'image *'background'*.
- X : représente une classe inconnue au sens du mouvement ; ensemble d'étiquettes.

Une Chaîne de Markov Cachée, est un processus à temps discret, doublement stochastiques (X, Y)[36, 37, 45]. Le processus X est une chaîne de Markov, tel que chaque variable aléatoire X_t prend ses valeurs dans un ensemble fini de classes, donné par : $\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$. Tandis que, chaque Y_t est une valeur réelle (mesure). On introduit les quantités $x = (x_t)_{t \in T}$ et $y = (y_t)_{t \in T}$ représentant les réalisations de X et Y respectivement.

Rappelons, tout d'abord, la notion d'indépendance conditionnelle énoncée précédemment, et donnée par la définition ci-après :

Définition :

***Indépendance conditionnelle*:**

Soit X, Y et Z des variables aléatoires quelconques. On dit que X et Y sont indépendants relativement à (ou sachant) Z si et seulement si :

$$P(X, Y / Z) = P(X / Z) \cdot P(Y / Z)$$

Conformément à cette définition, on suppose que les variables aléatoires $Y = (Y_t)_{t \in T}$ sont conditionnellement indépendants à X, et la distribution de chaque Y_t conditionnelle à X est égale seulement à sa distribution conditionnelle à X_t. On suppose encore, que la distribution de Y_t conditionnelle à X_t, est représentée par une densité de probabilité gaussienne de Y_t ; notée $f_{X_t}(y_t)$. Ainsi, l'ensemble de ces distributions constitue K densités $f_1, f_2, ..., f_K$ correspondantes aux K classes considérées.

Donc, la loi a posteriori $P(Y = y / X = x)$ est donnée par :

$$P(Y = y / X = x) = \prod_{t \in T} P(Y_t = y_t / X_t = x_t) = \prod_{t \in T} f_{X_t}(Y_t = y_t) \ldots\ldots\ldots\ldots\ldots\ldots \quad (3.1)$$

où chaque gaussienne $f_{X_t}(y_t)$ est écrite sous la forme suivante:

$$f_{X_t}(y_t) = \frac{1}{\sqrt{2 \cdot \pi} \cdot \sigma_t} \cdot \exp\left[\frac{(y_t - \mu_t)^2}{2 \cdot \sigma_t^2} \right] \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.2)$$

Les μ_t et σ_t représentent les deux caractéristiques de la gaussienne ; moyenne et variance.

Nous allons donc, modéliser les relations entre les variables aléatoires $(X_t)_{t \in T}$, en considérant que la distribution à priori de X ; $P(X)$, est un processus markovien [45]. Il est de nature cachée car le vecteur X n'est pas, directement observable. Le problème de détection est ainsi, converti à celui d'une estimation ; où l'on estime la réalisation x de X à partir de celle observée y de Y.

3.3 : Lois de probabilité liées aux *MMCs*

Comme on l'a signalé précédemment, les chaînes constituent le modèle markovien le plus simple. Elles sont utilisées pour représenter un signal unidimensionnel (1D). Les applications liées aux images présentées dans l'espace bidimensionnel, nécessitent alors une étape préliminaire, destinée à établir la transformation de cet espace (2D) vers un autre unidimensionnel (1D).

Par conséquent, l'entrée de l'algorithme de traitement sera un vecteur obtenu par une transformation bien spécifiée, qui peut être :

- Ligne par ligne.
- Colonne par colonne.
- En diagonal.
- Soit en utilisant des parcours spécifiques ; citons à titre d'exemple le parcours d' '*Hilbert-piano* '

3.3.1 : Les probabilités jointes d'une chaîne de Markov cachée

Soit $X = (X_t)_{t=1}^{T}$ une chaîne de Markov cachée, supposée homogène et stationnaire, et soit $Y = (Y_t)_{t=1}^{T}$ le vecteur d'observations. Les déférentes probabilités jointes liées de cette chaîne, peuvent s'écrire en fonction des probabilités jointes notées c_{ij}, supposées indépendantes de t [37].

3.3.1.1 : Les probabilités jointes c_{ij}

La probabilité jointe c_{ij} donne la probabilité d'être à l'état ω_i à l'instant t et à l'état ω_j à l'instant suivant. Elle est exprimée par :

$$c_{ij} = P(X_t = \omega_i, X_{t+1} = \omega_j) \quad (3.3)$$

3.3.1.2 : Les probabilités initiales π_i

La mesure π_i détermine la probabilité que le modèle soit à l'état ω_i au départ. Elle peut s'écrire en fonction des probabilités jointes c_{ij} comme suit :

$$\pi_i = P(X_1 = \omega_i) = \sum_{1 \le j \le K} c_{ij} \quad (3.4)$$

3.3.1.3 : Les probabilités de transition

La probabilité a_{ij} représente le saut du processus de la classe (état) ω_i à la classe ω_j, entre deux temps successifs. Elle est écrite comme suit :

$$a_{ij} = P(X_t = \omega_j / X_{t-1} = \omega_i) = \frac{c_{ij}}{\sum_{1 \le j \le K} c_{ij}} \quad (3.5)$$

3.3.1.4 : La loi de X

C'est la probabilité à priori de X, notée $P(X)$. Par définition, elle est totalement déterminée par la connaissance de sa distribution initiale π_i de X_1, et ses distributions de transition a_{ij}. Elle s'écrit :

$$P(X) = P(X_1 = \omega_{X_1}, X_2 = \omega_{X_2}, ..., X_T = \omega_{X_T}) = \pi_{X_1} \cdot a_{X_1X_2} \cdot a_{X_2X_3} \cdot ... \cdot a_{X_{T-1}X_T} .... \quad (3.6)$$

En substituant (3.4) et (3.5) dans (3.6), on constate que les paramètres $(c_{ij})_{1\le i\le K, 1\le j\le K}$ suffiront pour déterminer la probabilité $P(X)$ [37, 45].

3.3.1.5 : Les probabilités '*Forward*' et ' *Backward*'

Nous rappelons dans cette sous section, la définition des probabilités '*Forward*' $\alpha_t(i)$, et '*Backward*' $\beta_t(i)$, qui jouent un rôle crucial, aussi bien au niveau de l'estimation des paramètres qu'à celui de la détection proprement dite. Le calcul des déférentes probabilités liées aux algorithmes utilisées, est basé directement sur la détermination de ces deux quantités. Une fois encore, on rappelle leurs définitions :

$$\alpha_t(i) = P(X_t = \omega_i, Y_1 = y_1, Y_2 = y_2, ..., Y_t = y_t) ..... \quad (3.7)$$

et :

$$\beta_t(i) = P(Y_{t+1} = y_{t+1}, Y_{t+2} = y_{t+2}, ..., Y_T = y_T / X_t = \omega_i) ..... \quad (3.8)$$

Elles se calculent récursivement. Celle '*Forward*' est à récurrence montante d'où son nom [36], et donnant la probabilité de n'importe quel état ω_j à l'instant suivant par :

$$\begin{aligned} \alpha_{t+1}(j) &= P(X_{t+1} = \omega_j, Y_1 = y_1, Y_2 = y_2, ..., Y_t = y_t, Y_{t+1} = y_{t+1}) \\ &= \sum_{1\le i\le K} P(X_{t+1} = \omega_j, X_t = \omega_i, Y_1 = y_1, Y_2 = y_2, ..., Y_t = y_t, Y_{t+1} = y_{t+1}) \\ &= \sum_{1\le i\le K} P(X_{t+1} = \omega_j, Y_{t+1} = y_{t+1} / X_t = \omega_i) \cdot P(X_t = \omega_i, Y_1 = y_1, Y_2 = y_2, ..., Y_t = y_t) \end{aligned} \quad (3.9)$$

D'où la relation dite d'induction :

$$\alpha_{t+1}(j) = \left(\sum_{1 \le i \le K} \alpha_t(i) \cdot a_{ij} \right) \cdot f_j(y_{t+1}) \quad \text{(3.10)}$$

Tandis que les probabilités *'Backward'* se calculent par récurrence descendante :

$$\begin{aligned} \beta_t(i) &= P(Y_{t+1} = y_{t+1}, Y_{t+2} = y_{t+2}, ..., Y_T = y_T / X_t = \omega_i) \\ &= \sum_{1 \le j \le K} P(X_{t+1} = \omega_j, Y_{t+1} = y_{t+1}, Y_{t+2} = y_{t+2}, ..., Y_T = y_T / X_t = \omega_i) \\ &= \sum_{1 \le j \le K} a_{ij} \cdot P(Y_{t+1} = y_{t+1}, Y_{t+2} = y_{t+2}, ..., Y_T = y_T / X_{t+1} = \omega_j) \quad \text{(3.11)} \\ &= \sum_{1 \le j \le K} a_{ij} \cdot f_j(y_{t+1}) \cdot P(Y_{t+2} = y_{t+2}, Y_{t+3} = y_{t+3}, ..., Y_T = y_T / X_{t+1} = \omega_j) \end{aligned}$$

Par induction, elle devient :

$$\beta_t(i) = \sum_{1 \le j \le K} a_{ij} \cdot f_j(y_{t+1}) \cdot \beta_{t+1}(j) \quad \text{(3.12)}$$

remarques ↑ :

- Notons que la loi de Y peut se calculer de manières diverses, par exemple:

$$P(Y = y) = \sum_{1 \le i \le K} P(X_T = \omega_i, y)$$

ou bien : $P(Y = y) = \sum_{1 \le i \le K} \alpha_T(i)$

ou encors :

$$P(Y = y) = \sum_{1 \le i \le K} \pi_i \cdot f_i(y_1) \cdot \beta_1(i)$$

- L'intérêt des probabilités *'Forward'* et *'Backward'* se voit dans leurs utilisations dans le calcul des probabilités spécifiques ; celle du mode des marginales par exemple, exprimée par :

$$\begin{aligned} P(X_t = \omega_i, Y = y) &= P(X_t = \omega_i, Y_1 = y_1, ..., Y_t = y_t, Y_{t+1} = y_{t+1}, ..., Y_T = y_T) \\ &= P(X_t = \omega_i, Y_1 = y_1, ..., Y_t = y_t) \cdot P(Y_{t+1} = y_{t+1}, ..., Y_T = y_T / X_t = \omega_i) \end{aligned}$$

On a donc :

$$P(X_t = \omega_i, Y = y) = \alpha_t(i) \cdot \beta_t(i)$$

3.3.2 : Les probabilités a posteriori d'une chaîne de Markov cachée

Le calcul respectif des formules *'Forward'* et *'Backward'* dérivées de (3.7) et (3.8) (ainsi que dans une moindre mesure celui des probabilités $\delta_t(i)$ de l'algorithme de *Viterbi*), se heurte à des difficultés d'ordre numérique [35, 37, 45]. En effet, les quantités figurant dans les expressions de $\alpha_t(i)$ et $\beta_t(i)$ sont très petites. D'un point de vue pratique, on dépasse rapidement les capacités de la plus part des machines à représenter des nombres réels aussi petits. En addition, la présence des sommes dans les formules manipulées, interdit tout recours au logarithme. A fin de s'affranchir de ce problème de dépassement de capacité, *Divijver* et ses collègues, ont proposé une formulation en termes de probabilités à posteriori, et non plus jointes [37, 45].

3.3.2.1 : Les probabilités '*Forward*' et '*Backward*'

Les reformulations de $\alpha_t(i)$et $\beta_t(i)$ sont les suivantes :

$$\alpha_t(i) \approx P(X_t = \omega_i / Y_1 = y_1,...,Y_t = y_t) \quad (3.13)$$

et

$$\beta_t(i) \approx \frac{P(Y_{t+1} = y_{t+1},...,Y_T = y_T / X_t = \omega_i)}{P(Y_{t+1} = y_{t+1},...,Y_T = y_T / Y_1 = y_1,...,Y_t = y_t)} \quad (3.14)$$

La procédure complète de calcul de la probabilité '*Forward*' devient celle du tableau **Tab (3.1)**. Et de même, la probabilité '*Backward*' devient celle de **Tab (3.2)**.

3.3.2.2 : Les probabilités a posteriori marginales

Notée $\xi_t(i)$, elle représente la probabilité que l'élément d'ordre t appartienne à la classe ω_i, en connaissant la séquence entière d'observations Y. Elle est exprimée en fonction des fameuses probabilités $\alpha_t(i)$ et $\beta_t(i)$ comme suit :

$$\xi_t(i) = P(X_t = \omega_i / Y = y) = \frac{\alpha_t(i) \cdot \beta_t(i)}{\sum_{1 \le l \le K} \alpha_t(l) \cdot \beta_t(l)} \quad (3.15)$$

3.3.2.3 : Les probabilités jointes conditionnelles

Chacune est notée par $\psi_t(i,j)$, et représentant la probabilité d'être à l'instant t dans la classe ω_i et d'appartenir à la classe ω_j juste après, tout en connaissant la séquence d'observations Y. Elle s'écrit :

$$\psi_t(i,j) = P\left(X_t = \omega_i, X_{t+1} = \omega_j / Y = y\right) \quad \text{.....} \quad (3.16)$$

Exprimée en fonction de $\alpha_t(i)$ et $\beta_t(i)$, elle devient :

$$\psi_t(i,j) = \frac{\alpha_t(i)\cdot a_{ij} \cdot f_j(y_{t+1})\cdot \beta_{t+1}(j)}{\sum_{1\le l\le K} f_l(y_t)\cdot \sum_{1\le m\le K}\alpha_t(m)\cdot a_{ml}} \quad \text{.....} \quad (3.17)$$

D'où :

$$\psi_t(i,j) = \frac{\alpha_t(i)\cdot a_{ij} \cdot f_j(y_{t+1})\cdot \beta_{t+1}(j)}{\sum_{1\le m\le K}\sum_{1\le l\le K}\alpha_t(m)\cdot a_{ml} \cdot f_l(y_{t+1})\cdot \beta_{t+1}(l)} \quad \text{.....} \quad (3.18)$$

remarques ↑

- La quantité : $\sum_{1\le j\le K}\alpha_t(l)\cdot\beta_t(l)$, n'est qu'un facteur de normalisation. En d'autres termes, il donne le rapport entre la probabilité jointe et celle a posteriori. Il représente la probabilité $P(Y=y)$. Et de même pour le dénominateur de la formule donnant $\psi_t(i,j)$.
- Aussi, les probabilités $\psi_t(i,j)$ et $\xi_t(i)$ sont liées entre elles par la relation :

  $$\xi_t(i) = \sum_{1\le j\le K}\psi(i,j).$$

- Nous remarquons par la suite, que l'étape d'estimation des états cachés X à partir de l'observation Y, s'achève par le calcul de la loi a posteriori de X donnée par $P(X = x/Y = y)$. Il peut être montré que celle-ci est une chaîne de Markov (non stationnaire) [37, 45], dont les probabilités de transition sont données par:

  $$t_{ij}^t = P\left(X_{t+1} = \omega_j / X_t = \omega_i, Y = y\right)$$

Donc :

$$t_{ij}^{t} = \frac{a_{ij} \cdot f_j(y_{t+1}) \cdot \beta_{t+1}(j)}{\sum_{1 \le l \le K} a_{il} f_l(y_{t+1}) \cdot \beta_{t+1}(l)} \quad \text{..... (3.19)}$$

- En utilisant la formule ci-dessus, on peut directement simuler des réalisations a posteriori de X suivant une démarche récursive, et sans recours à des méthodes itératives, coûteuses en temps de calcul [37] (par exemple : échantillonneur de Gibbs utilisé par les champs de Markov cachés.

Ainsi, la classe du premier pixel est choisie aléatoirement à partir de la distribution a posteriori marginale $\xi_t(i)$ donnée par (3.15). Ensuite et par récurrence, la classification du nouveau pixel passe par les étapes suivantes :

- Fixer la classe ω_i du pixel précédant.
- Calculer les probabilités t_{ij}^{t} suivant (3.19).
- Prélever aléatoirement la classe du pixel à classer ω_j selon t_{ij}^{t}.

Tab (3.1) : *Algorithme 'Forward'*

- Initialisation :

$$\alpha_1(i) = \frac{\pi_i \cdot f_i(y_1)}{\sum_{1 \le j \le K} \pi_j \cdot f_j(y_1)} \qquad pour \quad : 1 \le i \le K$$

- Induction :

$$\alpha_t(i) = \frac{f_i(y_t) \cdot \sum_{1 \le j \le K} \alpha_{t-1}(j) \cdot a_{ij}}{\sum_{1 \le l \le K} f_l(y_t) \cdot \sum_{1 \le j \le K} \alpha_{t-1}(j) \cdot a_{jl}} \qquad pour \quad : 1 \le i \le K, \ t = 2,...,T$$

et :

Tab (3.2) : *Algorithme 'Backward'*

- Initialisation :

$$\beta_T(i) = 1 \qquad pour \quad : 1 \le i \le K$$

- Induction :

$$\beta_t(i) = \frac{\sum_{1 \le j \le K} a_{ij} \cdot f_j(y_{t+1}) \cdot \beta_{t+1}(j)}{\sum_{1 \le l \le K} f_l(y_{t+1}) \cdot \sum_{1 \le j \le K} \alpha_t(j) \cdot a_{jl}} \qquad pour \quad : 1 \le i \le K, \ t = T-1,...,1$$

3. 4 : Transformation d'une image en une chaîne

Lorsqu'on modélise par une Chaîne de Markov Cachée, il est nécessaire, à tout prés, de faire une transformation de la représentation (2D) de cette image, à une autre unidimensionnelle (1D). Ce passage est effectué selon une stratégie bien définie. Une méthode simple consiste à scanner l'image *« ligne par ligne »* ou même *« colonne par colonne »*. Malheureusement, ces parcours horizontaux et verticaux ne respectent pas le contexte spatial de l'image [37]. Quant on considère, par exemple la lecture horizontale, deux pixels voisins et appartenants à une même ligne seront proches spatialement, mais éloignés au sens de la chaîne de Markov. A fin d'améliorer, donc l'adéquation du contexte temporel de la chaîne au contexte spatial de l'image, *Benmiloud* et *Pieczynski* [35, 37, 45], proposent un parcours du type *d'Hilbet-piano.*

Celui-ci est une courbe formée d'objets fractals particuliers conçus pour parcourir tous les points d'un plan de taille $2^{K'} \times 2^{K'}$ et donner un chemin sans discontinuités respectant le voisinage spatial. Cette courbe est obtenue par reproduction d'un élément de base, nommé « *générateur* ». La figure **Fig (3.1)**, donne un exemple d'un parcours couvrant un espace 16×16 ($K' = 4$), ainsi que les étapes de sa génération.

Des parcours adaptés aux images rectangulaires peuvent être trouvés. Ces derniers sont appelés « *parcours généralisés* ».

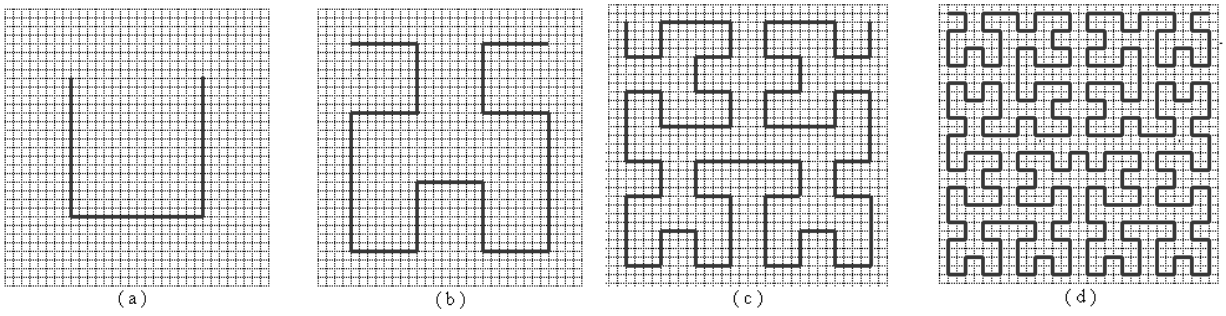


Fig (3.1) : ***Construction du Parcours d'Hilbet-piano pour une image 16x16***
(a)Initialisation, (b) et (c) Etapes intermédiaires, (d) Résultat final

Exemple

Pour montrer l'effet de ces parcours, nous avons pris deux images. La première est appelée « *Carrés* ». Elle est obtenue par un logiciel de dessin. Tandis que la deuxième appelée « *Disque* », est réelle. Elle est prise par une caméra, et elle représente un disque posé sur une table. Elles sont toutes les deux de taille 32x32 pixels.

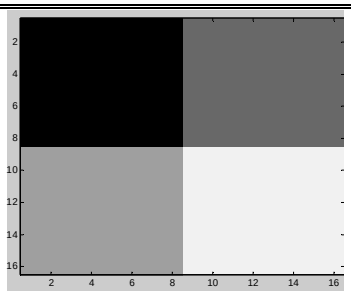


Image « *carrés* » : (a)

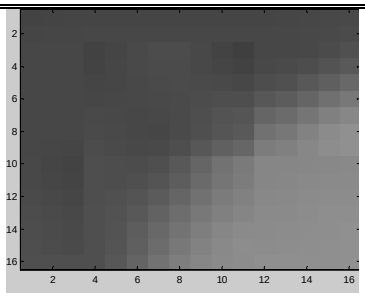


Image « *Disque* » : (a)

Image « *carrés* » : (b)

Image « *Disque* » : (b)

Image « *carrés* » : (c)

Image « *Disque* » : (c)

Fig (3.2) : ***Comparaison entre Parcours d'Hilbet-piano et celui Ligne par ligne : (a) images originales en niveau de gris, (b) Parcours : Ligne par ligne, (c) : Parcours d'Hilbert- Piano***

3.5 : Estimation Bayésienne et modèles *MMCs*

Dans cette partie, nous ferons un tour d'horizon rapide, des méthodes d'optimisation bayesienne associées aux modèles Markoviens (soit lors d'estimation des paramètres, soit pendant la classification). Nous allons en premier, rappeler les déférents critères utilisés au niveau de l'optimisation des états. Ensuite nous présentons l'algorithme souvent choisi pour estimer les paramètres du modèle *MMC*. Notons seulement que l'estimation peut se diviser en deux types :

Une première dédiée à estimer les paramètres proprement dits de la chaîne de Markov. Une autre vise à définir les caractéristiques des densités modélisant les interactions observations-états cachés. Nous résumons ces méthodes par le schéma synoptique illustré par la figure **Fig (3.3)** suivante.

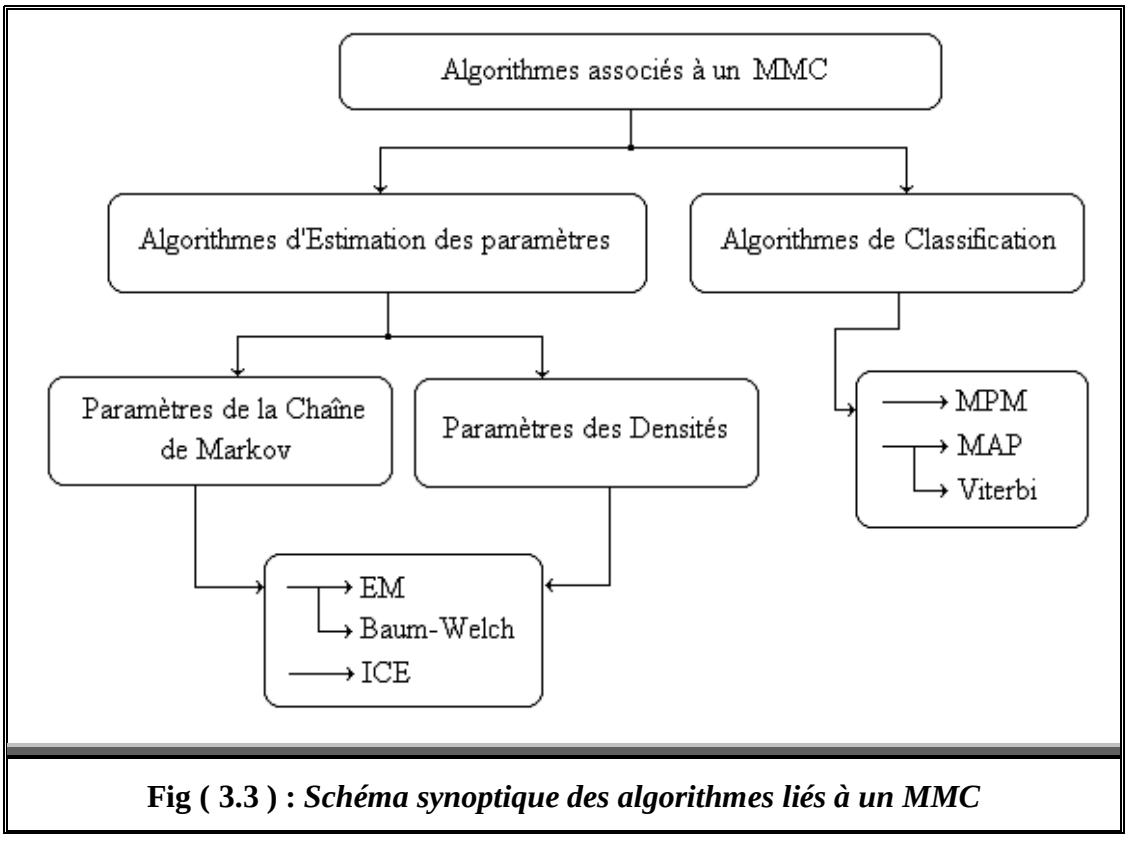


Fig (3.3) : ***Schéma synoptique des algorithmes liés à un MMC***

Le but espéré est celui de l'estimation des variables dites cachées ; encore appelées « *étiquettes* X », à partir de l'observation y .

Les techniques existantes font généralement appel à deux distributions. La première est la distribution à priori des champs d'étiquettes $P(X)$, dépendant des paramètres de la chaîne. Cette distribution représente l'information contextuelle dont on dispose loin de toute

observation. La seconde distribution est la vraisemblance des observations. Autrement dit, la loi conditionnelle $P(Y/X)$. Sa forme est généralement choisie d'avance. On suppose qu'elle apparient à une famille paramétrée (dans notre cas, on considère la famille gaussienne, avec comme paramètres les moyennes et les variances) [35, 37].

L'estimation Bayesienne tire son nom de l'usage intensif de loi de Bayes, qui permet à partir des deux distributions précédentes $P(X)$et $P(Y/X)$de remonter à la loi jointe $P(X,Y)$, donnée ainsi par :

$$P(X,Y) = P(Y/X)\cdot P(X) \quad \text{(3.20)}$$

Nous citons parmi ces méthodes deux d'entre elles, correspondant à deux fonctions de coût différentes et permettant l'estimation de X .

3.5.1 : Le Maximum A Posteriori *MAP*

Ce critère est considéré comme celui le plus classique. Il est souvent utilisé. C'est le Maximum a Posteriori donnant l'estimé $\hat{x}$ par :

$$\hat{x} = \arg\max_x P(X = x/Y = y) \quad \text{(3.21)}$$

Il peut s'écrire en fonction de la probabilité jointe comme suit :

$$\hat{x} = \arg\max_x P(X = x, Y = y) \quad \text{(3.22)}$$

(Puisque $P(y)$ ne dépend pas de x : il n'y a pas de relation directe entre les réalisations x et y [35]).

La solution analytique est donnée par l'algorithme de *Viterbi*.

3.5.2 : Le Mode de la Marginale à Posteriori *MPM*

C'est l'estimateur par Mode des Marginales à Posteriori. Il s'intéresse à chaque point (à l'instant t) de côté. Il s'exprime par :

$$\hat{x}_t = \arg\max_{x_t} P(X_t = x_t / Y = y), \quad \forall t \quad \text{(3.23)}$$

L'estimée de chaque point est fournie donc, à travers la loi du mode de la marginale à posteriori $P(x_t / y)$, d'où son nom [35, 37].

3.5.3 : Le Maximum de Vraisemblance *MV*

Ce critère est classé par la communauté comme non Bayesien ; du fait qu'il n'y a pas de fonction de coût correspondante [35]. Cependant, il est proche des autres. Il est basé sur la maximisation de la Vraisemblance des observations, et il est donné par :

$$\hat{x} = \arg\max_x P(Y = y / X = x) \quad \text{(3.24)}$$

Dans la pratique, on fait souvent l'hypothèse suivante, dite d'indépendance conditionnelle :

$$P(Y / X) = \prod_t P(Y_t / X_t) \quad \text{(3.25)}$$

Celle-ci engendre l'inconvénient majeur de la Vraisemblance ; celui d'ignorer l'information contextuelle donnant ainsi l'estimation de X par :

$$\forall t, \quad \hat{x}_t = \arg\max_{x_t} P(Y_t = y_t / X_t = x_t) \quad \text{(3.26)}$$

3.6 : Algorithmes associés aux modèles *MMCs*

Le chois d'un modèle *MMC* en vue de résoudre les problèmes liés à une application bien définie, fait appel à des algorithmes spécifiques. Ces derniers sont dirigés soit pour définir le modèle (cas d'un apprentissage non supervisé), soit pour restituer les états de la chaîne. Ils sont montés à partir des déférentes probabilités liées à la chaîne de Markov cachée, et citées ci loin. On classe ces procédures donc en :

3.6.1 : Algorithmes de classification

Soit $(X_t, Y_t)_{t=1}^T$ une chaîne de Markov cachée homogène. Les probabilités de transition sont notées a_{ij} avec $a_{ij} = P(X_{t+1} = \omega_j / X_t = \omega_i), \forall t$. De plus, le vecteur d'observation Y est supposé indépendant relativement au celui d'états ; X.

3.6.1.1 : Le *MPM* pour une chaîne de Markov cachée

Cet algorithme consiste à maximiser pour chaque état la marginale à posteriori $P(X_t = x_t / Y = y)$ [35, 37]. Celle-ci est précédemment nommée par $\xi_t(i)$. En plus, elle a été exprimée en fonction des probabilités '*Forward*' et '*Backward*'. Si on adopte l'algorithme *MPM*, la solution est calculée directement, sans procédures itératives. On se base seulement sur les $\alpha_t(i)$et $\beta_t(i)$.

Ainsi pour chaque élément t de la chaîne et pour chaque classe ω_i, on calcule :

- les probabilités '*Forward*' $\alpha_t(i)$.
- les probabilités '*Backward*' $\beta_t(i)$.
- les probabilités à posteriori marginales; $\xi_t(i)$.

Par la suite, la classification est établie :

Pour chaque pixel t, on choisie la classe maximisant $\xi_t(i)$, tel que :

$$x_t = \arg\max_{\omega_i} P(X_t = \omega_i / Y = y) \quad \text{…..…….……..…….} \quad (3.27)$$

3.6.1.2 : Algorithme de *Viterbi*

Employé au départ dans le cadre de la programmation dynamique pour la recherche des chemins optimaux, il a été par la suite largement utilisé pour résoudre les problèmes de reconnaissance de la parole, et par ailleurs de l'écriture [31, 37].

Il s'appuie sur le calcul des probabilités $\delta_t(i)$ (déjà annoncées), correspondantes aux « *sous chemins* optimaux» ; chacun s'étalant de X_1 à ω_i visitée à l'instant t. La quantité $\delta_t(i)$ s'écrit de la manière suivante:

$$\delta_t(i) = \max_{x_1, x_2, \ldots, x_{t-1}} P(X_1 = x_1, X_2 = x_2, \ldots, X_{t-1} = x_{t-1}, X_t = \omega_i, Y_1, Y_2, \ldots, Y_{t-1}) \ldots\ldots\ldots \quad (3.28)$$

En d'autre termes, on cherche le *MAP* sur le sous chemin amenant à ω_i fixée [35]. Par récurrence, on obtient :

$$\delta_{t+1}(j) = \max_{x_1, x_2, \ldots, x_t} P(X_1 = x_1, X_2 = x_2, \ldots, X_t = x_t, X_{t+1} = \omega_j, Y_1, Y_2, \ldots, Y_t) \ldots\ldots\ldots\ldots \quad (3.29)$$

Ou :

$$\delta_{t+1}(j) = \max_{x_t}\left[P(X_{t+1} = \omega_i / X_t = x_t) \max_{x_1,...,x_{t-1}} P(X_1 = x_1, X_2 = x_2,..., X_t = x_t, Y_1, Y_2,..., Y_t) \right]$$

$$\delta_{t+1}(j) = \max_{\omega_i}\left[a_{ij} \cdot P(y_t / X_t = \omega_i) \cdot \max_{x_1,...,x_{t-1}} P(X_1 = x_1, X_2 = x_2,..., X_t = \omega_i, Y_1, Y_2,..., Y_{t-1}) \right]$$

On aura donc :

$$\delta_{t+1}(j) = \max_{\omega_i}\left(a_{ij} \cdot f_i(y_t) \cdot \delta_t(i)\right) \quad (3.30)$$

On détecte et on sauvegarde l'argument qui réalise le maximum par :

$$\gamma_{t+1}(j) = \arg\max_{\omega_i}\left(a_{ij} \cdot f_i(y_t) \cdot \delta_t(i)\right) \quad (3.31)$$

L'algorithme de *Viterbi* est résumé par le tableau **Tab (3.3)** .

remarques ↑ :

- L'algorithme de *Viterbi* est non itératif, donc très rapide. Cependant, il est coûteux en espace mémoire [35], car il faut stoquer deux tables (pour les $\delta_t(.)$ et les $\gamma_t(.)$). Il est clair que la taille nécessaire dépend de l'espace d'état, aussi bien de la longueur de la chaîne.
- Le calcul de l'algorithme de *Viterbi* est similaire à celui de l'algorithme 'forward', à l'exception de l'étape de maximisation remplacée dans le dernier par une sommation. Une autre différence à signaler, est l'étape connue par le '*Backtracking'* dans laquelle on effectue une lecture descendante, ou bien un retour arrière sur le tableau de pointeurs en partant de $\hat{x}_T = \arg\max_i \delta_T(i)$.

Tab (3.3) : ***Algorithme de Viterbi***

- Initialisation :

$$\delta_1(i) = P(X_1 = \omega_i) = \pi_i \qquad pour \quad : 1 \le i \le K$$

- Récurrence montante pour $2 \le t \le T$: on conserve les $\delta_t(j)$ pour tous les j *et les* $\gamma_t(j)$ correspondant dans une table, avec :

$$\delta_{t+1}(j) = \max_{\omega_i}\left(a_{ij} \cdot f_i(y_t) \cdot \delta_t(i)\right)$$

et $$\gamma_{t+1}(j) = \arg\max_{\omega_i}\left(a_{ij} \cdot f_i(y_t) \cdot \delta_t(i)\right)$$

- Fin de récurrence, le MAP est donné par :

$$\max_x P(X = x, Y = y) = \max_i \delta_T(i)$$

et $$\hat{x}_T = \arg\max_i \delta_T(i)$$

- Une dernière étape consiste alors à effectuer une lecture descendante de la table. Le chemin optimal est donné par :

$$\hat{x}_t = \gamma_{t+1}(\hat{x}_{t+1}), T - 1 \ge t \ge 1$$

3.6.2 : Algorithmes d'estimation des paramètres

Nous présentons ici l'un des algorithmes dédiés à l'estimation des paramètres du modèle. On parle d'estimation des paramètres lorsqu'on travaille dans un cadre « *non supervisé* » [37]. Cet algorithme est nommé '*Expectation-Modification EM* '. Si ce dernier est associé à une chaîne de Markov, il est souvent appelé par l'algorithme de '*Baum-Welch*', ou encore l'algorithme '*forward-Backward*'.

Ainsi avec un *MMC* ayant K classes, la loi à priori de X est caractérisée par K^2 inconnus (paramètres à estimer). Aussi le choix des gaussiennes ajoute un nombre de $2K$ inconnus (à savoir, moyenne et variance de chacune des K densités).

Ces paramètres varient selon l'application. De plus, il ne faut pas oublier les π_i de la chaîne. Notons par θ, l'ensemble de ces paramètres. Le problème alors, est celui de l'estimation de θ à partir de la seule mesure existante y constituant la réalisation de l'observation Y.

3.6.2.1 : Algorithme de *Baum-Welch* pour une chaîne de Markov cachée

Soit θ l'ensemble des paramètres à estimer dont dépend de la distribution jointe $P(X,Y/\theta)$. On cherche à maximiser la fonction de vraisemblance $P(y/\theta)$, disons :

$$\hat{\theta} = \arg\max_{\theta} P(y/\theta) \quad \text{(3.32)}$$

Cette procédure s'appuie sur un calcul itératif permettant de définir à partir d'une valeur initiale $\theta^{(0)}$ une suite $\left(\theta^{(q)}\right)_{1\le q\le Q}$ (avec Q le nombre total d'itérations), dont $P(y/\theta)$ est une fonction croissante [31, 35, 45]. A chaque itération q, on effectue deux sous-procédures essentielles :

- Etape *E*, pour '*Expectation*' : dans laquelle on calcule l'espérance.
- Etape *M*, pour '*Maximisation*' : dans laquelle on établie une mise à jours des paramètres à travers la maximisation de cette espérance.

Pour un *MMC*, le calcul est totalement attaché aux deux probabilités $\xi_t(i)$ et $\psi_t(i,j)$. Ainsi, il est possible d'extraire les formules de réestimation des paramètres à l'itération suivante $(q+1)$. Elles sont sous forme de :

- Paramètres propres de la chaîne de Markov :

$$\pi_i^{(q+1)} = \frac{1}{T} \cdot \sum_{1 \le t \le T} \xi_t^{(q)}(i) \quad \text{(3.33)}$$

Et :

$$a_{ij}^{(q+1)} = \frac{\sum_{1 \le t \le T} \psi_t^{(q)}(i, j)}{\sum_{1 \le t \le T} \xi_t^{(q)}(i)} \quad \text{(3.34)}$$

- Paramètres caractérisant les gaussiennes :

$$\mu_i^{(q+1)} = \frac{\sum_{1 \le t \le T} y_t \cdot \xi_t^{(q)}(i)}{\sum_{1 \le t \le T} \xi_t^{(q)}(i)} \quad \text{(3.35)}$$

et :

$$\left(\sigma_i^2\right)^{(q+1)} = \frac{\sum_{1 \le t \le T} \left(y_t - \mu_i^{(q+1)}\right)^2 \cdot \xi_t^{(q)}(i)}{\sum_{1 \le t \le T} \xi_t^{(q)}(i)} \quad \text{(3.36)}$$

A la fin, on peut résumer l'algorithme de '*Baum-Welch*' par le tableau **Tab (3.4)** [37].

remarques ↑ :

- l'utilisation de cet algorithme nécessite une initialisation préalable de ses paramètres. Cela veut dire que l'estimée finale dépend fortement de l'initialisation. Cette dernière a alors, une grande influence sur la qualité de l'estimation, ainsi que le temps de convergence de l'algorithme *EM*.
- La convergence donc, n'est pas garantie. Par conséquent, il existe un risque que l'algorithme soit piégé dans un maximum local (non absolu) [31, 35, 37].

Dans le but de résoudre ces problèmes annoncés par les remarques précédentes, un certain nombre de solutions sont proposés. Parmi elles, on introduit des procédures

préliminaires permettant d'avoir une bonne initialisation pour l'algorithme *EM*. A titre d'exemple, on cite l'algorithme des *'K-means'* [37].

D'autres sont allés à augmenter la performance de cet algorithme *EM*, en introduisant des variantes diverses dont lesquelles on ajoute généralement d'autres étapes (étape d'échantionnage, par exemple) à l'algorithme initial. Nous citons et sans explications :

- Le '*SEM'* pour '*Stockastic EM'*.
- Le '*SAEM'* pour '*Stockastic Approximation EM'*.
- Le '*MCEM'* pour '*Monte-Carlo EM'*.
- … et l'algorithme '*ICE'* pour '*Iterative Conditional Estimation'*, introduit par Pieczynski et choisi pour notre travail.

Tab (3.4) : *Algorithme EM*

- Initialisation des paramètres :

  $\pi_i^{(0)}, a_{ij}^{(0)}, f_i^{(0)}$ *pour* $:1 \le i, j \le K$

- A chaque itération q :

  a- Etape E : calcul des probabilités suivantes :

  - $\alpha_t(i)$ et $\beta_t(i)$

  - Déduction des $\psi_t^{(q)}(i,j)$ et $\xi_t^{(q)}(i)$ à partir de $\alpha_t(i)$ et $\beta_t(i)$ calculées sur la base des paramètres $a_{ij}^{(q1)}, f_i^{(q)}$

  b- Etape M : calcul des paramètres $\pi_i^{(q+1)}, a_{ij}^{(q+1)}, f_i^{(q+1)}$:

$$\pi_i^{(q+1)} = \frac{1}{T} \cdot \sum_{1 \le t \le T} \xi_t^{(q)}(i)$$

$$a_{ij}^{(q+1)} = \frac{\sum_{1 \le t \le T} \psi_t^{(q)}(i,j)}{\sum_{1 \le t \le T} \xi_t^{(q)}(i)}$$

$$\mu_i^{(q+1)} = \frac{\sum_{1 \le t \le T} y_t \cdot \xi_t^{(q)}(i)}{\sum_{1 \le t \le T} \xi_t^{(q)}(i)}$$

$$\left(\sigma_i^2\right)^{(q+1)} = \frac{\sum_{1 \le t \le T} \left(y_t - \mu_i^{(q+1)}\right)^2 \cdot \xi_t^{(q)}(i)}{\sum_{1 \le t \le T} \xi_t^{(q)}(i)}$$

3.7 : Exemples de segmentation d'images par modèles *MMCs*

Pour réaliser la segmentation par les Modèles de Markov Cachés *MMCs*, nous considérons deux images. Une première intitulée « *AB* », contient les deux lettres *A* et *B* écrites en noir sur un fond blanc. L'autre appelée « *Voiture* », illustre bien entendue une voiture. C'est une image réelle. Les deux sont de taille 32x32 pixels, et sont traitées en considérant la caractéristique intensité. Signalons que la procédure de segmentation est réalisée en utilisant l'algorithme *ICE* qui sera suffisamment détaillé dans le chapitre suivant

En fin une petite comparaison entre l'algorithme *ICE_MPM* et ceux de *EM_MPM* et *EM_Viterbi,* effectuée sur trois image déférentes, dans le but de mettre en évidence son efficacité dans le domaine de segmentation.

remarque ↑:

Dans les exemples suivants, la notation est comme suit :

P : le vecteur initial de la chaîne de Markov.

A : la matrice de transition de la chaîne de Markov.

Mu : vecteur de moyennes des deux classes

Seg : vecteur de variances des deux classes.

Q : nombre d'itérations de l'algorithme *ICE*

EM_MPM : l'algorithme *EM* suivit de l'algorithme *MPM*

EM_Viterbi : l'algorithme *EM* suivit de l'algorithme de *Viterbi.*

Exemple 01 : Image *« AB »*

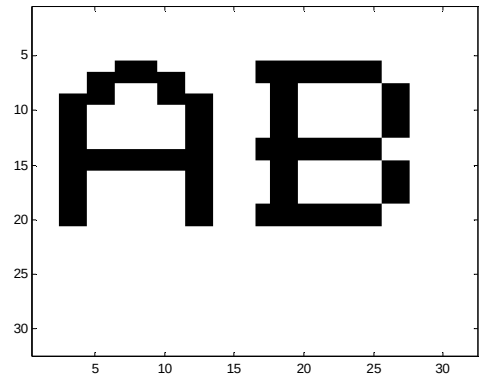
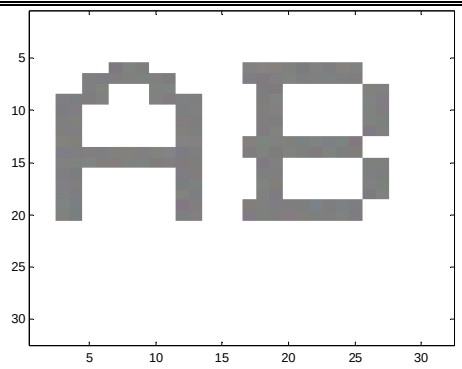


Fig (3.4) *: Image originale « AB » de taille : 32x32 pixels*

a : ***Segmentation (1-1)***



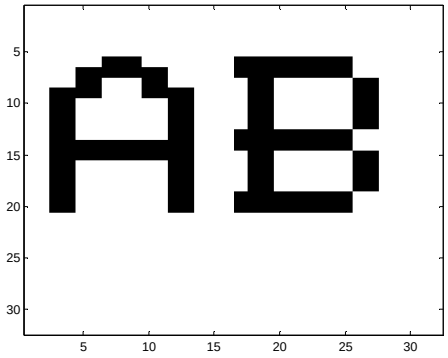


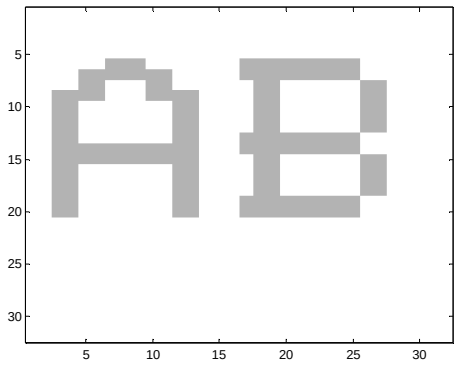
Image originale + bruit gaussien de :
moyenne : 0.5
variance : 0.0001

```
Q=1;
Paramètres de MMc :
 P = 0.4 0.6
 A =0.3 0.7
     0.2 0.8
 Seg = 40 40
 Mu =  20 50
```

Fig (3.5) : ***Segmentation (1-1) : Mu=20 50; Seg=40 40;***

l'image « AB »+ bruit gaussien de moyenne : 0.5 et de variance : 0.0001

b : ***Segmentation (1-2)***



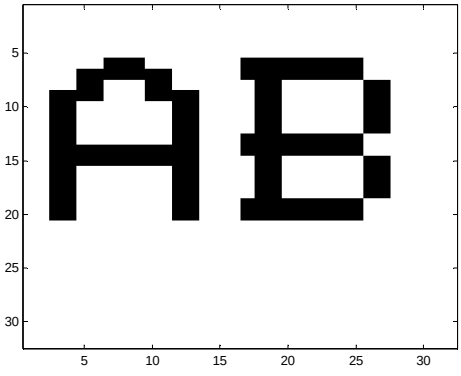


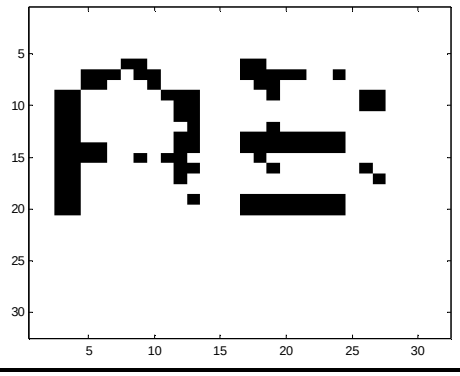
Image originale + bruit gaussien de :
moyenne : 0.7
variance : 0

```
Q=1 ;
Paramètres de MMc:
P = 0.4 0.6
A =0.3 0.7
    0.2 0.8
Seg = 10 10
Mu =  200 250
```

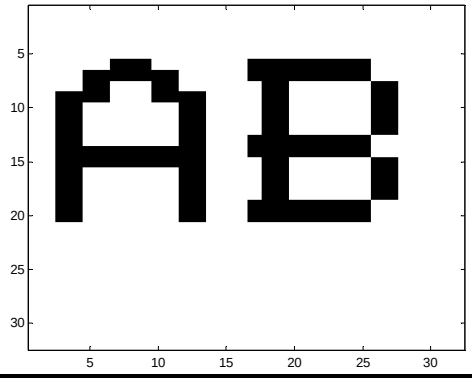
Fig (3.6) : ***Segmentation (1-2) : Mu=200 250; Seg=10 10;***

l'image « AB »+ bruit gaussien de moyenne : 0.7 et de variance : 0

c : ***Segmentation (1-3)***



Q=1 ;
Paramètres de MMc :
P = 0.4 0.6
A =0.3 0.7
0.2 0.8
Seg = 10 10
Mu = 90 130

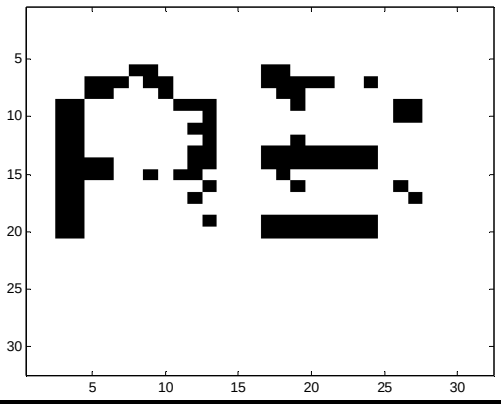


Q=3 ;
Paramètres de MMc :
P = 0.4 0.6
A =0.3 0.7
0.2 0.8
Seg = 10 10
Mu = 90 130

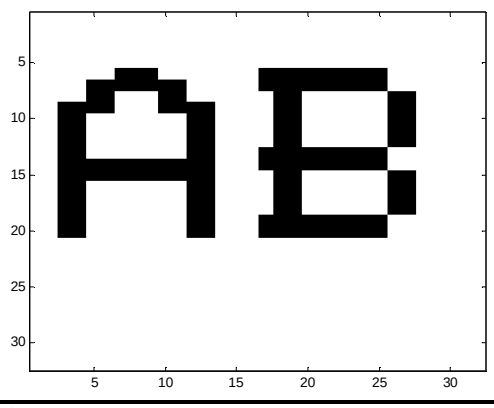
Fig (3.7) *: Segmentation (1-3) : Mu=90 130; Seg = 10 10;*

l'image « AB »+ bruit gaussien de moyenne : 0.7 et de variance : 0

d: ***Segmentation (1-4)***



Q=1 ;
Paramètres de MMc :
P = 0.4 0.6
A =0.3 0.7
0.2 0.8
Seg = 100 100
Mu = 200 250



Q=2 ;
Paramètres de MMc :
P = 0.4 0.6
A =0.3 0.7
0.2 0.8
Seg = 100 100
Mu = 200 250

Fig (3.8) : *Segmentation (1-4) : Mu=200 250; Seg=100 100;*

l'image « AB »+ bruit gaussien de moyenne : 0.7 et de variance : 0

e: ***Segmentation (1-5)***

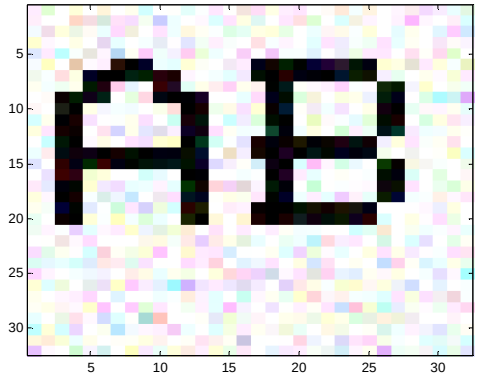
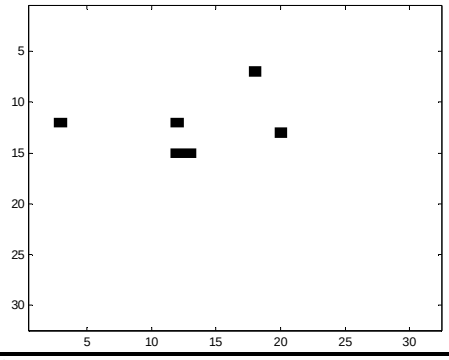
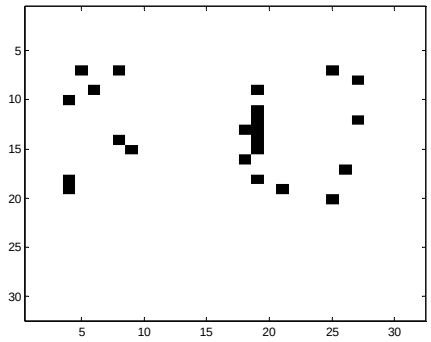


Image « AB » + bruit gaussien de :

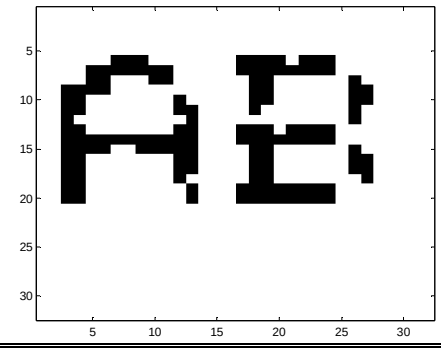
moyenne : 0 et de variance : 0.01



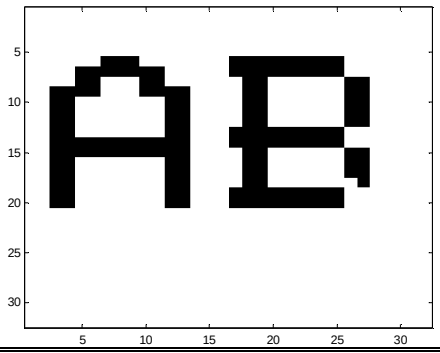
```
Q=1 ;
Paramètres de MMc :
P =0.4 0.6
A =0.3 0.7
   0.2 0.8
Mu = 100 230
Seg =100 400
```



```
Q=2 ;
Paramètres de MMc :
P = 0.4 0.6
A = 0.3 0.7
    0.2 0.8
Mu =  100 230
Seg = 100 400
```



```
Q=3 ;
Paramètres de l'HMM :
P = 0.4 0.6
A = 0.3 0.7
    0.2 0.8
Mu =  100 230
Seg = 100 400
```



```
Q=4 ;
Paramètres de l'HMM :
P = 0.4 0.6
A =0.3 0.7
   0.2 0.8
Mu =  100 230
Seg = 100 400
```

Fig (3.9) : ***Segmentation (1-5) : Mu=100 230; Seg=100 400;***

l'image « AB »+ bruit gaussien de moyenne : 0 et de variance : 0.01

Exemple 02 : Image « Voiture »

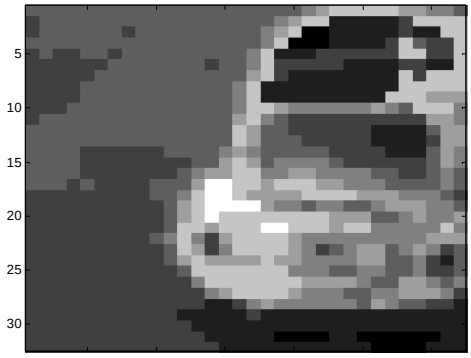
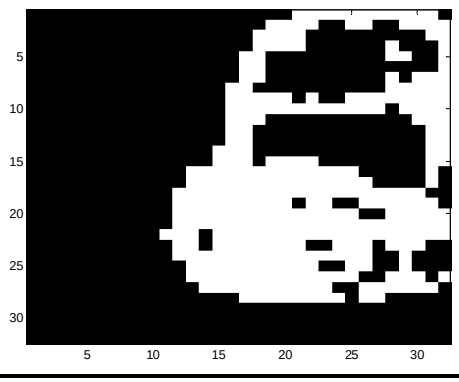
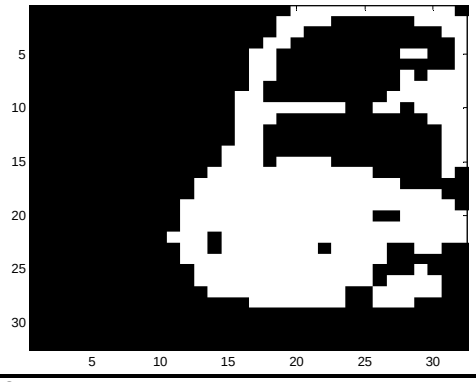


Fig (3.10) : ***Image originale « Voiture » de taille : 32x32 pixels***

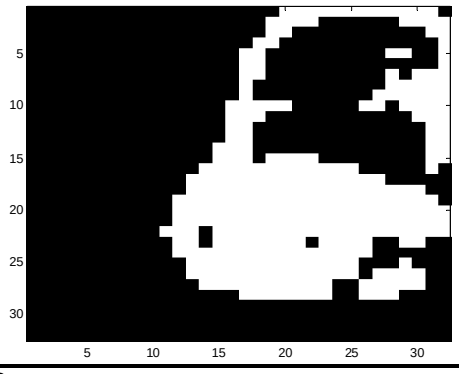
a : ***Segmentation (2-1)***



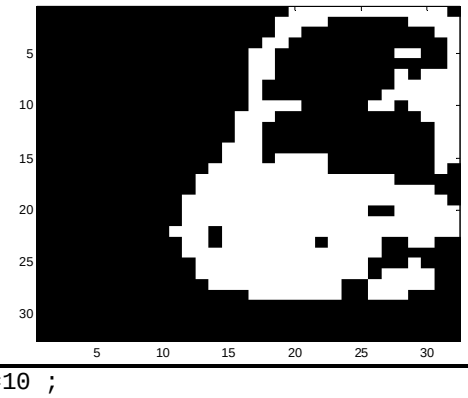
```
Q=1 ;
Paramètres de MMc :
P = 0.4 0.6
A =0.3 0.7
    0.2 0.8
Mu =  150 100
Seg = 10 10
```



```
Q=2 ;
Paramètres de MMc :
P =0.4 0.6
A =0.3 0.7
    0.2 0.8
Mu =  150 100
Seg = 10 10
```



```
Q=3 ;
Paramètres de MMc :
P = 0.4 0.6
A =0.3 0.7
    0.2 0.8
Mu =  150 100
Seg = 10 10
```



```
Q=10 ;
Paramètres de MMc :
P = 0.4 0.6
A =0.3 0.7
    0.2 0.8
Mu =  150 100
Seg = 10 10
```

Fig (3.11) : ***Segmentation (2-1) : Image originale « Voiture »***

Mu=150 100; Seg=10 10;

b : ***Segmentation (2-2)***

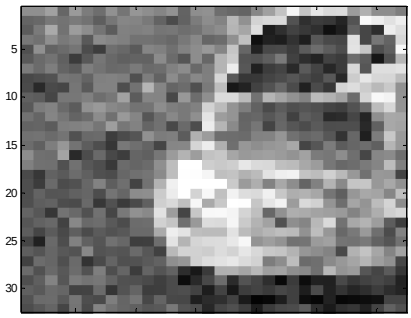
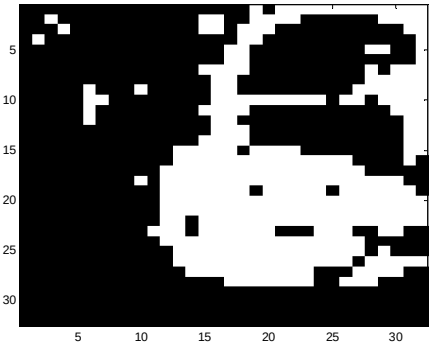
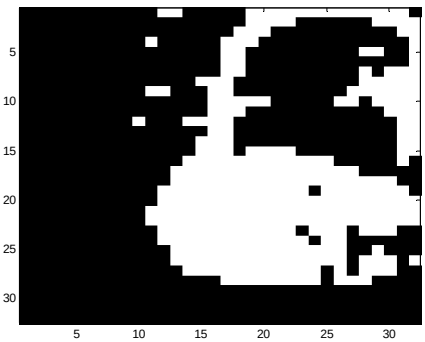
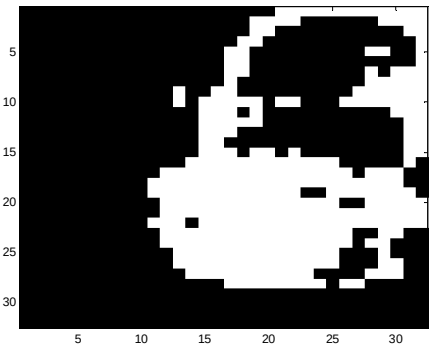
<table>
<tr><td>  </td><td>  </td></tr>
<tr><td>Image originale « Voiture » + bruit gaussien de :
moyenne : 0.10
et variance : 0.01</td><td>Q=1;
Paramètres de l’HMM :
P = 0.4 0.6
A =0.3 0.7
0.2 0.8
Mu = 150 100 Seg = 10 10</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Q=2;
Paramètres de l’HMM :
P = 0.4 0.6
A =0.3 0.7
0.2 0.8
Mu = 150 100 Seg = 10 10</td><td>Q=3;
Paramètres de l’HMM :
P = 0.4 0.6
A =0.3 0.7
0.2 0.8
Mu = 150 100 Seg = 10 10</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Q=4;
Paramètres de l’HMM :
P = 0.4 0.6
A =0.3 0.7
0.2 0.8
Mu = 150 100 Seg = 10 10</td><td>Q=10;
Paramètres de l’HMM :
P = 0.4 0.6
A =0.3 0.7
0.2 0.8
Mu = 150 100 Seg = 10 10</td></tr>
</table>

Fig (3.12) : ***Segmentation (2-2) : Mu=150 100; Seg=10 10; l’image « Voiture »+bruit de moyenne :0.1 et variance :0.01***

Les moyennes et les variances des densités obtenues à partir de la segmentation (2-1) sont groupées dans la table suivante :

Mu	Seg
100 150	10 10
(1.0e+002) *	(1.0e+002) *
0.81469011264081 1.51237022222222	1.13483464185556 2.02168689837636
0.71394992223951 1.39672152230972	1.35655653282335 1.43225738445743
0.67361455604076 1.56809347181009	1.24762358626140 1.39620402944086
0.68845426136364 1.58296531250000	1.40552072819750 1.41793189627288
0.69723388203018 1.63707525423729	1.37467057424512 1.48837437518364
0.67945607344633 1.61444873417722	1.44483970600880 1.47962020306240
0.69284606896552 1.63514147157191	1.41844284038460 1.47526814035324
0.69211980742779 1.64326464646465	1.43690073220938 1.49137132082870
0.68569775910365 1.61816935483871	1.45736532355465 1.47234488816402
0.68821289875174 1.63372673267327	1.43058600104375 1.47190534844355

Tab (3.5) ***: Evolution des moyennes et des variances lors de la segmentation (2-1)***

Nous pouvons représenté l'évolution de ces deux caractéristiques en fonction du nombre d'itération de l'algorithme *ICE*, par les courbes ci après.

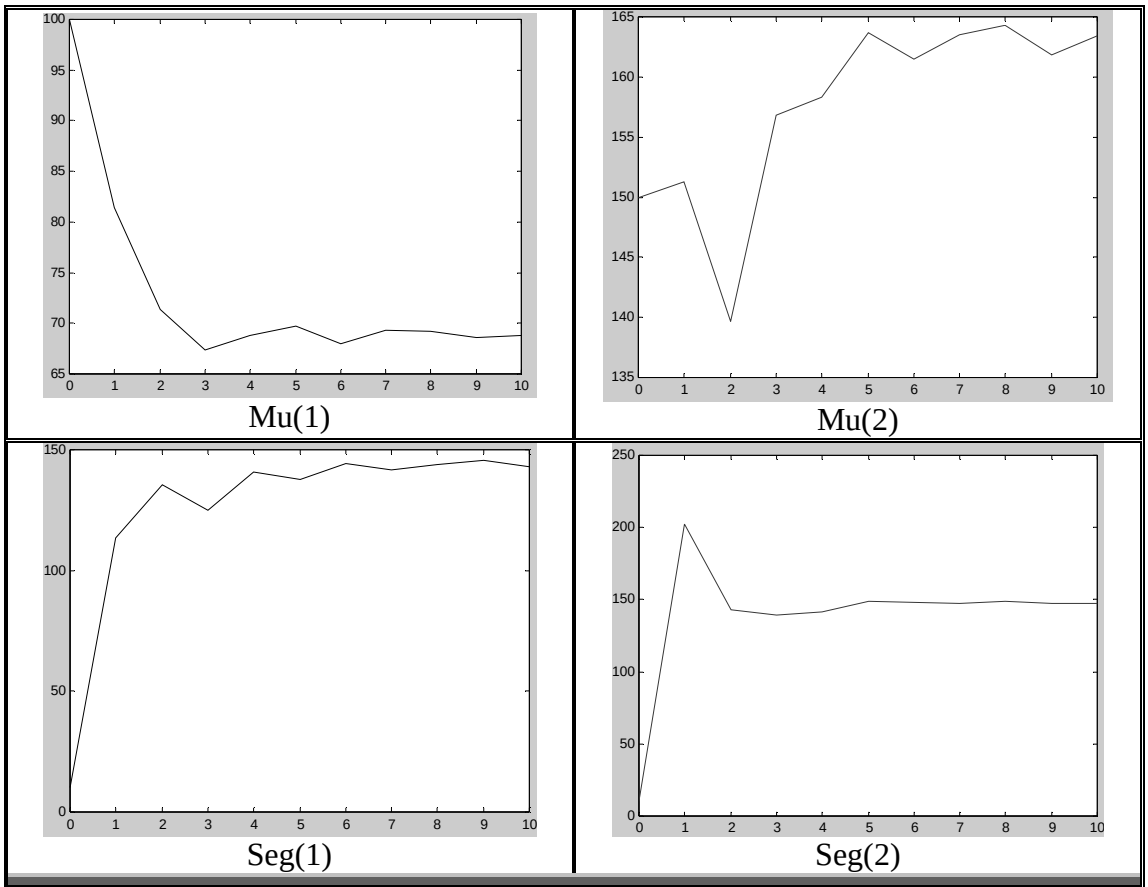


Fig (3.13) ***: Evolution des moyennes et des variances lors de la segmentation (2-1) en fonction du nombre d'itérations***

Ceux de la segmentation (2-2) sont montrés dans la table qui suit.

Mu	Seg
100 150	10 10
1.0e+002 *	1.0e+002 *
1.04120326975477 1.65032965517241	1.16081914874951 1.76629626602654
0.91603602693603 1.62491488372093	1.45585688371490 1.61664061058570
0.89619308681672 1.70499179104478	1.45214334007640 1.53403381166925
0.91103681818182 1.76251236263736	1.52967801824564 1.54626935162353
0.91837405797101 1.82383443113772	1.52135293186699 1.64695323564310
0.92051287988423 1.82211531531532	1.58839248274974 1.65086050150863
0.92712661870504 1.81910577507599	1.57504134496070 1.66226917202392
0.93397064606742 1.85208878205128	1.53051283444834 1.72258982887368
0.92941645207439 1.82515907692308	1.59665544021406 1.68712266298707
0.92812751773050 1.84485548589342	1.55526252805495 1.70164508398975

Tab (3.6) *: Evolution des moyennes et des variances lors de la segmentation (2-2)*

Et de même, ces derniers sont présentés en fonction du nombre d'itération de l'algorithme *ICE*, par les graphes de la figure Fig (3.14) suivante.

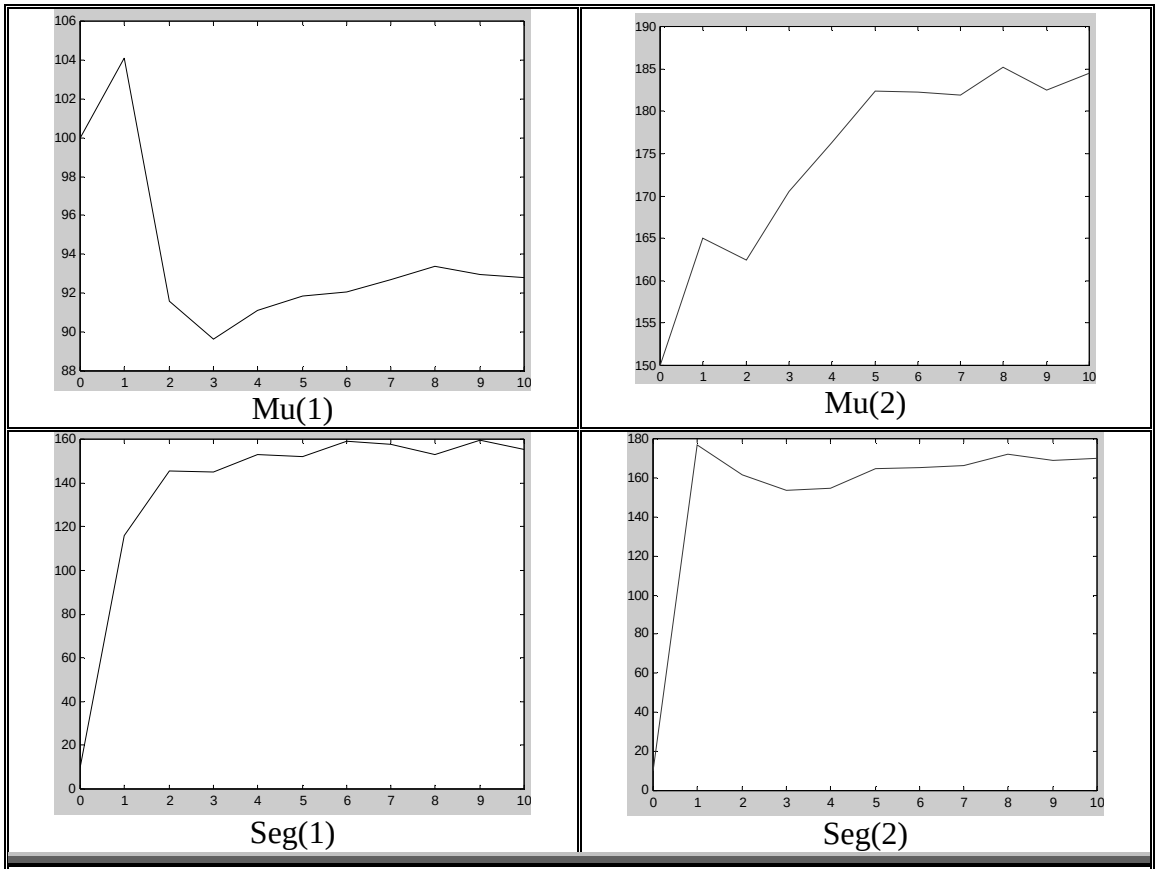


Fig (3.14) *: Evolution des moyennes et des variances lors de la segmentation (2-2) en fonction du nombre d'itérations*

Effet du vecteur P

Image originale « *Voiture* » avec les paramètres de *MMC* :

A =0.3 0.7

0.2 0.8

Mu = 150 100

Seg = 10 10

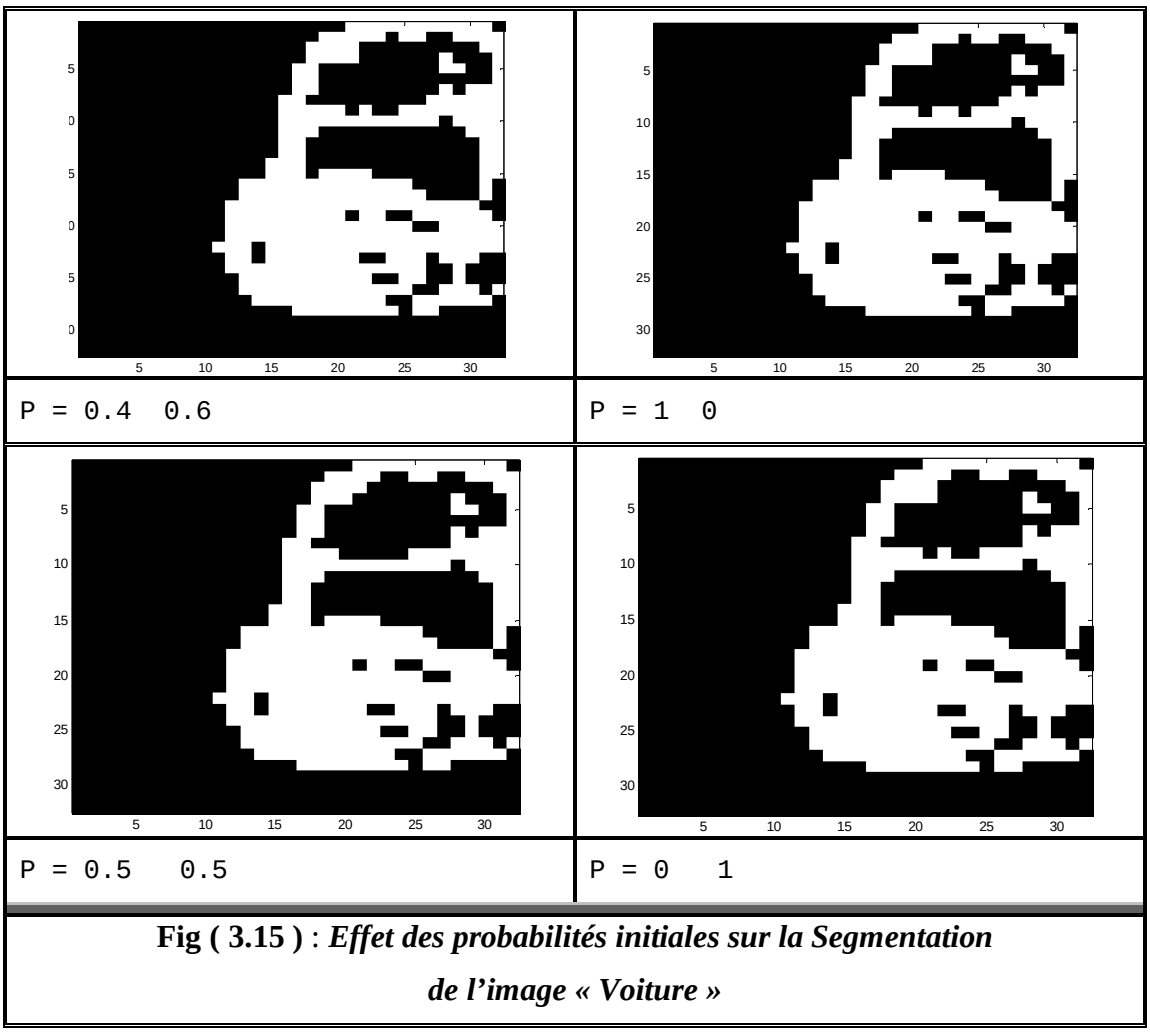


Fig (3.15) : ***Effet des probabilités initiales sur la Segmentation de l'image « Voiture »***

Effet de la matrice A

Image originale « Voiture » avec les paramètres de MMc :
Q=1 ; P =0.5 0.5;

Mu = 150 100;

Seg = 10 10;

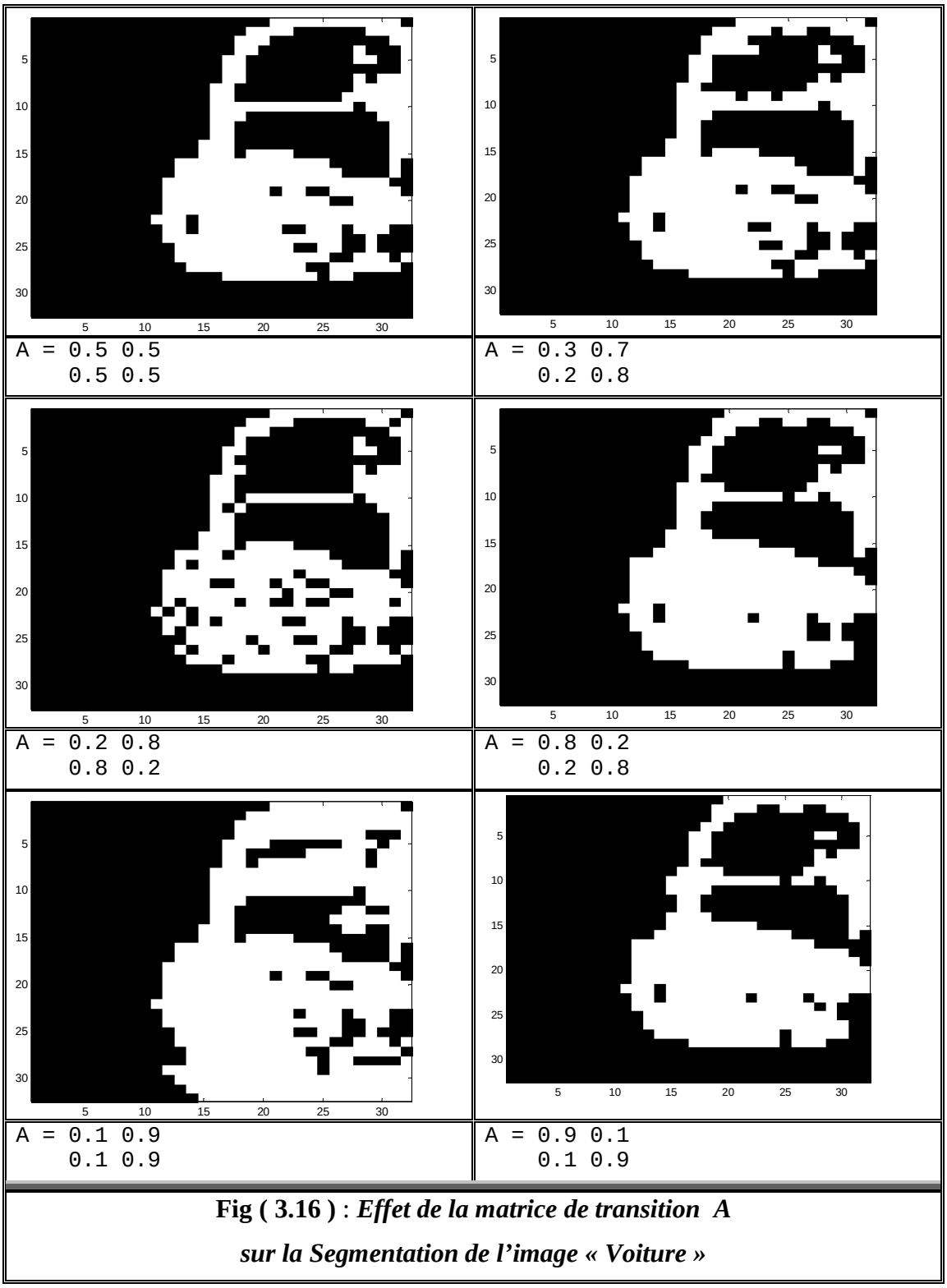


Fig (3.16) : ***Effet de la matrice de transition A sur la Segmentation de l'image « Voiture »***

Aller à : Fichier Segment1

3.8 : Conclusion

Ce chapitre a été consacré à la définition des déférentes probabilités nécessaires à la modélisation markovienne par les Chaînes de Markov Cachées. Cependant l'utilisation directe de ces probabilités est impossible.

Cela nous a conduit donc, à chercher des transformations et des algorithmes bien définis capables de diminuer la complexité et l'accès de calcul accompagnant ces modèles.

Alors, les chaînes de Markov cachées modélisent l'image par des parcours de la forme *d'Hilbert_Piano,* présentant une supériorité par rapport aux autres transformations citées dans ce chapitre.

Ce chapitre se termine par une segmentation d'images à base d' *MMCs,* dont laquelle sont montrées l'efficacité et la souplesse des Chaînes de Markov Cachées, dans ce domaine.

Détection d'objet mobile par les Modèles de Markov Cachés

4.1 : Introduction

La détection d'objet mobile en utilisant les modèles de Markov cachés ; *MMCs,* a pour but final de classer les pixels de chaque image incidente en deux régions : « *objet mobile* » et « *Background* ». Cela revient à déterminer la réalisation x des états cachés X , en partant de la seule information disponible représentée par la réalisation y de Y.

Dans ce cas, on travaille dans un contexte dit « *non supervisé* ». Cela veut dire que les paramètres de ce modèle, sont estimés uniquement à partir de la réalisation y de l'observation Y, puisque la réalisation de X est totalement inconnue. Rappelons que ces paramètres s'arrangent en deux groupes : ceux propres à la chaîne de Markov cachée et ceux associés aux classes. Une fois ces paramètres estimés, on procède alors, à la détection.

4.2. Estimation des paramètres

Fréquemment, on ne dispose d'aucune connaissance a priori sur les classes. En d'autres termes, nous savons plus quels pixels représentatifs pour chacune. Leurs paramètres donc doivent être estimés.

Pour nous, l'attache aux données (probabilités conditionnelles) est une distribution supposée « *gaussienne* » dont ses paramètres révèlent les moyennes et les variances des classes. Nous supposons aussi, que les données mesurables se présentent par leurs niveaux de gris (Intensités). La chaîne de Markov est supposée ergodique, basculant entre deux états F et B. La première représente l'objet mobile et l'autre définisse le *Background,* **Fig (4.1)**.

Comme signaler ci avant, il existe plusieurs méthodes dédiées à l'estimation des paramètres. Cependant dans cette partie, on s'intéresse en particulier à l'algorithme connu sous le nom de '*ICE : Iterative Conditionnal Estimation*'.

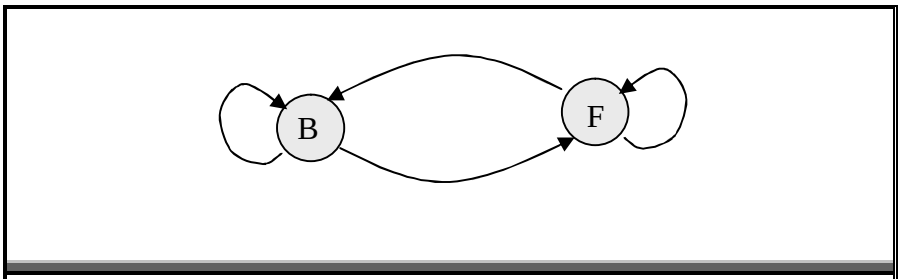


Fig (4.1) *: Graphe d'états modèle MMC choisi*

4.2.1 : Principe de l'algorithme *'ICE'*

L'algorithme '*ICE*' introduit par *Pieczynski* [37], représente une méthode générale d'estimation des paramètres, acceptant toute forme de distributions. Il ressemble à l'algorithme *EM* [35]. Il s'agit encore d'un algorithme itératif.

On considère un estimateur $\hat{\theta}(X,Y)$ de l'ensemble θ des paramètres du modèle à estimer, basé sur les données complètes (X,Y), et on veut l'approcher par une fonction de l'observation Y, qui représente la seul donnée disponible. La meilleure approximation de cet estimateur au sens de l'erreur quadratique moyenne n'est autre que l'espérance conditionnelle par rapport à *Y,* donnée par $E\left[\hat{\theta}(X,Y)/Y=y\right]$. Cette espérance dépend du paramètre θ, et n'est pas calculable explicitement [35, 37]. Avec une démarche itérative, dont q le nombre d'itérations, on calcule la valeur suivante des paramètres en fonction de leur valeur courante, et on écrit :

$$\theta^{(q+1)} = E\left[\hat{\theta}(X,Y)/Y = y, \theta^{(q)}\right] \quad \text{(4.1)}$$

Ainsi la procédure de l'algorithme '*ICE*' est la suivante [35, 37] :

- Initialisation de θ, soit $\theta^{(0)}$.
- Calcul des paramètres suivants à partir de la donnée mesurée y et des paramètres courants $\theta^{(q)}$ selon:

$$\theta^{(q+1)} = E\left[\hat{\theta}(X,Y)/Y = y, \theta^{(q)}\right]$$

remarque ↑:

Lorsque l'espérance de (4.1) n'est pas explicitement calculable, on peut l'estimer par des tirages aléatoires selon la loi conditionnelle à *Y*, [35, 37, 45]. On simule ainsi, *M* réalisations de X notées $x^1, x^2, ..., x^M$, selon la loi conditionnelle à *Y*, et basées sur les paramètres $\theta^{(q)}$. La valeur $\theta^{(q+1)}$ des paramètres à l'itération $q+1$, sera exprimée par la moyenne tirée de ces *M* échantillons [37] . Elle s'écrit par:

$$\theta^{(q+1)} = \frac{1}{M}\left[\hat{\theta}(x^1, y) + \hat{\theta}(x^2, y) + ... + \hat{\theta}(x^M, y)\right]$$

4.2.2 : L'algorithme '*ICE*' pour une chaîne de Markov cachée

L'application de l'algorithme *ICE* au stade de l'apprentissage associé aux Chaînes de Markov Cachées, permet d'estimer les paramètres définissant la loi de *X*, ainsi que ceux du bruit [37].

a / Ré-estimation des paramètres de la loi de *X*

Le calcul de ces paramètres est basé essentiellement sur les probabilités c_{ij}. Conformément au principe général *d'ICE*, on considère que X est observable. Les quantités c_{ij} donc, peuvent être estimées selon :

$$\hat{c}_{ij}(X) = \frac{1}{T-1}\sum_{t=1,T-1} 1_{|X_t=\omega_i, X_{t+1}=\omega_j|} \quad \text{(4.2)}$$

Celle-ci représente la fréquence empirique d'aller de l'état ω_i à l'état ω_j pour une chaîne de longueur T. La valeur suivante de c_{ij} est donnée par l'espérance conditionnelle de $\hat{c}_{ij}$, comme suit :

$$c_{ij}^{(q+1)} = E\left[\hat{c}_{ij}(X)/Y, c_{ij}^{(q)}\right] \quad \text{(4.3)}$$

Elle est écrite en fonction de c_{ij} par :

$$c_{ij}^{(q+1)} = \frac{1}{T-1}\sum_{t=1,T-1} P^{(q)}\left(X_t = \omega_i, X_{t+1} = \omega_j / Y = y\right) \quad \text{(4.4)}$$

avec $P^{(q)}\left(X_t = \omega_i, X_{t+1} = \omega_j / Y = y\right) = c_{ij}^{(q)}$.

Selon l'équation (3.5), les éléments $a_{ij}^{(q+1)}$ de la matrice de transition à l'itération $(q+1)$ seront donnés par :

$$a_{ij}^{(q+1)} = \frac{c_{ij}^{(q)}}{\sum_{1\le j\le K} c_{ij}^{(q)}} = \frac{\sum_{t=1,T-1} P^{(q)}\left(X_t = \omega_i, X_{t+1} = \omega_j / Y = y\right)}{\sum_{j=1,K}\sum_{t=1,T-1} P^{(q)}\left(X_t = \omega_i, X_{t+1} = \omega_j / Y = y\right)}$$

Donc à l'itération $(q+1)$, les paramètres de la chaîne s'écrient comme suit:

$$a_{ij}^{(q+1)} = \frac{\sum_{1\le t\le T-1}\psi_t^{(q)}(i,j)}{\sum_{1\le t\le T-1}\xi_t^{(q)}(i)} \quad \text{(4.5)}$$

Et :

$$\pi_i^{(q+1)} = \sum_{1\le t\le T-1}\xi_t^{(q)}(i) \quad \text{(4.6)}$$

b / Ré-estimation des paramètres de la lois de Y_t conditionnelle à X_t :

Considérant le cas gaussien, ces paramètres représentent les moyennes μ_i et les variances σ_i. Elles sont données par *ICE* :

$$\overset{\wedge}{\mu}_i^{(q+1),m} = \frac{\sum_{1\le t\le T} y_t . 1_{\left|X_t^{(q.m)}=\omega_i\right|}}{\sum_{1\le t\le T} 1_{\left|X_t^{(q.m)}=\omega_i\right|}} \quad \text{(4.7)}$$

Et :

$$\left(\overset{\wedge}{\sigma}_i^{(q+1),m}\right)^2 = \frac{\sum_{1\le t\le T}\left(y_t - \overset{\wedge}{\mu}_i^{(q+1),m}\right)^2 . 1_{\left|X_t^{(q),m}=\omega_i\right|}}{\sum_{1\le t\le T} 1_{\left|X_t^{(q),m}=\omega_i\right|}} \quad \text{(4.8)}$$

Ces formules sont dérivées d'une procédure d'échantillonnage [35, 45], dans laquelle on simule M des réalisations de X indicée par m, selon la loi a posteriori donnée par l'équation (3.19). L'estimée final des paramètres des densités est donnée par la moyenne calculées de ces échantillons pour chaque caractéristique.

Finalement, la procédure *ICE* pour l'estimation des paramètres θ, traverse les étapes suivantes, tout en partant des paramètres initiaux $\pi_i^{(0)}$, $a_{ij}^{(0)}$ et des données $f_i^{(0)}$ [45].

Pour chaque itération *q* de l'algorithme *ICE*, on se base sur les probabilités *Forward* et *Backward* selon la démarche ci après :

Pour chaque élément *t* de la chaîne, et pour chaque classe ω_i possible, et en utilisant les valeurs des paramètres $\theta^{(q-1)}$ à l'itération $(q-1)$, on calcule :

- Les probabilités *Forward*, $\alpha_t^{(q)}(i)$.
- Les probabilités *Backward*, $\beta_t^{(q)}(i)$.
- Les probabilités a posteriori marginales , $\xi_t^{(q)}(i)$.

Ainsi, on peut calculer :

- Les nouvelles probabilités jointes conditionnelles , $\psi_t^{(q)}(i,j)$.
- Les nouveaux éléments de la matrice de transition stationnaire, $a_{ij}^{(q)}$ tel que:

$$a_{ij}^{(q)} = \frac{\sum_{1\le t\le T-1} \psi_t^{(q)}(i,j)}{\sum_{1\le t\le T-1} \xi_t^{(q)}(i)} \quad \text{(4.9)}$$

- et les nouvelles probabilités initiales $\pi_i^{(q)}$:

$$\pi_i^{(q)} = \sum_{1\le t\le T-1} \xi_t^{(q)}(i) \quad \text{(4.10)}$$

On calcule une série de réalisations a posteriori à l'aide de l'équation (3.19), basées sur $\alpha_t^{(q)}(i)$, $\beta_t^{(q)}(i)$ et $f_i^{(q-1)}$. Pour chaque réalisation indicée par *m*, nous estimons les paramètres de chaque classe (sa moyenne et sa variance) [45].

$$\overset{\wedge}{\mu}_i^{(q),m} = \frac{\sum_{1\le t\le T} y_t . 1_{\left|X_t^{(q),m}=\omega_i\right|}}{\sum_{1\le t\le T} 1_{\left|X_t^{(q),m}=\omega_i\right|}} \quad \text{(4.11)}$$

Et :

$$\left(\overset{\wedge}{\sigma}_i^{(q),m}\right)^2 = \frac{\sum_{1\le t\le T} \left(y_t - \overset{\wedge}{\mu}_i^{(q),m}\right)^2 . 1_{\left|X_t^{(q),m}=\omega_i\right|}}{\sum_{1\le t\le T} 1_{\left|X_t^{(q),m}=\omega_i\right|}} \quad \text{(4.12)}$$

Ces deux derniers sont moyennés dans le but d'obtenir les paramètres $\theta^{(q)}$ à l'itération (q) comme suit :

$$\overset{\wedge}{\mu}_i^{(q)} = \frac{1}{M} \sum_{1\le m\le M} \overset{\wedge}{\mu}_i^{(q),m} \quad \text{(4.13)}$$

Et :

$$\left(\overset{\wedge}{\sigma}_i^{(q)}\right)^2 = \frac{1}{M} \sum_{1\le m\le M} \left(\overset{\wedge}{\sigma}_i^{(q),m}\right)^2 \quad \text{(4.14)}$$

Avec M , le nombre totale de réalisations simulées.

L'algorithme *ICE* complet est donné par la table **Tab (4.1)** .

remarque ↑:

Nous allons limiter le nombre de réalisations a posteriori à un (*01*) pour chaque itération de *ICE*.

Tab (4.1) : ***Algorithme 'ICE' pour une Chaîne de Markov Cachée***

- Initialisation des paramètres : $\pi_i^{(0)}$, $a_{ij}^{(0)}$ et des données $f_i^{(0)}$.
- Répéter jusqu'à convergence :

1/ Forward

** Initialisation*

$$\alpha_1^{(q)}(i) = \frac{\pi_i^{(q)} \cdot f_i^{(q)}(y_1)}{\sum_{1 \le j \le K} \pi_j^{(q)} \cdot f_j^{(q)}(y_1)} \qquad pour \;\; : 1 \le i \le K$$

** Récurrence*

$$\alpha_t^{(q)}(i) = \frac{f_i^{(q)}{}_i(y_t) \cdot \sum_{1 \le j \le K} \alpha_{t-1}^{(q)}(j) \cdot a_{ij}^{(q)}}{\sum_{1 \le l \le K} f_l^{(q)}(y_t) \cdot \sum_{1 \le j \le K} \alpha_{t-1}^{(q)}(j) \cdot a_{jl}^{(q)}} \qquad pour \;\; : 1 \le i \le K, \;\; t = 2,...,T$$

2/ Backward

** Initialisation*

- $\beta_T^{(q)}(i) = 1 \qquad pour \;\; : 1 \le i \le K$

- $\xi_T^{(q)}(i) = \frac{\alpha_T^{(q)}(i)}{\sum_{1 \le j \le K} \alpha_T^{(q)}(j)} \qquad pour \;\; : 1 \le i \le K$

- $\psi_T^{(q)}(i, j) = \frac{\alpha_{T-1}^{(q)}(i) \cdot a_{ij}^{(q)} \cdot f_j^{(q)}(y_T) \cdot}{\sum_{1 \le l \le K} f_l^{(q)}(y_T) \cdot \sum_{1 \le j \le K} \alpha_T^{(q)}(j) \cdot a_{jl}^{(q)}} \qquad pour : 1 \le i \le K, 1 \le j \le K$

** Récurrence pour* $t = T-1,...,1$

- $\beta_t^{(q)}(i) = \frac{\sum_{1 \le j \le K} a_{ij}^{(q)} \cdot f_j^{(q)}(y_{t+1}) \cdot \beta_{t+1}^{(q)}(j)}{\sum_{1 \le l \le K} f_l^{(q)}(y_{t+1}) \cdot \sum_{1 \le j \le K} \alpha_t^{(q)}(j) \cdot a_{jl}^{(q)}} \qquad pour \;\; : 1 \le i \le K$

- $\xi_t^{(q)}(i) = \frac{\alpha_t^{(q)}(i) \cdot \beta_t^{(q)}(i)}{\sum_{1 \le j \le K} \alpha_t^{(q)}(j) \cdot \beta_t^{(q)}(j)} \qquad pour \;\; : 1 \le i \le K$

- $\psi_t^{(q)}(i, j) = \frac{\alpha_t^{(q)}(i) \cdot a_{ij}^{(q)} \cdot f_j^{(q)}(y_{t+1}) \cdot \beta_t^{(q)}(j)}{\sum_{1 \le l \le K} f_l^{(q)}(y_{t+1}) \cdot \sum_{1 \le j \le K} \alpha_t^{(q)}(j) \cdot a_{jl}^{(q)}} \qquad pour : 1 \le i \le K, 1 \le j \le K$

...

3/ Echantillonnage

- Tirage des échantillons $\left(x^{(q),1}, x^{(q),2},...,x^{(q),m},...,x^{(q),M}\right)$ selon la loi $P\left(X/Y=y,\theta^{(q)}\right)$.

- Tirage de $x_1^{(q),m}$ selon la loi $P\left(X_1=\omega_i/Y=y,\theta^{(q)}\right)=\xi_1^{(q),m}(i)$.

- Tirage de $x_t^{(q),m}$ selon la loi $P\left(X_{t+1}=\omega_j/X_t=\omega_i,Y=y,\theta^{(q)}\right)=\psi_t^{(q),m}(i,j)$.

- Estimation des paramètres pour chaque échantillon :

$$- \overset{\wedge}{\pi}_i^{(q),m} = \frac{\sum_{1\le t\le T}\xi_t^{(q),m}(i)}{T}$$

$$- \hat{a}_{ij}^{(q),m} = \frac{\sum_{1\le t\le T}\psi_t^{(q),m}(i,j)}{\sum_{1\le t\le T}\xi_t^{(q),m}(i)}$$

$$- \overset{\wedge}{\mu}_i^{(q),m} = \frac{\sum_{1\le t\le T} y_t.1_{\left|X_t^{(q),m}=\omega_i\right|}}{\sum_{1\le t\le T}1_{\left|X_t^{(q),m}=\omega_i\right|}}$$

$$- \left(\overset{\wedge}{\sigma}_i^{(q),m}\right)^2 = \frac{\sum_{1\le t\le T}\left(y_t-\overset{\wedge}{\mu}_i^{(q),m}\right)^2.1_{\left|X_t^{(q),m}=\omega_i\right|}}{\sum_{1\le t\le T}1_{\left|X_t^{(q),m}=\omega_i\right|}}$$

4/ Mise à jours des paramètres

$$1/ \quad \overset{\wedge}{\pi}_i^{(q)} = \frac{1}{M}\sum_{1\le m\le M}\overset{\wedge}{\pi}_i^{(q),m}$$

$$2/ \quad \hat{a}_{ij}^{(q)} = \frac{1}{M}\sum_{1\le m\le M}\hat{a}_{ij}^{(q),m}$$

$$3/ \quad \overset{\wedge}{\mu}_i^{(q)} = \frac{1}{M}\sum_{1\le m\le M}\overset{\wedge}{\mu}_i^{(q),m}$$

$$4/ \quad \left(\overset{\wedge}{\sigma}_i^{(q)}\right)^2 = \frac{1}{M}\sum_{1\le m\le M}\left(\overset{\wedge}{\sigma}_i^{(q),m}\right)^2$$

4.3 : Détection d'objet mobile

4.3.1 : Procédure de détection

La procédure de détection d'objet mobile, permet de délimiter ce dernier dans son *Background*. Elle débute dés l'arrivée de l'image d'entrée Y contenant l'objet mobile. Cette dernière sera traitée par l'algorithme de détection basé sur un *MMC*. En premier, elle sera comparée à une image référence représentant le *Background* vide donnant ainsi l'image appelée *différence*. Puis, celle ci passera par les deux étapes essentielles de traitement, illustrées par **Fig (4.2)**. Une première, constituée de l'algorithme *ICE*, pour l'estimation des paramètres du modèle, suivie d'une deuxième, basée sur l'algorithme *MPM*, celle de la détection proprement dite. A la sortie, on obtient l'image d'entée, mais segmentée en deux (02) régions. Une couvrant le *Background* et l'autre délimitant l'objet mobile. Et de même pour l'image suivante.

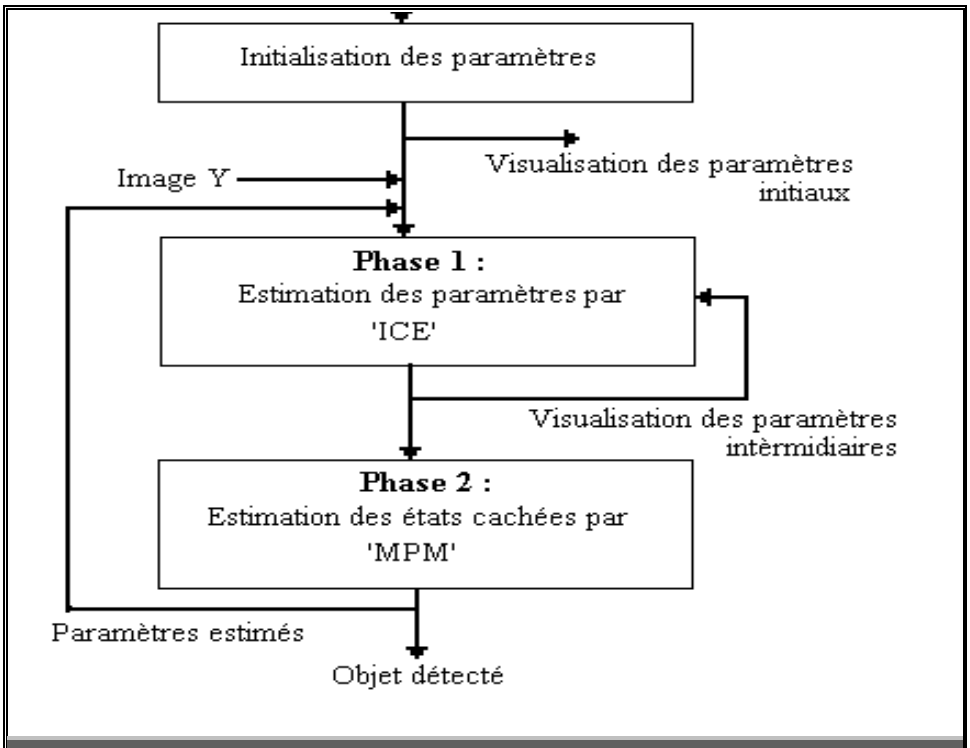


Fig (4.2) : ***Organigramme de la procédure de détection d'objet mobile***

4.3.2 : Phase de détection

Cette phase permet l'extraction d'objet mobile. Elle utilise les paramètres du modèle issus de l'algorithme *ICE*. L'algorithme choisi pour effectuer cette classification est celui dit *MPM* anoncé dans le chapitre précédant. Sa propriété connue est la minimisation des pixels mal classés.

L'algorithme *MPM* pour une Chaîne de Markov Cachée, se déroule donc de la manière suivante, **Tab (4.2)** :

Pour chaque élément *t* de la chaîne, et pour chaque classe ω_i possible, et en utilisant les valeurs dernières des paramètres on effectue :

a/ Le calcul des:

- probabilités Forward, $\alpha_t^{(q)}(i)$.
- probabilités Backward, $\beta_t^{(q)}(i)$.
- probabilités a posteriori marginales , $\xi_t^{(q)}(i)$.

b/ L'estimation de la réalisation de chaque X_t est donnée par l'état qui maximise $\xi_t^{(q)}(i)$:

$$X_t = \arg\max_{\omega_i} P(X_t = \omega_i / Y = y)$$

Tab (4.2) : ***Algorithme 'MPM' pour une Chaîne de Markov Cachée***

Pour chaque élément *t* de la chaîne

Pour chaque classe ω_i possible

1/ Forward

** Initialisation*

$$\alpha_1^{(q)}(i) = \frac{\pi_i^{(q)} \cdot f_i^{(q)}(y_1)}{\sum_{1 \le j \le K} \pi_j^{(q)} \cdot f_j^{(q)}(y_1)} \qquad pour \;\; :1 \le i \le K$$

** Récurrence*

$$\alpha_t^{(q)}(i) = \frac{f_{i\ i}^{(q)}(y_t) \cdot \sum_{1 \le j \le K} \alpha_{t-1}^{(q)}(j) \cdot a_{ij}^{(q)}}{\sum_{1 \le l \le K} f_l^{(q)}(y_t) \cdot \sum_{1 \le j \le K} \alpha_{t-1}^{(q)}(j) \cdot a_{jl}^{(q)}} \qquad pour \;\; :1 \le i \le K, \;\; t = 2,...,T$$

2/ Backward

** Initialisation*

- $\beta_T^{(q)}(i) = 1 \qquad pour \;\; :1 \le i \le K$

- $\xi_T^{(q)}(i) = \frac{\alpha_T^{(q)}(i)}{\sum_{1 \le j \le K} \alpha_T^{(q)}(j)} \qquad pour \;\; :1 \le i \le K$

** Récurrence* $pour \;\; t = T-1,...,1$

- $\beta_t^{(q)}(i) = \frac{\sum_{1 \le j \le K} a_{ij}^{(q)} \cdot f_j^{(q)}(y_{t+1}) \cdot \beta_{t+1}^{(q)}(j)}{\sum_{1 \le l \le K} f_l^{(q)}(y_{t+1}) \cdot \sum_{1 \le j \le K} \alpha_t^{(q)}(j) \cdot a_{jl}^{(q)}} \qquad pour \;\; :1 \le i \le K$

- $\xi_t^{(q)}(i) = \frac{\alpha_t^{(q)}(i) \cdot \beta_t^{(q)}(i)}{\sum_{1 \le j \le K} \alpha_t^{(q)}(j) \cdot \beta_t^{(q)}(j)} \qquad pour \;\; :1 \le i \le K$

3/ Classification

Pour chaque pixel, on retient la classe qui maximise la probabilité a posteriori marginale :

$$X_t = \arg\max_{\omega_i} P(X_t = \omega_i / Y = y)$$

Aller aux fichiers :

4_1,
4_2,
4_3,
4_4,
4_5,
4_6,
4_7,
4_8,
4_9,
4_10,
4_11,
4_12,
4_13.

Dispositif expérimental

5.1 : Introduction

Ce chapitre donne la description du dispositif expérimental et ces différents modules électroniques. Il regroupe les diverses parties constituant la chaîne de traitement d'images. Il permet le développement et l'implantation en temps réel, des algorithmes destinés au traitement d'image et de la vidéo, tel que la segmentation, la compression, ainsi que la détection d'objet mobile dans une séquence d'images; l'objectif de notre travail.

5.2 : Présentation de la plate-forme expérimentale

La plate-forme expérimentale illustrée par la figure **Fig (5.1)**, est utilisée pour valider les algorithmes de détection d'objet mobile. Elle est conçue autours d'un processeur de traitement de signal; le *'DSP TMS320C6711'* de '*TEXAS INSTRUMENTS'*. Le module carte mère est présenté par le kit de développement nommé *'DSK*[5]*TMS320C6711'* [41]. Celui-ci intègre des interfaces de communication, un interface analogique (voix et données), et des ports d'entrée/sortie, dans une seule carte pour former un système d'évaluation destiné aux applications liées à l'imagerie.

[5] : *DSP Starter Kit*

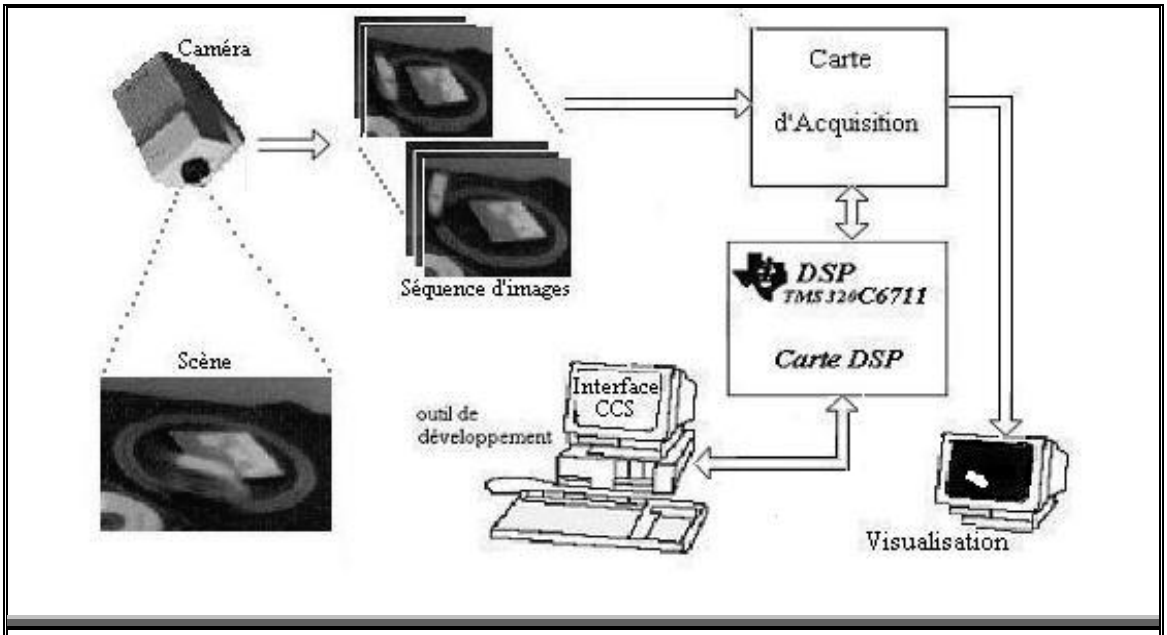


Fig (5.1) : ***Plate-forme expérimentale***

Cette carte '*DSK*' est reliée à un micro-ordinateur contenant un outil de développement software appelé '*CCS*[6]', simplifiant la saisie et le déboguage des algorithmes destinés aux traitements. Nous décrivons par la suite, les différents modules de ce dispositif. Une caméra représentant le capteur d'images, est reliée à une carte d'acquisition vidéo intitulée '*IDC*[7]'. Les deux précèdent la carte '*DSK*'. L'ensemble composé de la caméra, la carte '*DSK*', et celle '*IDC*' forme le Kit de traitement vidéo et images surnommé '*IDK*[8]'. Ce hardware est complété par un Software adéquat permettant de monter rapidement de la conception d'algorithmes à leur implantation sur processeur '*DSP*'. La figure **Fig (5.2)** illustre les constituants du kit '*IDK c6000*', tel livré de **'***TEXAS INSTRUMENTS*'.

[6] *:* Code Composer Studio
[7]: Imaging Daugther Card
[8] : Imaging Developper's Kit

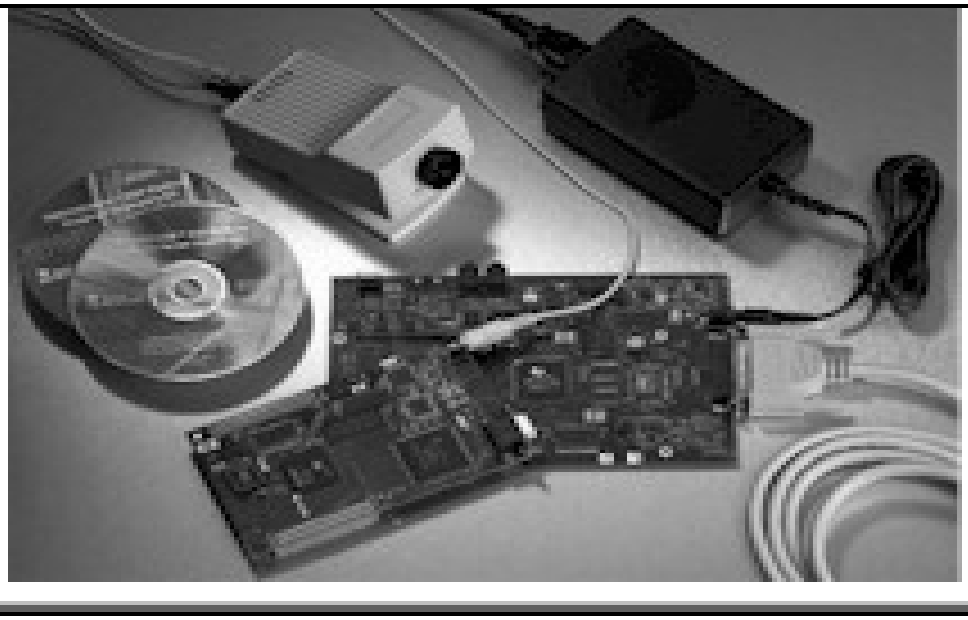


Fig (5.2) : C*omposants du kit 'IDK C6000'*

5.3 : Le capteur d'images

Le capteur d'images est une caméra de type '*CCD*[9]'. C'est le premier élément dans la chaîne d'acquisition d'une image, certes, sans lui pas d'images.

Une caméra *'CCD'* possède un élément sensible à la lumière reçue, nommé *capteur* '*CCD'*, ou « *dispositif de transfert de charges* ». Ainsi, Ces *capteurs* '*CCD'* fournissent un signal vidéo analogique échantillonné représenté ci après, par la **Fig (5.3)** . La tension de sortie est lue et amplifiée par un étage analogique.

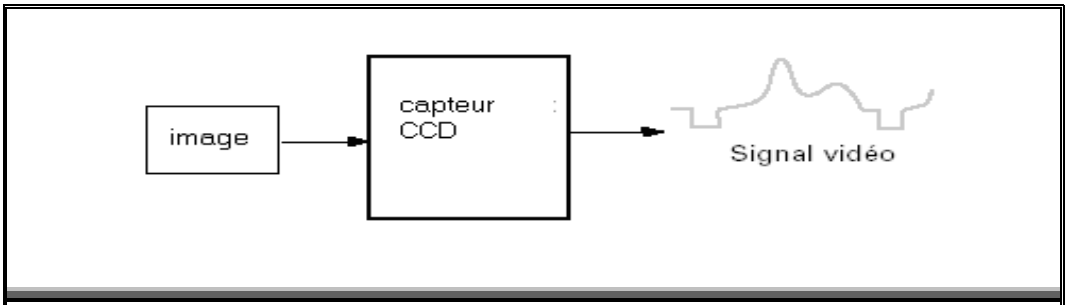


Fig (5.3) : *Signal de sortie d'un capteur CCD*

[9] *:* Charge Coupled Device

Le kit d'imges '*IDK*' accepte deux types de caméra '*NTSC*' ou '*PAL*' . chacune est responssable de génerer un signal *vidéo composite,* vihéculant l'information à traiter vers la carte '*IDC*'.

Un signal vidéo composite est le résultat d'un multiplexage analogique des tois signaux de couleur primaires '*R G B*[12]' , ainsi que signal de synchronisation [47]. Cependant, ces signaux ne sont pas pris dans leurs états initaux, mais sous forme de composantes Y représentant la « *Luminance* » et C donnant le vecteur « *Chrominance* ». la première est obtenue par une combinaison linéaire des valeurs primaires, selon la formule suivante :

$$Y = 0.299.R + 0.587.G + 0.114.B$$

Tandis que le vcecteur Chrominance appelée aussi « *différence de couleurs* », donne les deux composants Cr et Cb, calculés respectivement par :

$$Cr = R - Y \quad et \quad Cb = B - Y$$

Notons que le language fréquent, utilise l'expression de « ***la Chrominance*** **»** comme s'il s'agissait d'une entété unique. La figure **Fig (5.4)** suivante, présente ces signaux pour la génération d'une mire de barres couleur [46, 47].

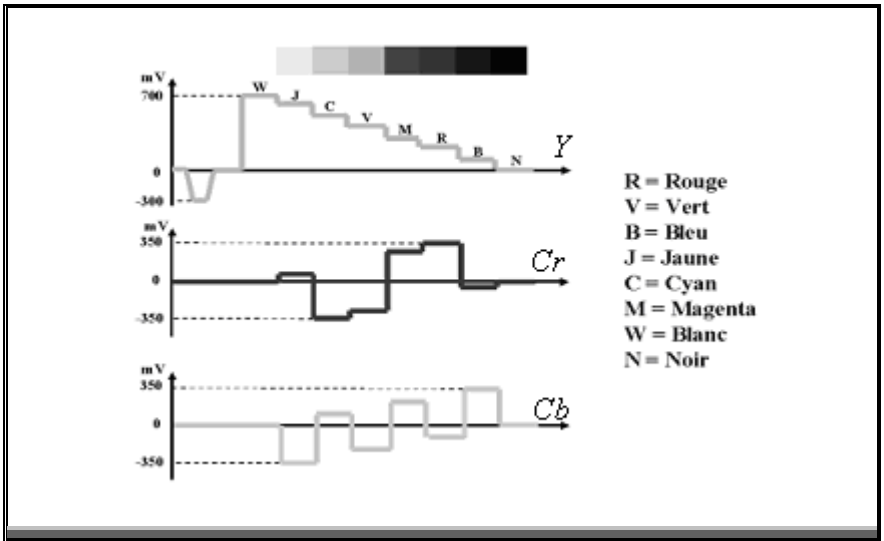


Fig (5.4) : ***Signaux de luminance et de chrominance pour une mire de barres couleur***

Pour la transmission d'images, on est amené à regrouper au sein d'un unique signal, les informations concernant les trois couleurs analysées. La composante *Y* est transmise avec la synchronisation, pratiquement sans modification. Cependant, le multiplexage de ces composants est réalisé par des procédés multiples, dont nous citons les deux variantes suivantes:

❖ **Procédé NTSC**

C'est un format américain. Son nom est tiré de l'expression '*National Television Systems Committee'*. Inventé en *1953,* historiquement il est le premier procédé apparu. Il est à balayage de *60* fois par seconde, soit 2 demi-images balayées *30* fois, soit *30* images complètes en une (01) seconde). La définition de l'image est de 640 points par 475 lignes utiles (x *25* par seconde). Le système *NTSC* utilise une base de couleurs 'Y I Q', dérivée de la représentation 'Y Cr Cb', comme suit:

$$I = 027.(B-Y) + 0.74.(R-Y)$$
$$Q = -0.41.(B-Y) + 0.48.(R-Y)$$

Ce changement d'axes est introduit dans l'objectif de tenir compte des caractéristique physiologiques de la vision humaine [47].

❖ **Procédé PAL**

Pour '*Phase Alternate Line'* . C'est un format européen, mis en application en *1962*, qui s'est fortement inspiré du modèle américain (*NTSC*), se profettant ainssi d'une bonne décennie d'expérience opérationnelle. Il est à balayage de *50* fois par seconde, soit 2 demi-images balayées *25* fois, soit *25* images complètes en une (01) seconde.

La définition de l'image est de *720* points par *576* lignes utiles (x *25* par seconde). Ce système utilise une base de couleurs 'Y U V', trée elle aussi de la présentation'Y Cr Cb'.

Les signaux image et vidéo numériques se présentent tout différemment. En effet, l'image numérisée est échantillonnée spatialement [47]. Cela veut dire quelle est constituée de points séparés dits '*Pixel*[12]', dont la valeur de la lumière est représentée par un chiffre codé sur un certain nombre de bits. L'image numérisée donc, n'est autre qu'un fichier (au sens informatique du mot), où les données sont rangées dans un tableau. Chaque pixel est donné par son rang. La synchronisation dans ce cas, n'est pas demandée. Il est clair que de telles images peuvent être manipulées facilement, puisque n'importe quel traitement est converti en un calcul sur des nombres. Les opérations tel que la rotation, l'extraction de contours, le seuillage, …etc, sont alors à la base d'effets spéciaux numériques.

La limite basse d'échantillonnage est imposée par le théorème de *Shannon*. De nombreuses possibilités de combinaisons pour la luminance et la chrominance peuvent être choisies. En admettant que les besoins en résolution sont moindres pour la chrominance que pour la luminance, des formats multiples ont été normalisés.

La figure **Fig (5.5)**, donne les dispositions des échantillons de luminance Y et de chrominance (Cr et Cb) dans les formats « *4 :2 :2* » et « *4 :2 :0* ». A titre d'exemple, le format « *4 :2 :2* » en 625 lignes comporte 720 points de luminance sur 576 lignes et 360 points de chrominance sur 288 lignes.

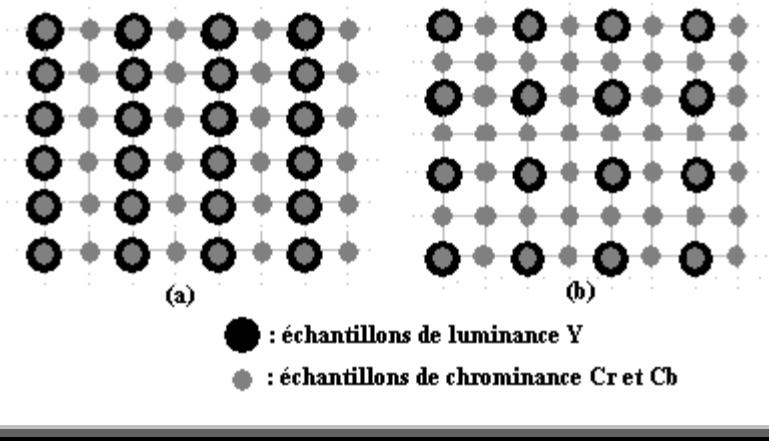


Fig (5.5) ***: Dispositions des échantillons de luminance et de chrominance (a) format« 4 :2 :2 », (b) format « 4 :2 :0 »***

5.4 : La carte d'acquisition vidéo

La carte d'acquisition vidéo représentée par la figure **Fig (5.6)**, appelée aussi « *carte fille* » ou '*Daugther Card'*, est destinée pour la capture, l'affichage et le formatage de données [48, 49, 51]. D'une part, elle est constituée d':

- Un circuit décodeur vidéo *TVP5022*.
- Une palette vidéo représentée par le circuit intégré *TVP3036*.
- Un circuit intégré Xilinx *FPGA 'Field Programable Gate Array*', qui se charge des deux tâches ; capture et affichage.
- Une mémoire ' *SDRAM*'de *16 Mbits* pour la capture.
- Un connecteur reliant la carte '*IDC'* à celle '*DSK'*.
- Un connecteur *RCA* pour l'entrée vidéo composite.
- Un connecteur *15* broches pour la sortie *RGB* vers le moniteur.

D'autre part, cette carte d'acquisition offre les possibilités suivantes :

- Le signal d'entrée vidéo est limité à un *NTSC* ou *PAL*.
- Le signal doit être de format composite.
- L'affichage est soit en niveau de gris sur *08bits*, ou en *RGB* sur 16bits.

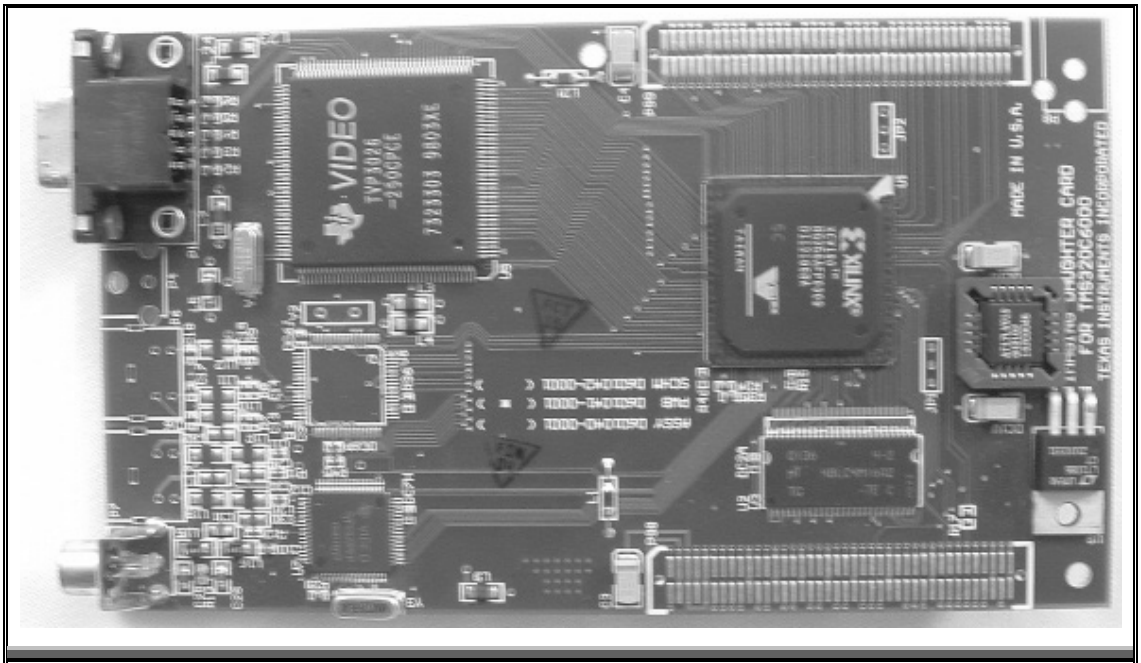


Fig (5.6) : ***La carte d'acquisition vidéo***

5.4.1: Le système de Capture

La figure **Fig(5.7)**, illustre le schéma bloc de la partie assurant la capture des images[49]. L'information issue de la caméra est transmise vers le décodeur *TVP5022*. Ce dernier effectue un échantillonnage du signal vidéo au format « *4 :2 :2* », puis un filtrage passe bas. Le circuit '*FPGA*' se charge par la suite, de la séparation de ce flux digital en trois composantes ; une luminance Y et deux chrominances Cr et Cb. Il loge ces informations dans une zone mémoire appelée '*capture frame buffer*', située sur la carte fille et composée de trois buffers de capture. Chacun est constituée de deux champs séparés '*odd*' et '*even*'. Le décodeur *TVP5022* génère une interruption extérieure '*EXTINT5*' à chaque changement de champs de capture.

Le mode de capture NTSC ou PAL est spécifié par logiciel, via le '*Driver*' de capture. Ce dernier supporte les deux modes suivants:

- NTSC: 640x480, YCbCr 4:2:2, 30 trames/seconde.
- PAL: 768x576, YCbCr 4:2:2, 25 trames/seconde.

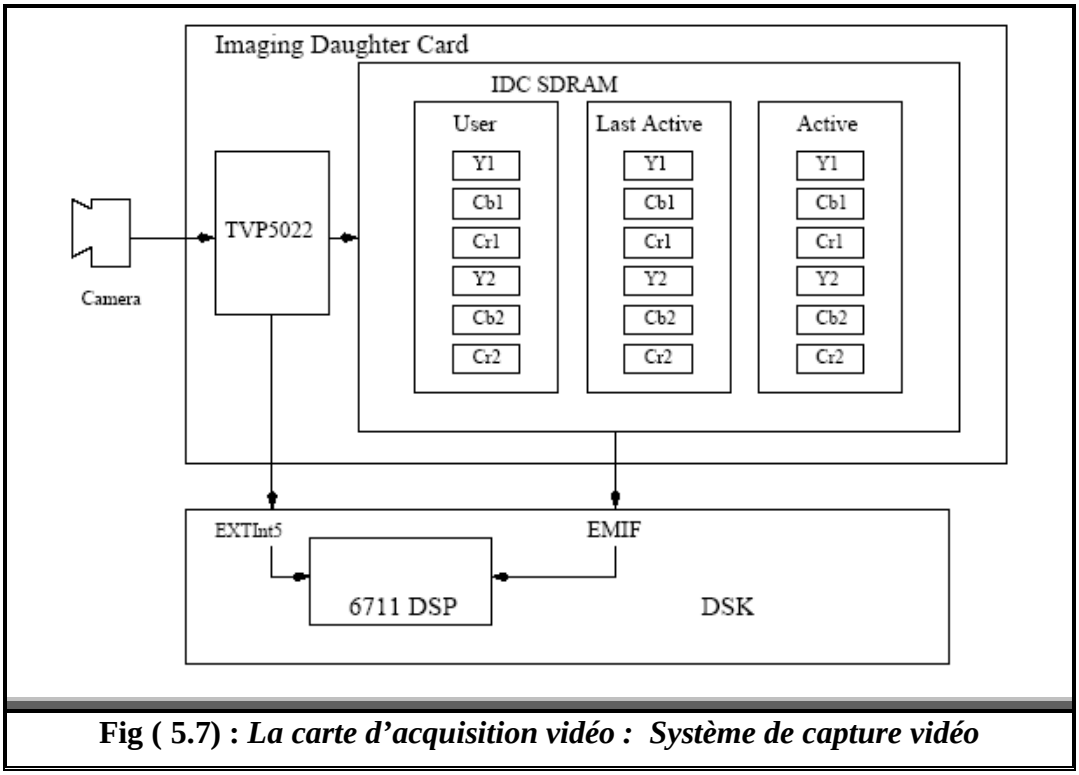


Fig (5.7) : ***La carte d'acquisition vidéo : Système de capture vidéo***

La figure **Fig(5.8)**, illustre l'organisation des trois buffers de capture associé à un format *NTSC*. Le regroupement de ces zones en une image complète est laissé à l'utilisateur.

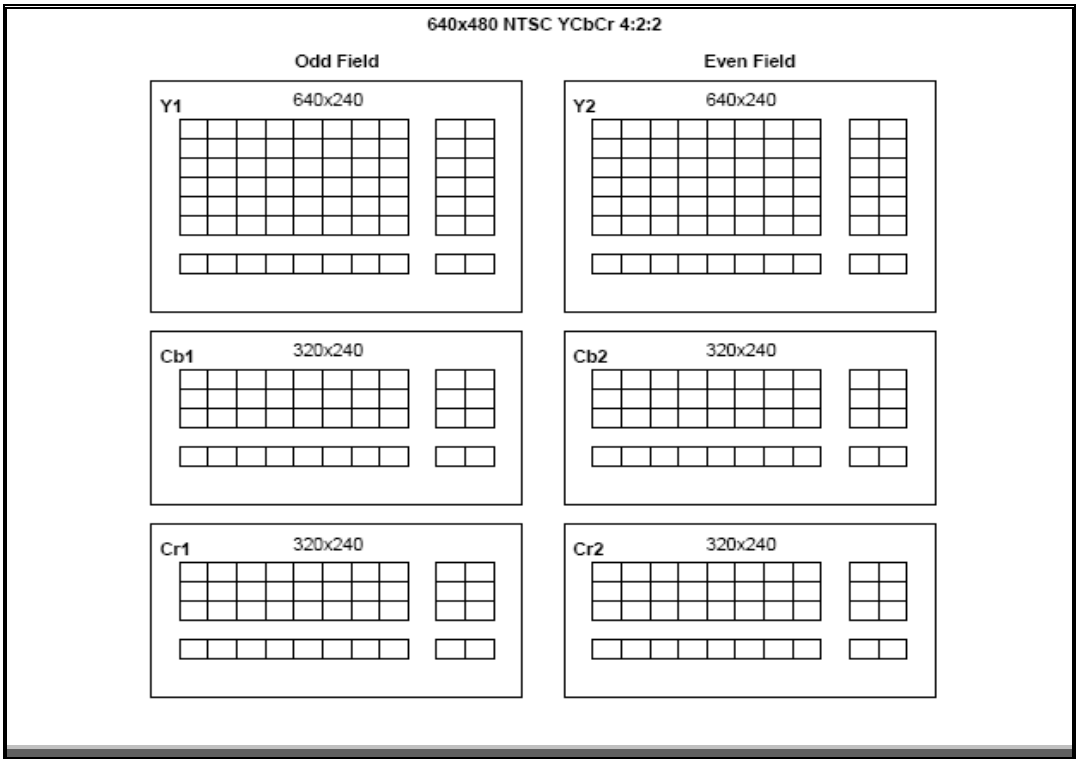


Fig (5.8) : ***Organisation de la mémoire de capture pour un format NTSC « 4 :2 :2 »***

Tous les signaux de synchronisation de l'entrée vidéo sont générés par le *TVP5022*, spécifiquement celui de la synchronisation verticale, qui donne une interruption vers le processeur *'DSP'*, marquant la fin d'une trame [48, 49, 51]. La zone mémoire de capture fait partie de l'espace mémoire total adressable par le *'DSP C6711'* et elle est configurée en lecture seule. D'autre part, elle est accessible via l'interface de mémoire externe *'EMIF , External Memory Interface'*. Une bufferisation triple est utilisée pour éliminer toute attente du part de l'application. Si celle-ci atteint la fréquence 30 trames/seconde, elle est physiquement en mode circulaire avec le circuit *'FPGA'*. Ce dernier contrôle deux buffers à tout moment. Tandis que l'application possède le troisième. Une application plus rapide que la normale amène à une duplication de trames. Par contre, une perte de trames est signalée avec une autre plus lente.

5.4.2 : Le système de Visualisation

La figure **Fig(5.9)** quant à elle, illustre le schéma bloc de la partie chargée de l'affichage [49]. Le circuit *'FPGA'* génère les signaux de synchronisation pour la sortie. D'une part celui de synchronisation horizontale, est connecté à l'entrée d'interruption extérieure *'EXTINT7'* configurée pour copier une ligne de données du buffer nommé *'Display Buffer*' vers la carte d'acquisition *'IDC' (FIFO*). Le circuit *TVP3036* est chargé de transmettre cette ligne vers le moniteur. D'autre part celui de la synchronisation verticale, est utilisé pour marquer la fin d'une trame.

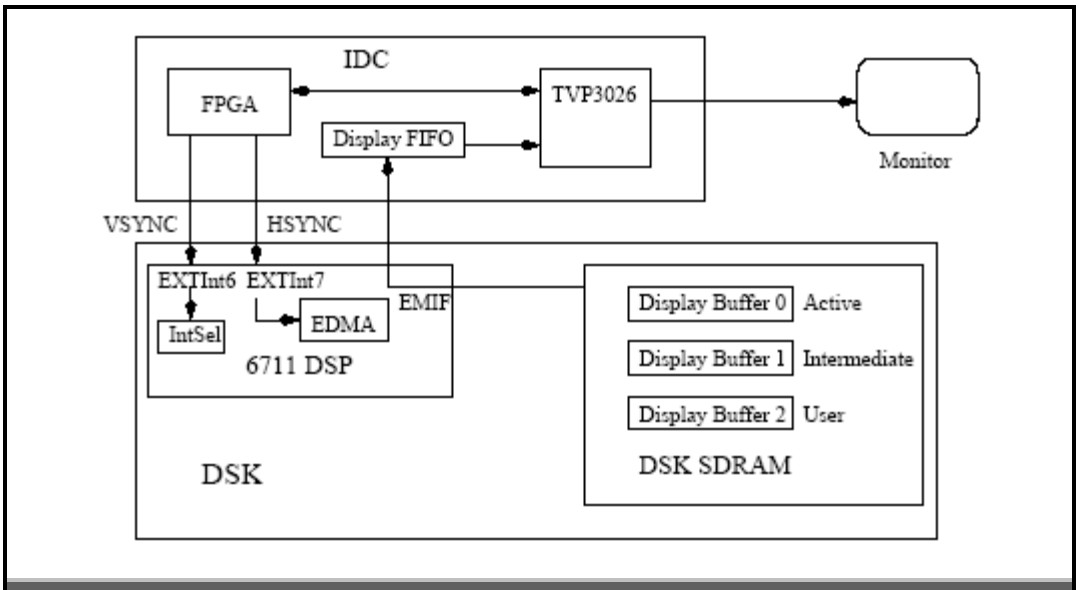


Fig (5.9) : ***La carte d'acquisition vidéo : Système de visualisation***

Une fois encore, une bufferisation triple est assurée pour éviter à l'application des états d'attente. L'organisation de chaque buffer est montrée par la figure **Fig (5.10)**. Seulement ces buffers se trouvent cette fois ci dans la mémoire *'SDRAM ; Synchronous Dynamic Random Access Memory'*, sur la carte *'DSK'*. Les données sont transférées de celle-ci vers la carte fille via le circuit *'EDMA* ; *Enhanced Direct Memory Access'*. Le buffer dit « *Actif »* est celui en utilisation par ce dernier. Celui dont l'application est propriétaire est nommé « *Utilisateur »*. Le troisième est « *Intermédiaire »*, et il sera prochainement visité par l'application. Si l'accès est lent, des trames seront affichées plus qu'une fois. Par contre, une application rapide entraîne une perte de trames.

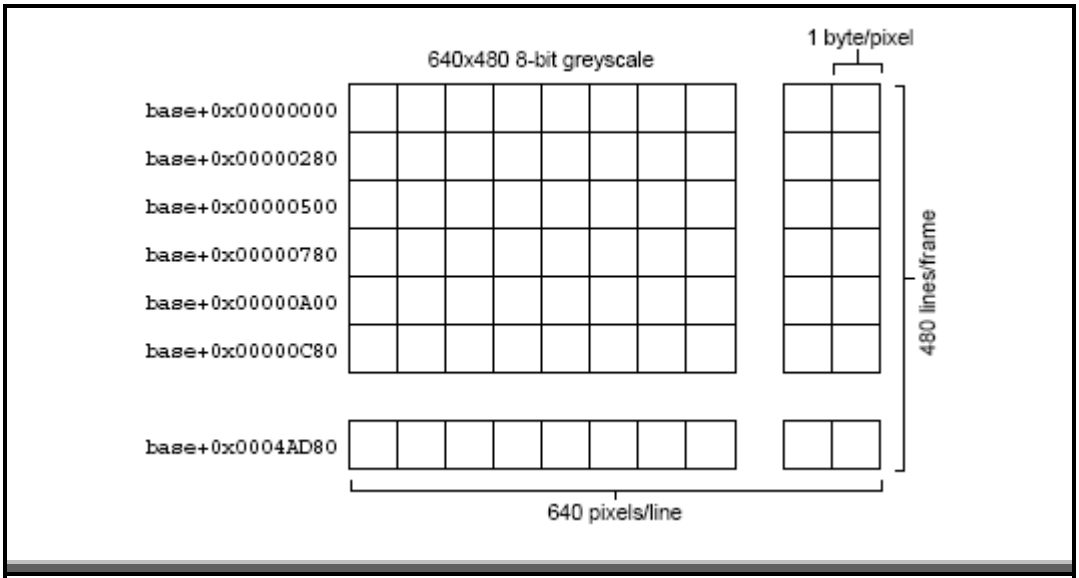


Fig (5.10) : ***La carte d'acquisition vidéo : Système de visualisation***

De même, le mode d'affichage doit être configuré par programmation dans la routine d'initialisation, via le *'Driver'* d'affichage. Ce dernier supporte les quatre modes suivants:

- 640X480X8: 640x480, en niveau de gris (08bits/pixel), à une fréquence de 60Hz
- 640X480X16: 640x480, en couleur (16bits/pixel, format565) à une fréquence de 60Hz
- 800X600X8: 640x480, en niveau de gris (08bits/pixel), à une fréquence de 60Hz
- 800X600X16: 640x480, en couleur (16bits/pixel, format565) à une fréquence de 60Hz

5.5 : Le kit de développement *'DSK TMS320C6711'*

La carte *'DSK TMS320C6711'* est le module chargé de traiter les données et d'assurer les transferts entre les déférents éléments de la chaîne, afin de permettre un bon déroulement des programmes et d'applications.

5.5.1 : Description générale

Ce Kit de développement, est une carte mère à base d'un processeur '*DSP*'. Il permet d'évaluer et de développer des applications autour du '*DSP TMS320C6711*' de '*TEXAS INSTRUMENTS*' [41]. Ces constituants principaux sont,:

- Le '*TMS320C6711*' : un *DSP* à virgule flottante.
- Une interface de périphériques parallèle.
- Une mémoire '*SDRAM*' et une autre '*ROM*'.
- Un circuit d'interface analogique '*AIC*' à 16bits.
- Un port d'entrée sortie.
- Un support d'émulation '*JTAG*' intégré.

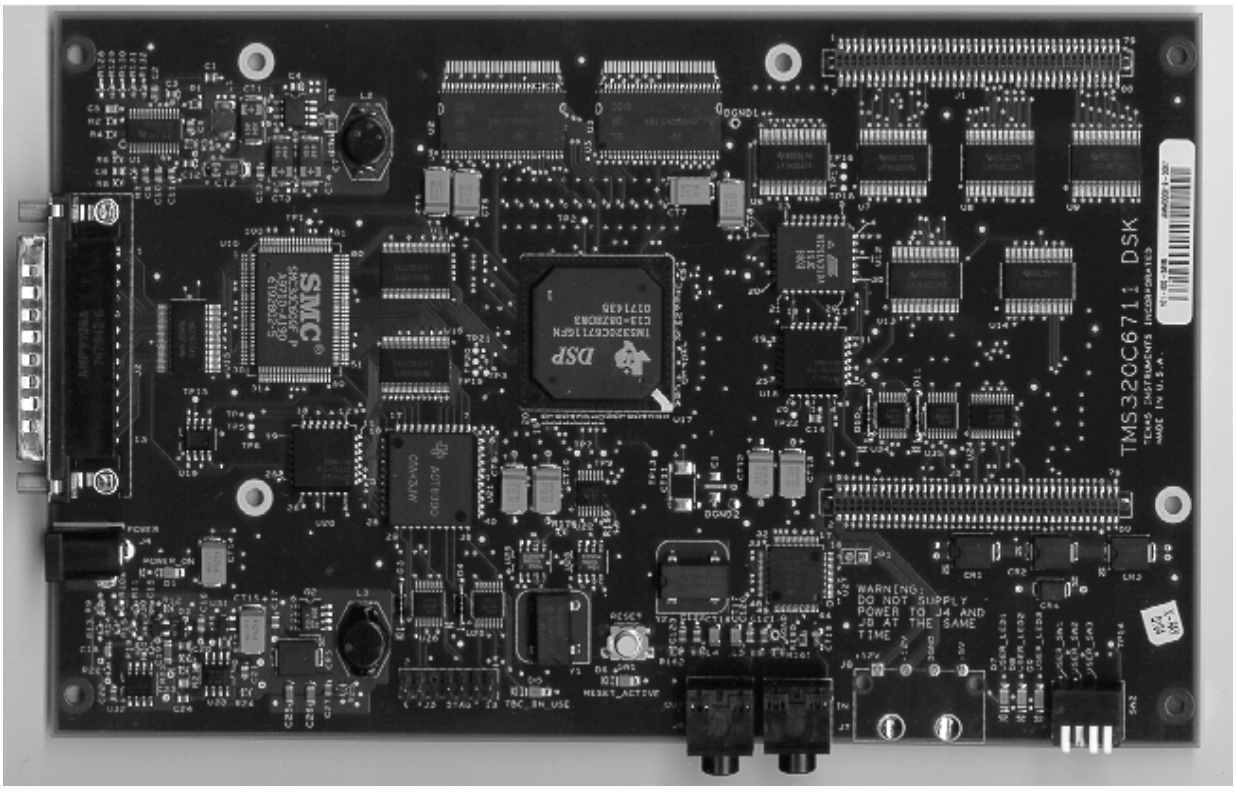


Fig (5.11) : ***Vue de haut du Kit 'TMS320C6711DSK'***

L'intégration de ce module dans une chaîne de traitement, nécessite l'utilisation de certains connecteurs disponibles sur la carte. La totalité de ces derniers est composée de:

- Un connecteur de *25*pins (male *DB-25*, *IEEE*Standard1284) pour interfaçage parallèle.
- Une entrée d'alimentation.
- Un connecteur de *14*pins pour émulation '*JTAG*' extérieure.

- Une entrée audio.
- Une sortie audio.
- Une paire de fiches connecteurs de *80*pins chacune pour associer la carte fille.

Ce kit *'DSK'* est fourni avec un Software constitué d'un compilateur de code 'code composer studio ; *CCS'*, offrant à l'utilisateur un environnement de développement confortable, soit en langage assembleur, soit en langage évolué *C*, ou même en langage *C++*.

La figure **Fig (5.11)** précédente donne une vue de haut de ce kit. Tandis que celle **Fig (5.12)** ci après, illustre le schéma bloc reliant ces déférents composants.

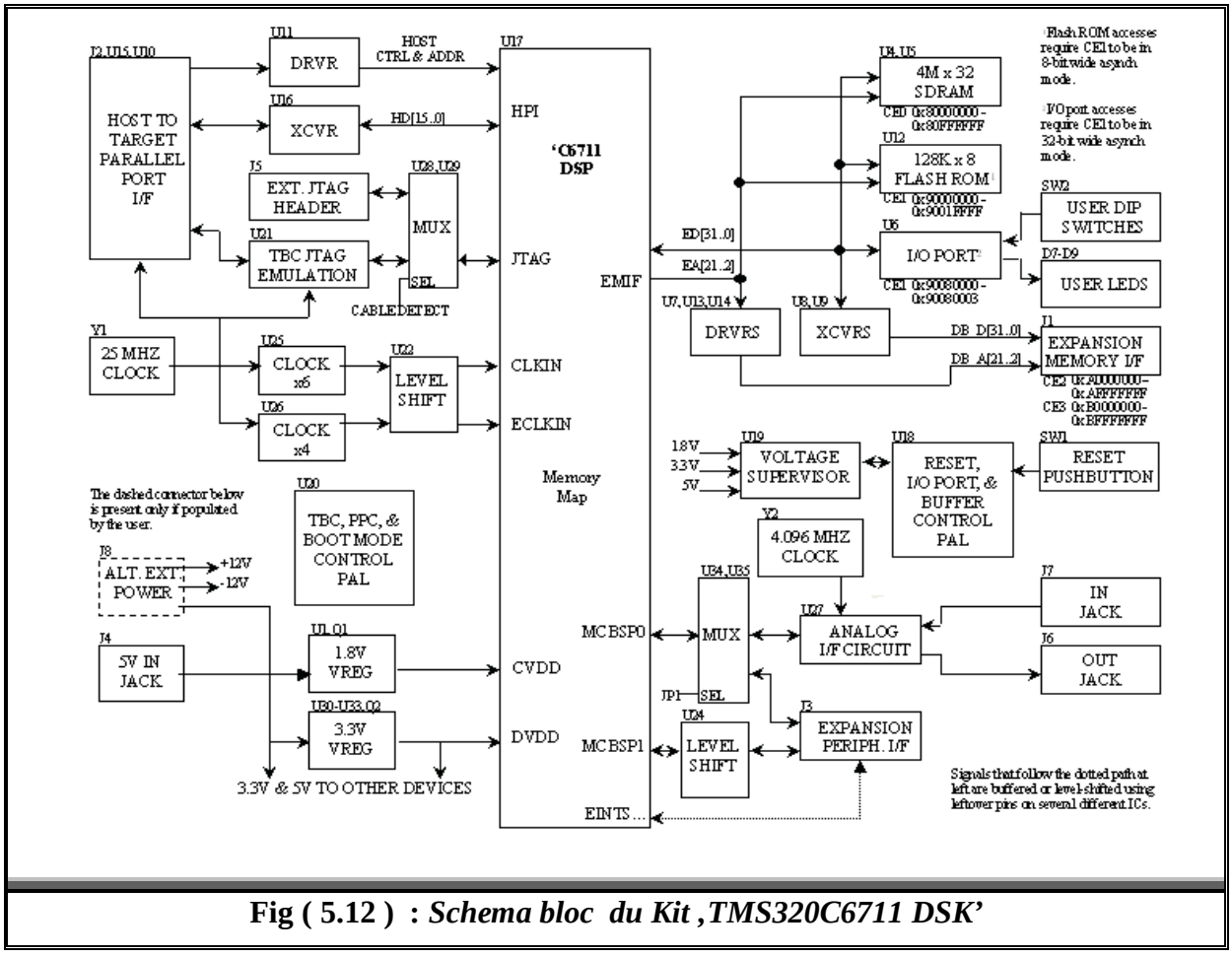


Fig (5.12) : *Schema bloc du Kit ,TMS320C6711 DSK'*

5.5.2: Caractéristiques principales du module *'DSK TMS320C6711'*

A fin d'assurer des conditions de traitement convenables aux applications d'imagerie, ce kit *'DSK'* présente les capacités suivantes

- Horloge de 150Mhz, capable d'exécuter 900 millions opérations à virgule flottante par seconde *'MFLOPS'*.
- Horloge duale: *'CPU'* travaillant à *150MHz* et interface mémoire externe *'EMIF'* de *100MHz.*

- Une mémoire de *16MBytes* de type '*SDRAM ; Synchronous Dynamic Random Access Memory'* à *100MHz.*
- Une mémoire *de 128KBytes* de type '*ROM: flash programmable and erasable Read Only Memory'*.
- Une interface '*HPI: Host Port Interface'* pour accéder à toute la mémoire du *DSP* via le port parallèle.
- Un codeur/décodeur audio '*audio codec'* à *16bit.*
- Un nombre de Six '*LEDs'* indicateurs; d'alimentation et reset cités à titre d'exemple.
- Mémoire d'extension et connecteurs de périphériques pour les cartes filles.

5.5.3 : Le processeur '*DSP TMS320C6711'*

Ce processeur fait partie de la génération des *DSPs TMS320C67X*, à virgule flottante. Sa plate forme est le *TMS320C6000*. Son architecture est du type '*VLIW*' pour '*Very Long Instruction Word'*. Cette caractéristique (mot d'instruction très long), lui offre une performance élevée surtout de point de vue vitesse, et faisant de ce *DSP* un excellent choix surtout pour les applications multi-cannaux '*multi-canals'*, et /ou multi-fonctions.

Opérant avec une fréquence de *150Mhz*, le '*C6711'* peut exécuter jusqu'à *900* millions d'instructions à virgule flottante par seconde '*MFLOPS'*. A l'aide d'une paire de multiplieurs à virgule flottante ou fixe, il pourrait faire jusqu'à *300* millions de multiplication-accumulation par seconde '*MMACS* ; *Million MACs per Second'*.

Le '*DSPC6711'* contient une mémoire interne avec une architecture à deux niveaux '*L1'* et '*L2'*. Le premier niveau est divisé en deux parties séparées :

- ✓ '*L1P'* : mémoire cache de programmes de *04Kbits*.
- ✓ '*L1D'* : mémoire cache de données de *04Kbits.*

Par contre le deuxième, peut être configurée en partie autant que mémoire cache ou *SRAM* (données et le programme).

La figure **Fig (5.13)**, donne le schéma fonctionnel du '*CPUC6711'*, montrant ses déférents constituants intégrés [40].

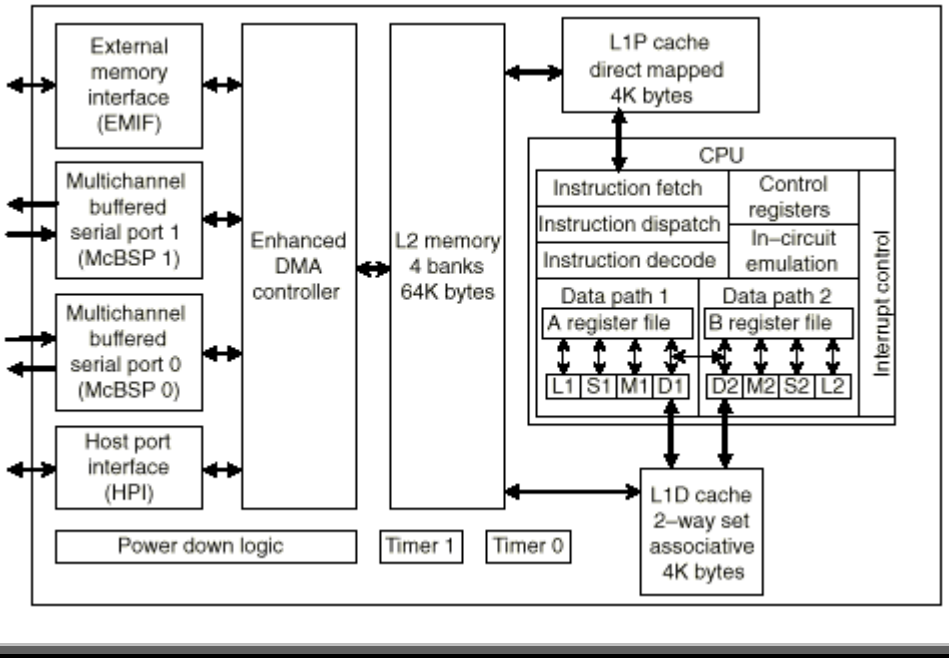


Fig (5.13) : ***Schéma fonctionnel du 'CPU C6711'***

5.6 : Outils de développement Software du kit '*IDK*'

Un ensemble complet d'outils *Software* accompagne le kit de développement d'images '*IDK*'. Il inclus un interface compositeur de code '*Code Composer Studio CCS*'. Celui-ci offre des utilitaires divers:

- Un compilateur *C/C++* optimisé.
- Un assembleur.
- Un éditeur de liens.
- Et un simulateur à base des familles *C6000, C5000* et *C2000*.

Une application sous *CCS* peut donc, être développée en langage *C*, ou *C++*, [41, 43] ou en assembleur.

Les phases d'un développement typique d'une application, sont présentées sur l'organigramme de la figure, **Fig (5.14)**.

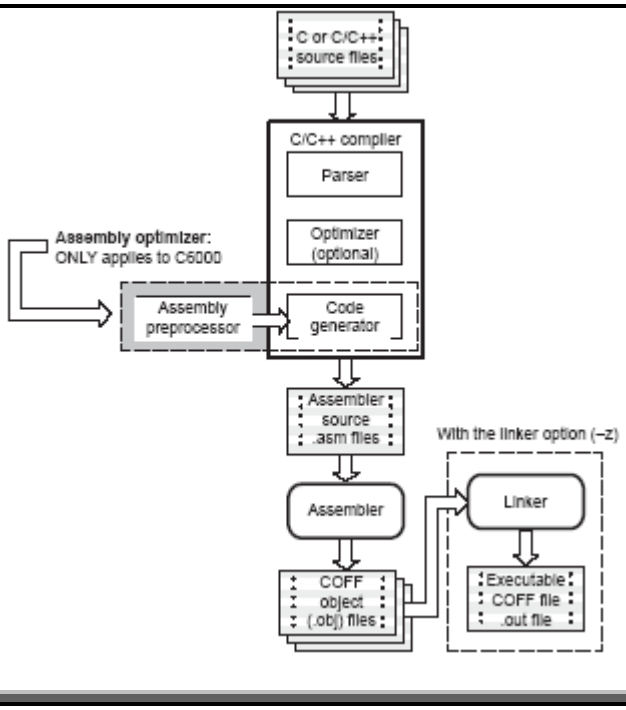


Fig (5.14) : ***Déroulement typique de génération de code exécutable d'une application sur 'DSP' sous 'CCS'.***

Les *DSPs* de **'***TEXAS INSTRUMENTS'* comporte un support d'émulation intégré permettant au '*CCS'* de contrôler l'exécution des programmes en temps réel. Cette émulation intégrée fournit des différentes possibilités hardware tel que :

- Démarrage, arrêt et remise a zéro du *DSP*.
- Chargement des codes depuis ou vers le *DSP*.
- Consultation des registres ou de la mémoire du *DSP*.
- Chargement de données en temps réel '*RTDX '* entre le moniteur et le *DSP*.

Le traitement et l'analyse en temps réel sont garantis par deux outils puissantes de l'interface 'CCS', simplifiant le développement et le déboguage des programmes:

- ✓ '*RTDX'* ; '*Real Time Data eXchange'* : permet à l'utilisateur du système de transférer les données entre le micro-ordinateur et le dispositif '*DSP'* sans arrêt de l'application en cours d'exécution sur le processeur *DSP* [43].
- ✓ Le noyau '*DSP/BIOS*' : ensemble d'instrumentations permettant d'assurer une communication synchrone entre le micro-ordinateur et le processeur '*DSP'* , offrant une abstraction totale du hardware et un analyse en temps réel [43].

Derrière cette interface utilisateur, la carte *'IDK'* offre un Software typique sous forme de modules regroupant chacun un ensemble de routines appelée *'Paquet'* [48]. Parmi ceux, on cite:

- *'ImageLIB'* : un ensemble de fonctions bibliothèques destinées aux traitements vidéo et image. Ceux sont des routines dites *'Ready to use'*, optimisées, écrites en assembleur et à usage général. Leur utilisation diminue considérablement le temps de développement et de calcul.
- *'Chip Support Labrary CSL'* : un ensemble d'interfaces d'application programmables *'API'*, utilisés pour configurer et contrôler les périphériques du *'DSP'*. Le but de cet utilitaire est de simplifier l'usage des derniers, avoir un code portable et en fin offrir une abstraction du hardware. Il est donc possible de développer une application sans avoir physiquement l'accès aux registres, et donc une programmation simple, rapide et avec peu d'erreurs. Par la suite, ce code est exécutable sur tous les *DSPs* de la famille *C6000*.
- *'Image Data Manager IDM'* : Cet ensemble de fonctions donne une abstraction des demandes *'DMA ;* Direct Memory Access' et permette le transfert des données en parallèle avec le traitement (en arrière plan). Il se charge de la mise à jours des pointeurs ainsi que le management des buffers. En fin, il utilise les appels *'CSL'*, pour assurer le transfert de données entre la mémoire interne et externe durant le traitement.

Ainsi l'implantation d'un algorithme sur le kit *'IDK'* est réalisé comme suit :

Le code source est écrit en langage *C* sur le *PC*, en exploitant les utilitaires de l'interface *CCS*. Puis, il est compilé, assemblé et lié, donnant un fichier de format objet *'COFF*[10]*'***.** Celui-ci est chargé sur la carte *'DSK'* à travers le port parallèle. Initialisé depuis le *PC*, l'exécution du programme sur ce kit devient autonome. Ainsi, la carte *'DSK'* reçoit via la carte fille, les images capturées par la caméra. Elle effectue le traitement nécessaire sur ces données. En traversant la carte *'IDC'* pour la seconde fois, les résultats seront finalement affichés sur le moniteur.

[10]: Common Object File Format

Nous terminons par un dernier et important point, concernant les conditions minimums du système pour accepter le kit de développement d'images '*IDK*'. Le PC doit donc, avoir les propriétés ci après :

- ✓ Etre un Pentium de *233MHz* (*Pentium* II de *500MHz* ou plus est recommandé).
- ✓ Avoir une mémoire de *64Mbytes* de *RAM* (*128Mbytes* est recommandé).
- ✓ Avoir un port parallèle à transfert bidirectionnel (*EPP* est recommandé).
- ✓ Utilisant comme système d'exploitation *Windows98, 2000* ou *NT4.0*.
- ✓ Avoir un deuxième moniteur VGA (SVGA est recommandé).

5.7 : Conclusion

Dans ce chapitre nous avons présenté la chaîne représentant le dispositif expérimental mis en œuvre et ces différents modules :

- o Le capteur : Caméra '*CCD*'.
- o La carte fille '*IDC*'
- o La carte '*DSK TMS320C6711*'.

D'une part, l'étude de cette chaîne de traitement, constituée essentiellement par le kit '*TMSC6000 IDK*', ainsi que la découverte des qualités nombreuses de ce dernier, ont montré l'adéquation parfaite de ce dispositif pour le traitement d'images et de la vidéo. D'autre part vu ses capacités d'exécution élevées, l'architecture en microcontrôleur de son processeur 'DSP*C6711*' et les possibilités efficaces offertes par sa carte vidéo, il est facile d'atteindre des implantations en temps réel.

En fin, les outils Software en association avec ce kit, et l'utilisation des langages évolués, permettent et aident considérablement à développer des applications avec simplicité et efficacité.

Conclusion générale et perspectives

1 : Conclusion générale

Le but de ce travail était de détecter un objet mobile dans une séquence d'images prises avec une caméra stationnaire, en utilisant les *Modèles de Markov Cachés*, spécifiquement les *Chaînes de Markov Cachées*. L'étape de la « *détection* » constitue alors, une phase indispensable pour les applications en liaison avec le mouvement, telles que la compression des images dynamiques.

Pour extraire l'objet mobile de son environnement, nous avons servi des méthodes d'estimation ayant un aspect probabiliste et statistique, et appartenant à la famille des outils représentables graphiquement. Nous avons donc, choisi un *Modèle de Markov Caché MMC* pour établir la discrimination entre le '*Background*' et l'objet mobile. L'originalité de ces Modèles Markoviens réside dans l'introduction du contexte spatial de l'image, montré par la matrice de transition, et la relation entre état actuel et état suivant, à l'aide des déférentes mesures de probabilités, ainsi que leur rapidité comparée aux champs de Markov cachés. L'algorithme *'ICE Eterative Condititinal Estimation'* qui est en principe utilisable en segmentation d'images a montré son efficacité dans ce domaine.

L'implantation de ces algorithmes dans une chaîne expérimentale de traitement, autour d'un processeur de la famille 'TMS320C6000' de '*TEXAS INSRTUMENTS'*, englobe l'aptitude à mettre en œuvre ces méthodes de détection, et d'obtenir des résultats concrets comparables avec celles de la simulation.

Nous avons donc, commencer par découvrir l'état de l'art concernant les techniques de détection. Cette étude montre le recourt croissant à des outils probabilistes de traitement. Les chaînes de Markov présentent un meilleur exemple. Ces dernières offrent une supériorité par rapport aux autres méthodes. Cela est du au fait qu'elles respectent la cohérence spatiale et temporelle des classes constituant la séquence d'images.

Un tour d'horizon couvrant plus au moins la théorie des modèles *MMCs*, est ensuite donné, exposant ainsi les trois problèmes en liaison avec l'utilisation de ces techniques. Ces problèmes doivent être résolus, dans n'importe qu'elle application à base des modèles *MMCs*. Cette partie se termine par proposer leurs solutions usuelles se trouvant dans la littérature.

Cependant, il est indispensable de spécifier les déférentes quantités associées à l'exploitation de ces modèles *MMCs.* D'une part, nous avons donné les mesures de probabilités nécessaires et souligner les difficultés rencontrées dans leur calcul. Pour cela, nous avons exprimé les transformations d'aide, ainsi que les algorithmes spécifiques pour la détermination de ces quantités. Ces algorithmes constituent le noyau de chaque mesure. D'autre part, nous avons cité les déférentes méthodes de passage de l'espace *2D* à celui *1D*, terminant par choisir celui d'Hilbert-piano, dont la propriété est d'assurer l'équivalence entre le contexte spatial de l'image et celui temporel de la chaîne de *Markov*.

La construction d'un modèle Markovien, dans le but de réaliser la détection d'objet mobile, nécessite à tout près d'effectuer une estimation de ses paramètres. La recherche bibliographique faite au début ce travail, nous a permet de découvrir les déférents algorithmes dans ce contexte, et leurs inconvénients. Ainsi, vu les avantages présentés par l'algorithme '*ICE'* qui est largement utilisé en segmentation, nous avons décidé de l'introduire, dans notre domaine de détection. La phase de la détection a été élaborée par une méthode de classification basée sur l'estimateur *MPM*. L'association de ces deux algorithmes, a abouti à des résultats satisfaisants.

Dans l'attente d'une implantation en temps réel, le dispositif expérimental constituant la chaîne de traitement d'images, a été conçue autour du processeur spécialisé d'images, le '*TMS320C6000'* de '*TEXAS INSRTUMENTS'*. Un tel choix est adopté à cause des qualités

Hardware et *Software* présentés par ce kit, assurant un environnement convenable au traitement d'images.

2 : Perspectives

Vu les limitations exprimées par les résultats de simulation, des extensions pour notre travail peuvent être trouvées, donnant ainsi les perspectives suivantes :

- Dans notre étude, nous avons exploité seulement, la caractéristique « *Intensité* » de l'image. Celle-ci engendre des détections fausses dans les régions de ressemblance entre les classes. Pour confronter ce problème amenant à une mauvaise décision, il est indispensable d'utiliser d'autres supports de calcul, citons à titre d'exemples : la caractéristique couleur, fréquence, …etc.
- Dans notre modélisation, nous avons considéré un nombre de deux classes : '*Background*' et '*Forground*'. Or, les modèles de Markov cachés présente une propriété puissante ; celle d'accepter de nouveaux états. Par conséquent, une extension en vue de la détection des « *ombres* » est possible. Elle peut être effectuée avec simple ajout d'un troisième état qui représente ces derniers. Et de même, la suppression d'un état est aussi possible. De telles actions sont imposées par la nature non stationnaire des phénomènes naturels. Des algorithmes visant la topologie du modèle peuvent être utilisés pour résoudre le problème de changement de structure.
- En fin, le choix de gaussiennes pour modéliser les interactions observations-états cachées, est arbitraire. Or, l'étude préalable de l'environnement et le phénomène à analysé, amène à des expressions de densités plus adaptées et donc une amélioration considérable du modèle, surtout dans les applications spécifiques et bien définies.

Table des figures

Liste des Tableaux

Notations

$X = (X_1 X_2 ... X_T)$: Séquence de variables aléatoires, ou chaîne de Markov et de même pour $X = (X_t)_{t=1}^{T}$.

$\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$: Ensemble fini d'états pour un processus de Markov.

K : Nombre d'états possibles (cardinal de l'espace d'états).

T : Longueur de la chaîne.

A : Matrice de transition.

a_{ij} : Elément indicé *ij* de la matrice de transition.

$P(X_t = \omega_j / X_{t-1} = \omega_i)$: Probabilités de transition.

$P(t)$: Vecteur stochastique à l'instant *t*.

$P(0)$: Vecteur stochastique à l'instant *t=0*.

π : Vecteur initial.

π_i : Elément du vecteur initial.

$P(X_1, X_2, ..., X_T)$: Probabilité d'une séquence d'états $X = (X_1 X_2 ... X_T)$.

O : Séquence d'observation.

$P(O / \text{mod}\,èle)$: Probabilité d'une séquence d'observation sachant le modèle.

$X = (X_t)_{t=1}^{T}$: Processus caché de la chaîne de Markov cachée.

$Y = (Y_t)_{t=1}^{T}$: Processus observable de la chaîne de Markov cachée.

$B = \{b_{X_t}(Y_t)\}$: Probabilités d'émission.

$\lambda = (\pi, A, B)$: Notation condensée du Modèle de Markov Caché.

$P(Y/\lambda)$: Probabilité d'une séquence d'observation Y sachant le modèle λ.

$\hat{\lambda}$: Notation désignant les paramètres estimés d'un *MMC*

$Y_{Training}$: Séquence d'apprentissage.

$P(Y, X/\lambda)$: Probabilité jointe de Y et X sachant ce modèle λ.

$\alpha_t(i)$: Probabilité *'Forward'*.

$\beta_t(i)$: Probabilité *'Backward'* .

$\xi_t(i)$: Probabilité a posteriori marginale : probabilité d'être à l'état ω_i à l'instant t sachant l'observation complète Y.

$\psi_t(i, j)$: Probabilité jointe conditionnelle : probabilité d'être à l'état ω_i à l'instant t, et à l'état ω_j à l'instant suivant, sachant la séquence d'observation Y et le modèle λ.

N : Nombre de pixels dans l'image.

L : Nombre de lignes ou de colonnes.

$x = (x_t)_{t\in S}$: Réalisation de X.

$y = (y_t)_{t\in S}$: Réalisation de Y

c_{ij} : Probabilité jointe d'être à l'état ω_i à l'instant t et à l'état ω_j à l'instant suivant.

$f_{X_t}(y_t)$: Distribution de Y_t conditionnelle à X_t, représentée par une densité de probabilité gaussienne de Y_t.

μ_i : Moyenne de la gaussienne.

σ_i : variance de la gaussienne associée à la classe *i*.

$P(X)$: Distribution à priori de X.

t_{ij}^t : Probabilités de transition de la chaîne non stationnaire.

K' : Niveau du parcours *d'Hilbet-piano*.

$\hat{x} = \arg\max_x P(X = x/Y = y)$: Estimée de X par *MAP*.

$\hat{x}_t = \arg\max_{x_t} P(X_t = x_t / Y = y), \quad \forall t$: Estimée de *X* par *MPM*.

$\delta_t(i)$: Probabilité correspondante au sous chemin optimal; allant de X_1 à ω_i visitée à l'instant t.

q : Itération d'ordre *q* de l'algorithme *EM* ou *ICE*.

$\pi_i^{(q)}, a_{ij}^{(q)}, f_i^{(q)}$: Paramètres du modèle à l'itération *q* de l'algorithme *EM* ou *ICE*.

θ : Notation condensée des paramètres du modèle.

$\theta^{(q)}$: ensemble des paramètres du modèle à l'itération *q* de l'algorithme *EM* ou *ICE*.

x^m : Réalisation de *X* d'indice *m* (la *i-ème* réalisation).

Y : Luminance.

(Cr, Cb) : Vecteur Chrominance.

Table des matières