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* ».

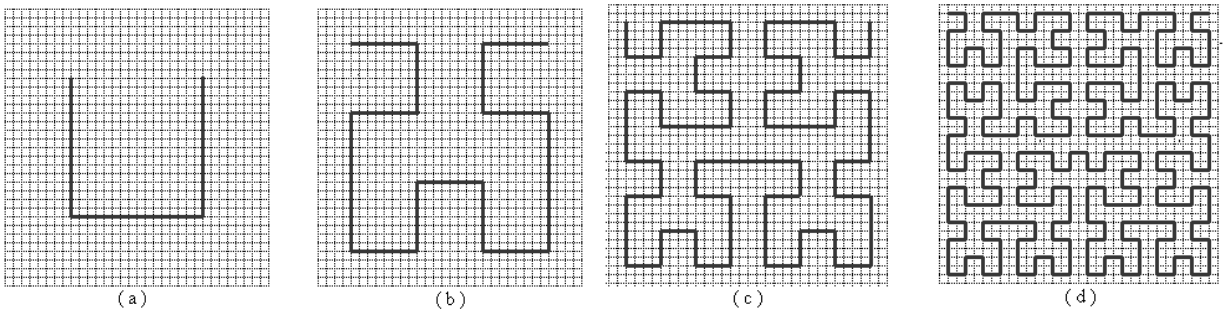


Fig (3.1) : ***Construction du Parcours d'Hilbet-piano pour une image 16x16***
(a)Initialisation, (b) et (c) Etapes intermédiaires, (d) Résultat final

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.

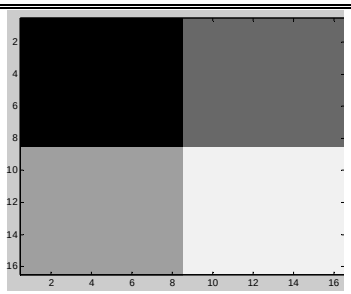


Image « *carrés* » : (a)

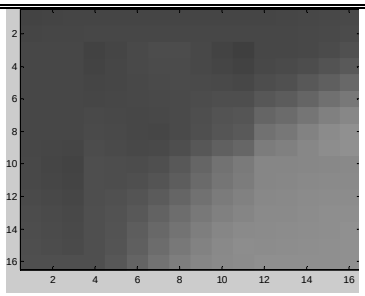


Image « *Disque* » : (a)

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.

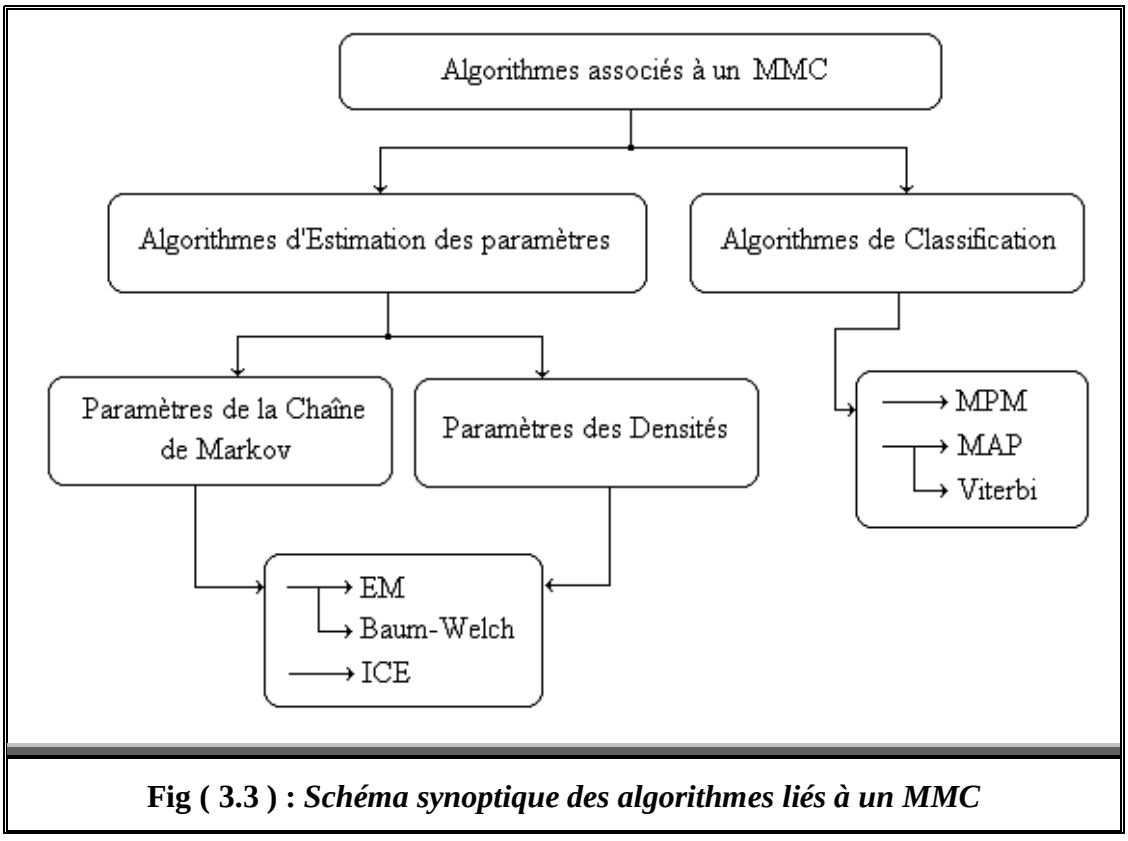


Fig (3.3) : ***Schéma synoptique des algorithmes liés à un MMC***

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 »*

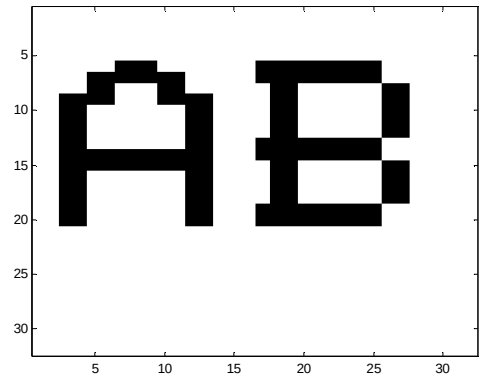
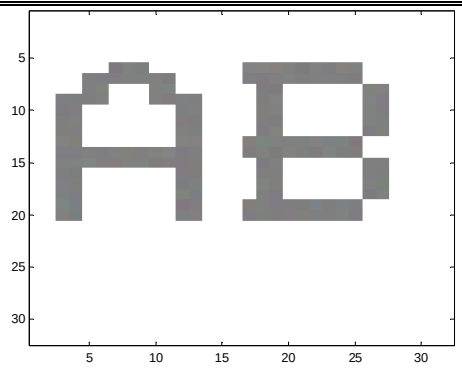


Fig (3.4) *: Image originale « AB » de taille : 32x32 pixels*

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



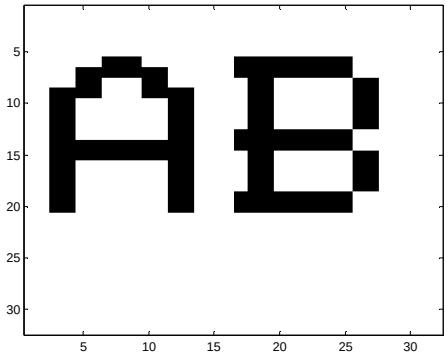


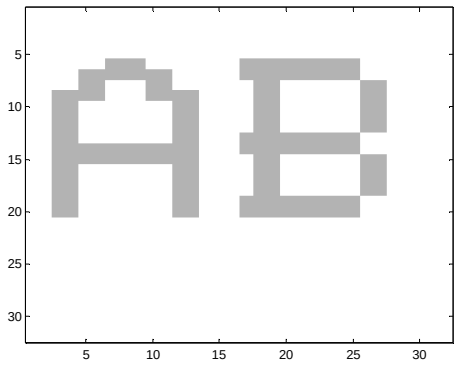
Image originale + bruit gaussien de :
moyenne : 0.5
variance : 0.0001

```
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)***



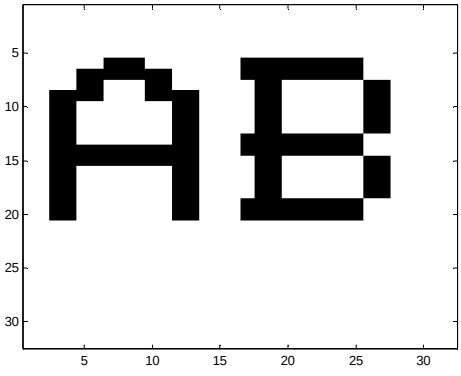


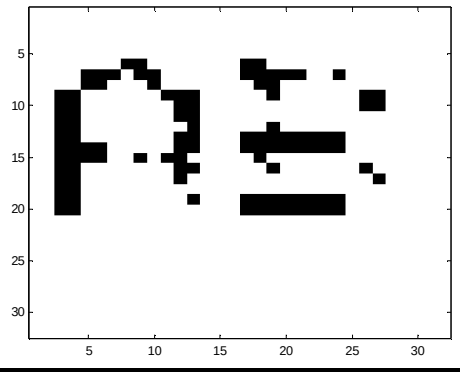
Image originale + bruit gaussien de :
moyenne : 0.7
variance : 0

```
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
```

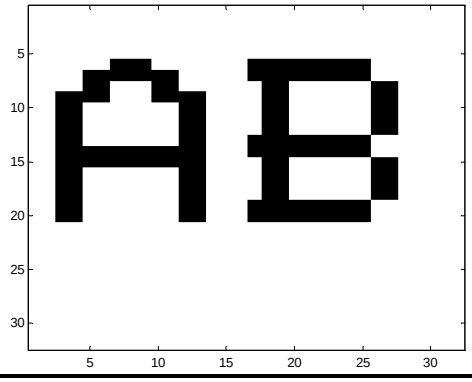
Fig (3.6) : ***Segmentation (1-2) : Mu=200 250; Seg=10 10;***

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

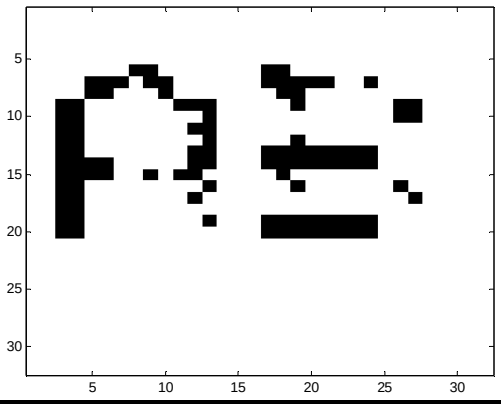


Q=3 ;
Paramètres de MMc :
P = 0.4 0.6
A =0.3 0.7
0.2 0.8
Seg = 10 10
Mu = 90 130

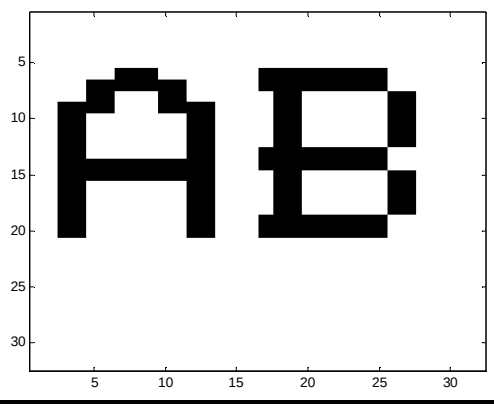
Fig (3.7) *: Segmentation (1-3) : Mu=90 130; Seg = 10 10;*

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)***

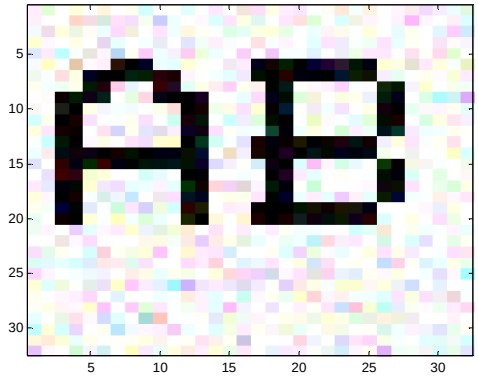
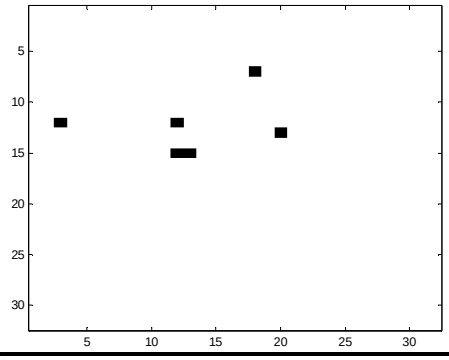
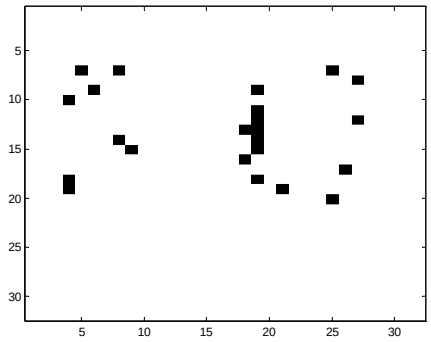


Image « AB » + bruit gaussien de :

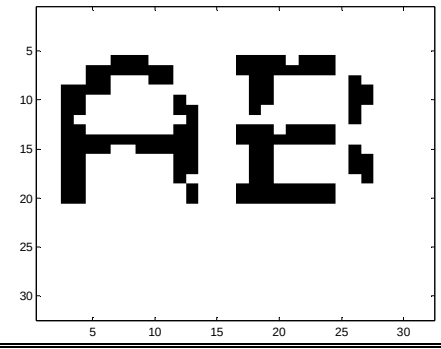
moyenne : 0 et de variance : 0.01



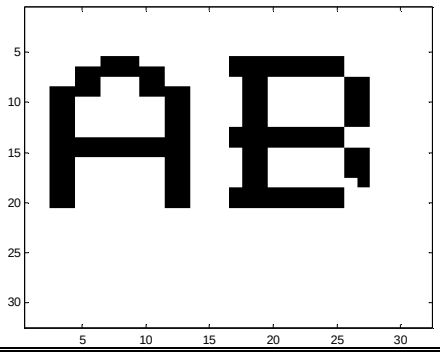
```
Q=1 ;
Paramètres de MMc :
P =0.4 0.6
A =0.3 0.7
   0.2 0.8
Mu = 100 230
Seg =100 400
```



```
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 »

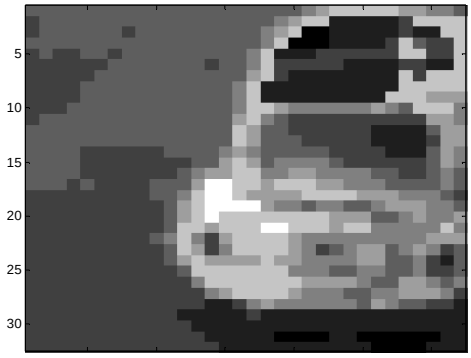
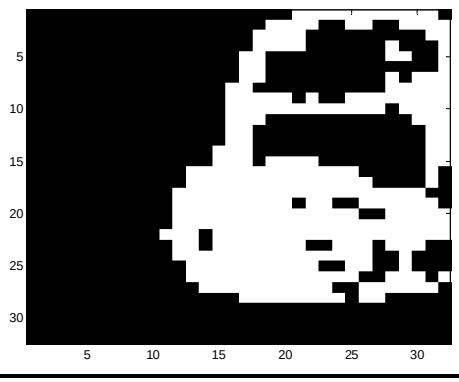
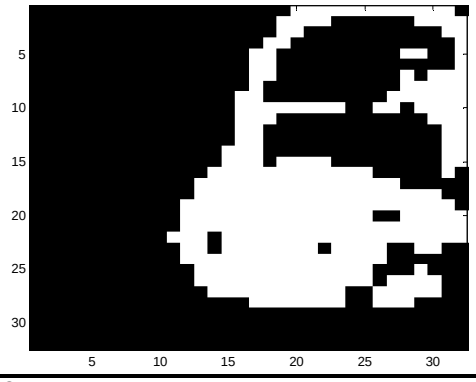


Fig (3.10) : ***Image originale « Voiture » de taille : 32x32 pixels***

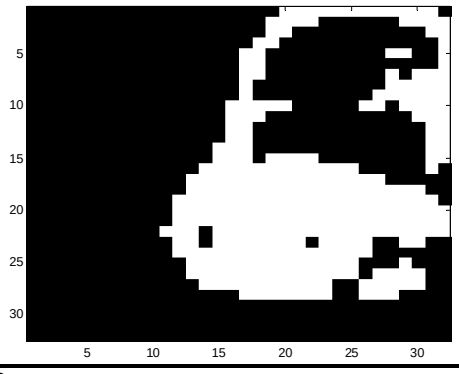
a : ***Segmentation (2-1)***



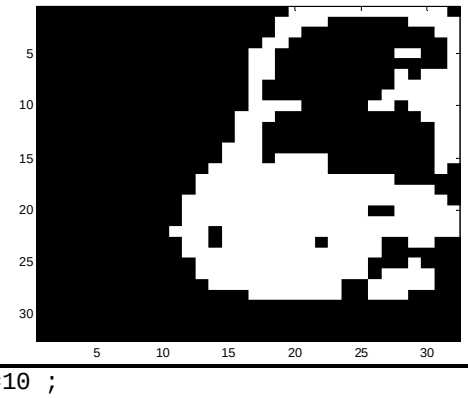
```
Q=1 ;
Paramètres de MMc :
P = 0.4 0.6
A =0.3 0.7
    0.2 0.8
Mu =  150 100
Seg = 10 10
```



```
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)***

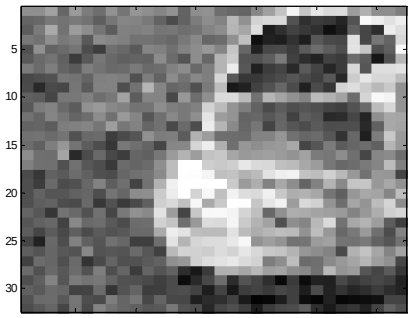
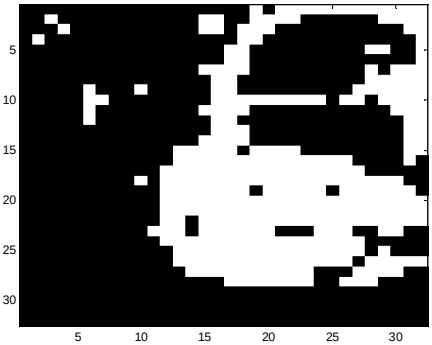
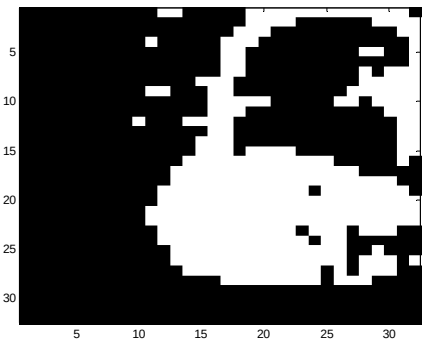
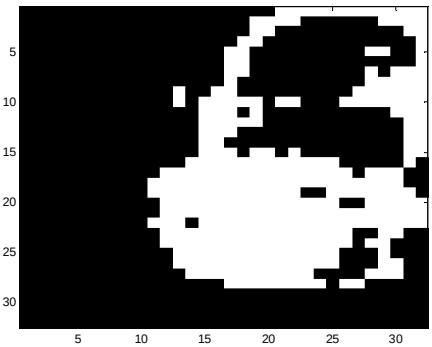
<table>
<tr><td>  </td><td>  </td></tr>
<tr><td>Image originale « Voiture » + bruit gaussien de :
moyenne : 0.10
et variance : 0.01</td><td>Q=1;
Paramètres de l’HMM :
P = 0.4 0.6
A =0.3 0.7
0.2 0.8
Mu = 150 100 Seg = 10 10</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Q=2;
Paramètres de l’HMM :
P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

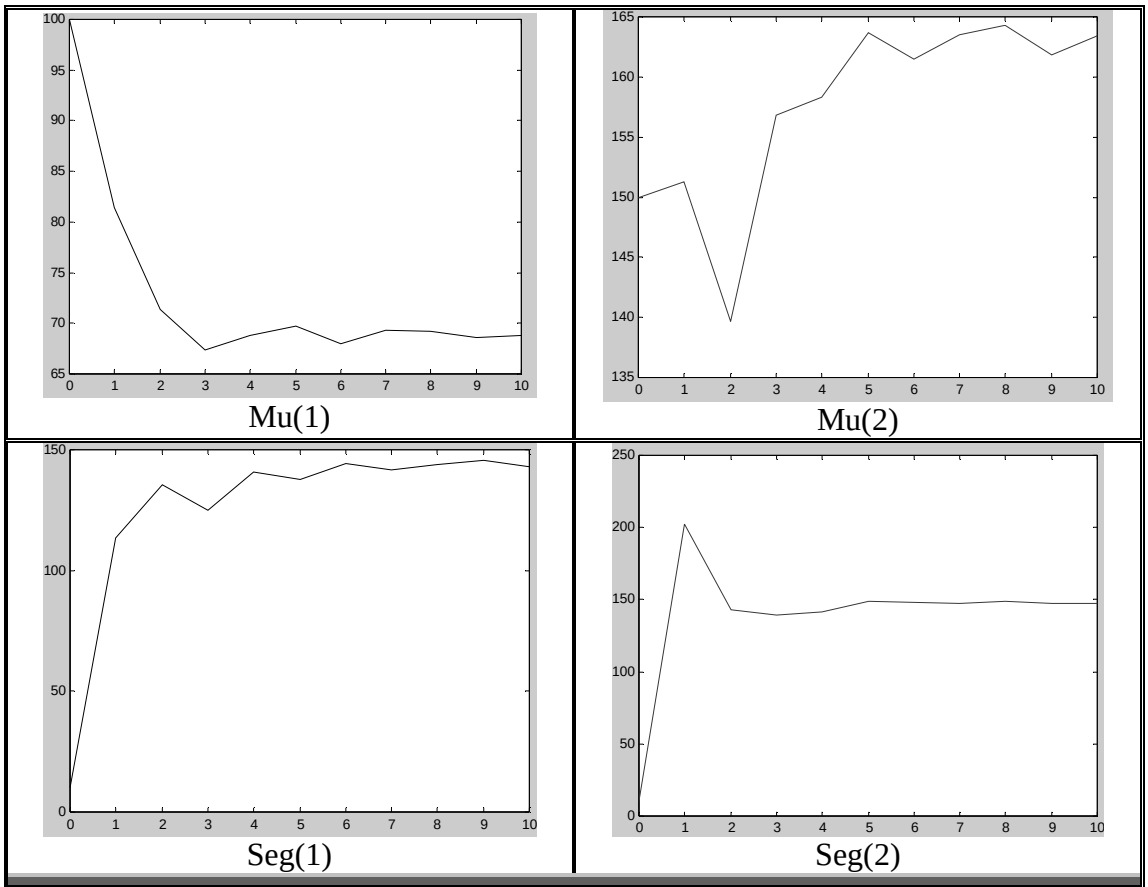


Fig (3.13) ***: Evolution des moyennes et des variances lors de la segmentation (2-1) en fonction du nombre d'itérations***

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.

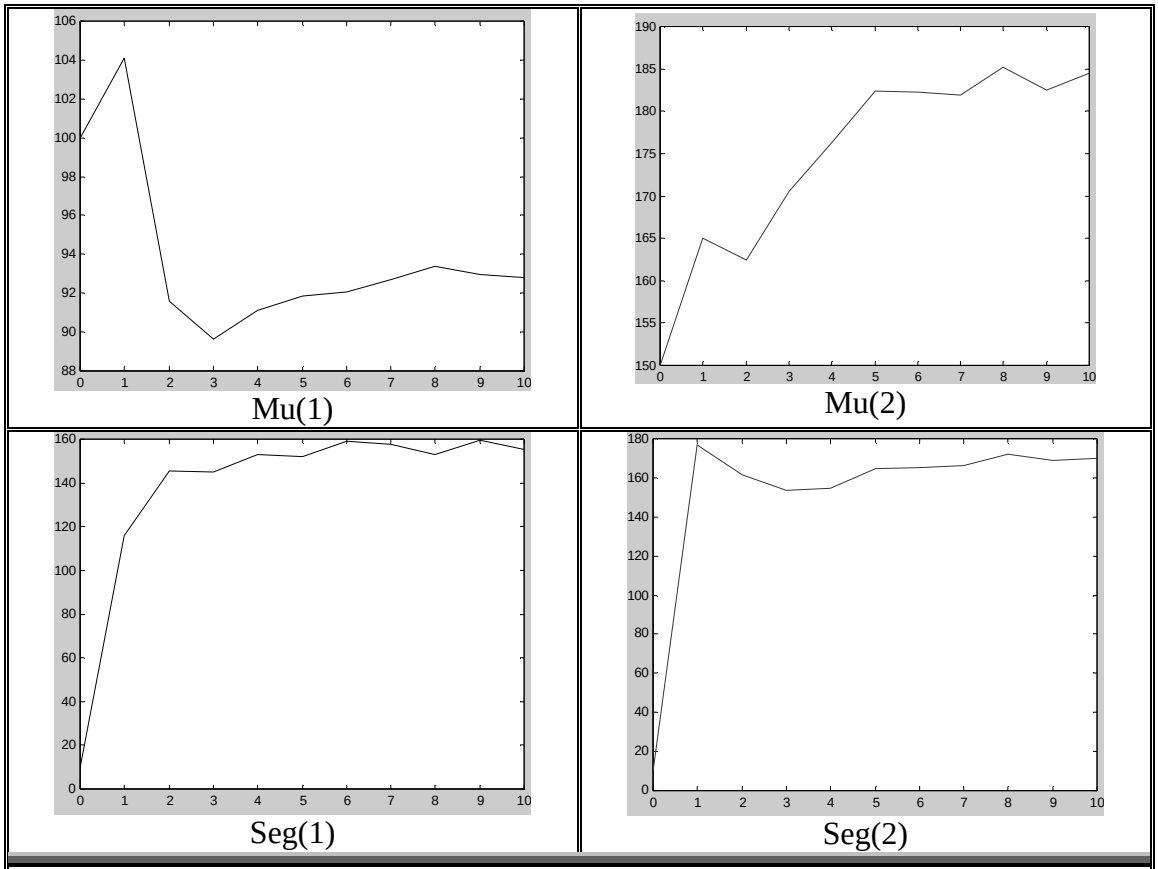


Fig (3.14) *: Evolution des moyennes et des variances lors de la segmentation (2-2) en fonction du nombre d'itérations*

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

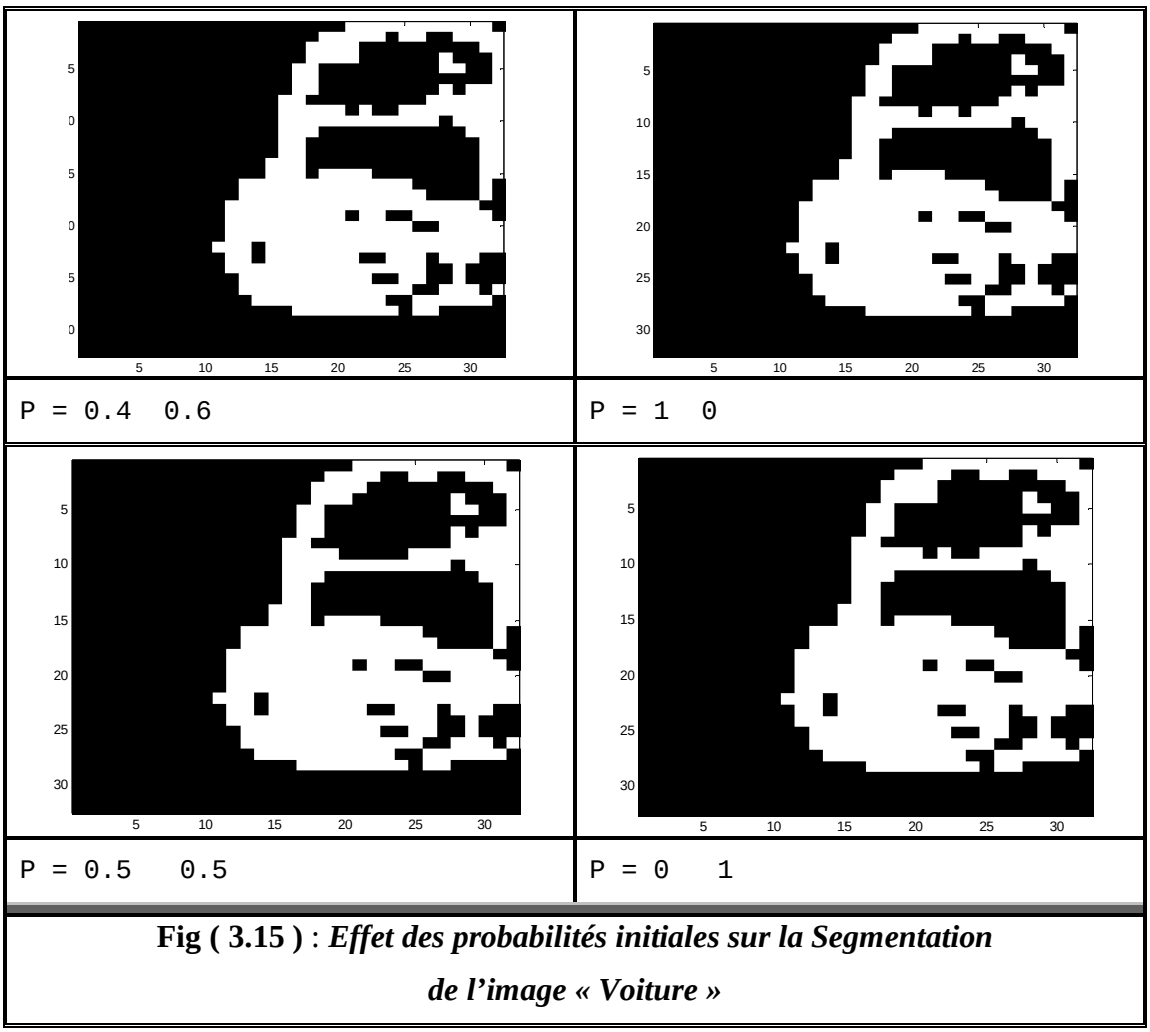


Fig (3.15) : ***Effet des probabilités initiales sur la Segmentation de l'image « Voiture »***

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;

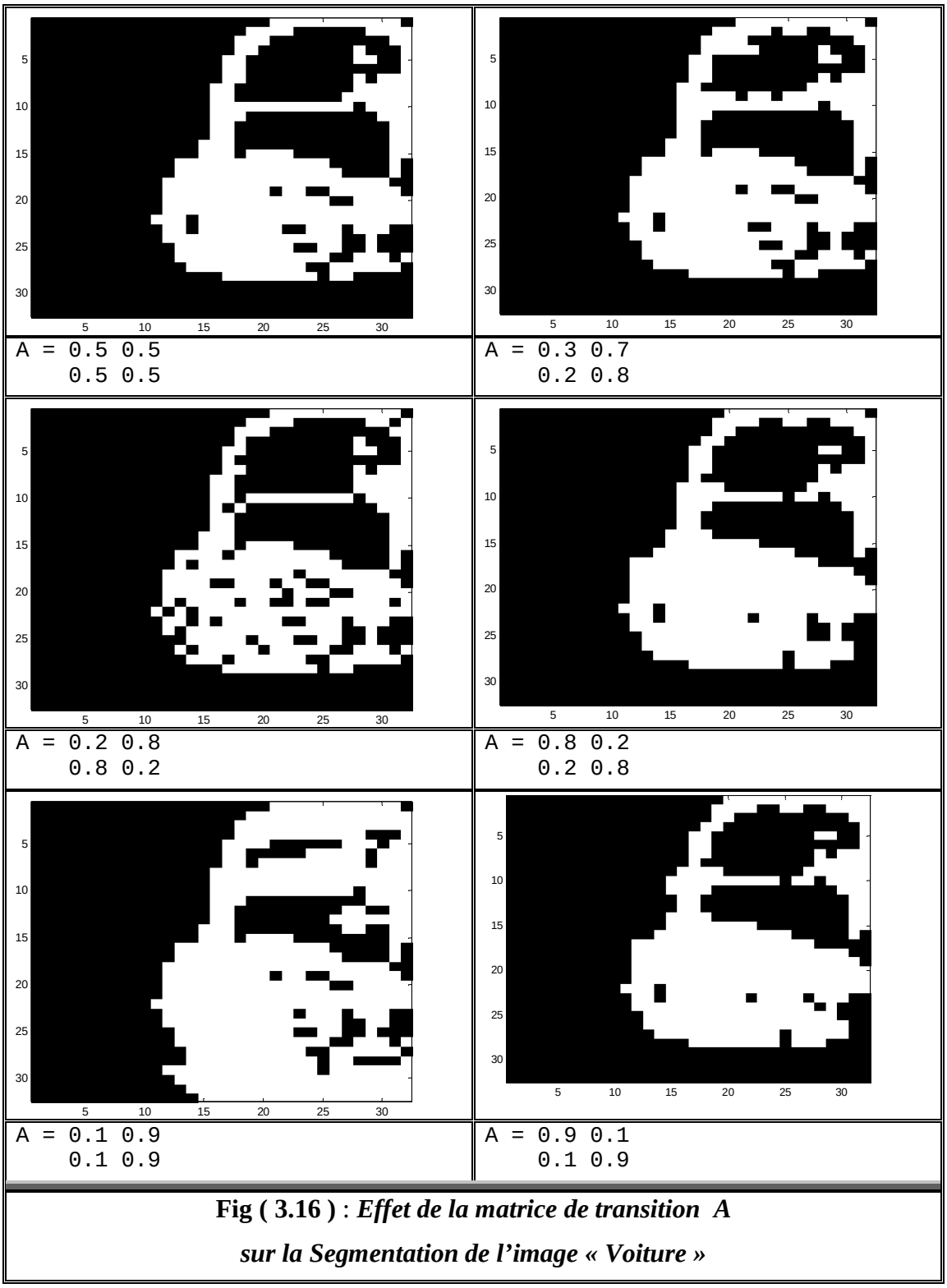


Fig (3.16) : ***Effet de la matrice de transition A sur la Segmentation de l'image « Voiture »***

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.