TP 2 : Chaînes de Markov cachées et segmentation d'image

Probabilistic Models and Machine Learning

Ambroise LAROYE–LANGOUËT

Télécom Paris

25 octobre 2025

1 Introduction

Ce TP est consacré à l'étude des chaînes de Markov cachées et à leur application à la segmentation d'images. La segmentation consiste à séparer une image ou signal en régions homogènes. Ce problème peut être formulé comme la recherche de variables cachées à partir de données observées.
L'approche bayésienne fournit la relation probabiliste entre le phénomène caché X et le phénomène observé Y. L'objectif est d'estimer la meilleure configuration de X de Y, selon un critère d'optimalité.
Dans le précédent TP nous avions vu 2 estimateurs : MAP (Maximum A Posteriori) et MPM (Mode des Marginales a Posteriori). Dans ce TP, on se concentre sur le critère **MPM**, plus adapté à la segmentation pixel par pixel.

2 Les chaînes de Markov

2.1 Principe

On considère un processus caché

$$X = (X_1, X_2, \ldots, X_N)$$

qui prend ses valeurs dans l'ensemble fini de classes $\Omega = \{\omega_1, \omega_2\}$, et un processus observable

$$Y = (Y_1, Y_2, \ldots, Y_N) \texttt{ à valeur dans } \mathbb{R}$$

Le couple (X, Y) est une **chaîne de Markov cachée**, alors on a :

$$p(x_1, \ldots, x_N) = p(x_1) \prod_{n=2}^{N} p(x_n|x_{n-1}),$$

Y_n conditionnellement indépendantes, et dépendent uniquement de l'état caché X_n :

$$p(y_1, \ldots, y_N|x_1, \ldots, x_N) = \prod_{n=1}^{N} p(y_n|x_n).$$

Loi jointe du modèle :

$$p(x, y) = p(x_1)p(y_1|x_1) \prod_{n=2}^{N} p(x_n|x_{n-1})p(y_n|x_n).$$

Dans ce TP, on suppose que la loi d'observation est gaussienne :

$$p(y_n|x_n = \omega_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(y_n - m_i)^2}{2\sigma_i^2}\right),$$

2.1.1 Forward

L'algorithme **Forward** permet de calculer de manière récursive les probabilités a posteriori partielles.
Initialisation :

$$\alpha_1(x_1) = p(x1)p(y_1|x_1)$$

$$\alpha_n(i) = p(y_1, \ldots, y_n, X_n = \omega_i).$$

Induction : $\forall n = 1, \ldots, N-1$

$$\alpha_{n+1}(x_{n+1}) = p(y_{n+1}|X_{n+1} = \omega_i) \sum_{x'_n \in \Omega} \alpha_n(x'_n) p(x_{n+1}|x'_n)$$

$p(x_{n+1}|x'_n) = p(X_n = \omega_i|X_{n-1} = \omega_j)$ sont les coefficients de la matrice de transition A.

2.1.2 Backward

L'algorithme **Backward** permet de calculer de manière récursive les quantités $\forall n\ \beta_n(x_n) = p(y_{n+1}, \ldots, y_N|x_n)$ qui peuvent être calculées de manière récursive par :
Initialisation :

$$\beta_N(x_N) = 1$$

$$\beta_n(i) = p(y_{n+1}, \ldots, y_N|X_n = \omega_i).$$

Induction : $\forall n = 1, \ldots, N-1$,

$$\beta_n(x_n) = \sum_{x'_n \in \Omega} \beta_{n+1}(x'_{n+1}) p(x'_{n+1}|x_n) p(y_{n+1}|x'_{n+1})$$

2.1.3 Rescaling

$\forall n\ \beta_n(x_n) \leq 1$ `et` $\alpha_n(x_n) \leq 1$. Ces quantités peuvent être très petite, car le calcul récursif des forwards et backwards implique un produit de N termes inférieur à 1.
Quand N est grand, les forwards et backwards vont prendre des valeurs très faible. Et ces valeurs risquent d'être assimilées à 0 par l'ordinateur.
On fait donc une **normalisation (rescaling)** à chaque itération :

$$\alpha_n(i) \leftarrow \frac{\alpha_n(i)}{\sum_{k=1}^K \alpha_n(k)}.$$

$$\beta_n(i) \leftarrow \frac{\beta_n(i)}{\sum_{k=1}^K \beta_n(k)}.$$

2.1.4 Estimation MPM (Mode des Marginales a Posteriori)

Le critère **MPM** (Mode des Marginales a Posteriori) consiste à estimer, pour chaque site n, la classe la plus probable sachant l'ensemble des observations :

$$\hat{x}_n = \arg\max_i P(X_n = \omega_i|Y).$$

En utilisant les résultats des algorithmes Forward et Backward, on a :

$$P(X_n = \omega_i|Y) = \frac{\alpha_n(i)\beta_n(i)}{\sum_{k=1}^K \alpha_n(k)\beta_n(k)}.$$

Le critère MPM permet de segmenter le signal en choisissant, pour chaque point, la classe la plus probable selon le modèle de chaîne de Markov cachée.

2.2 Résultats

Nous choisissons les matrices de transition suivantes :

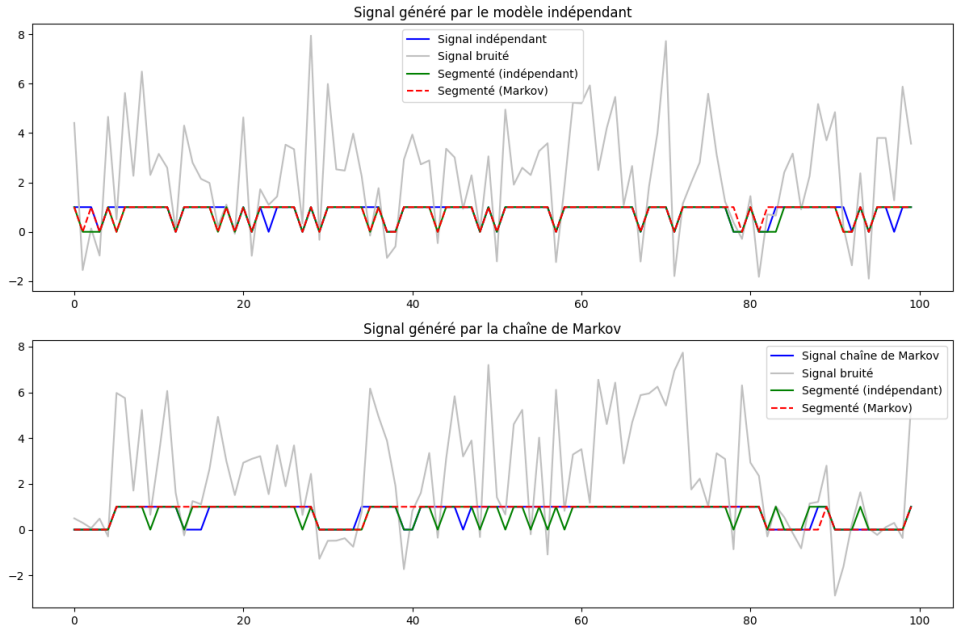
$$A_1 = \begin{pmatrix} 0.9 & 0.1 \\ 0.3 & 0.7 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0.6 & 0.4 \\ 0.2 & 0.8 \end{pmatrix}, \quad A_3 = \begin{pmatrix} 0.1 & 0.9 \\ 0.5 & 0.5 \end{pmatrix}.$$

Matrice	**Bruit** $(m_1, m_2, \sigma_1, \sigma_2)$	**Taux d'erreur**
A_1	(0, 3, 1, 2)	0.15
A_1	(1, 1, 1, 5)	0.16
A_1	(0, 1, 1, 1)	0.18
A_2	(0, 3, 1, 2)	0.49
A_2	(1, 1, 1, 5)	0.53
A_2	(0, 1, 1, 1)	0.47
A_3	(0, 3, 1, 2)	0.47
A_3	(1, 1, 1, 5)	0.51
A_3	(0, 1, 1, 1)	0.47

TABLE 1 – Taux d'erreur obtenus pour chaque matrice de transition et configuration de bruit.

Matrice A_1 : Le taux d'erreur est le plus faible pour le bruit 1. Le bruit (0, 1, 1, 1) donne le taux d'erreur le plus élevé, sans doute lié aux variances égales et moyennes rapprochées.
Matrice A_2 et A_3 : Les taux d'erreur sont plus élevés que pour A_1, avec un maximum de 0.53 pour le bruit 2. Les matrices A_2 et A_3 rendent la segmentation plus difficile. On pourrait expliquer cela par le fait que les matrices ont des **probabilités de transitions plus élevé entre états**. La segmentation serait donc plus difficile lorsque les transitions sont nombreuses, la séquence devient moins corrélée dans le temps et les observations successives sont plus indépendantes. Le modèle MPM dispose de moins d'information temporelle, ce qui augmente le taux d'erreur de segmentation.

Signal généré par le modèle indépendant et Chaîne de Markov :



Pour n = 1000 :

Type de signal	Modèle indépendant	Chaîne de Markov
Signal indépendant	0.16	0.54
Signal markovien	0.16	0.49

TABLE 2 – Comparaison des taux d'erreur MPM.

Cas signal indépendant :
La segmentation avec le modèle indépendant donne un très faible taux d'erreur. Ce qui était attendu puisqu'il est généré selon le même modèle.
La segmentation avec une chaîne de Markov donne un taux d'erreur plus élevé. L'utilisation d'un modèle de chaîne de Markov pour segmenter un signal généré par un modèle indépendant introduit des erreurs supplémentaires.

Cas signal markovien :
La segmentation avec le modèle indépendant donne un taux d'erreur très faible et le même que le cas avec le signal indépendant. Suggérant que les dépendances temporelles introduites par la chaîne de Markov n'ont pas d'impact sur la segmentation.
En revanche avec la chaîne de Markov le taux d'erreur reste très élevé et proche de celui avec le modèle indépendant. On aurait pu penser que en utilisant les chaîne de Markov nous allions largement améliorer, et donc diminuer le taux d'erreur. Une explication possible est le fait que nous avons **imposé les paramètres** et notamment la matrice de transition. Et ces paramètres ne sont pas forcément adaptés à notre cas.

3 Application à la segmentation d'images

3.1 Principe

3.1.1 Algorithme Stochastic Expectation Maximisation (SEM)

L'algorithme **SEM** est une méthode d'estimation itérative permettant d'estimer les paramètres d'un modèle probabiliste lorsque certaines variables sont cachées. Il repose sur l'idée d'utiliser une simulation des variables latentes à partir de leur loi a posteriori, plutôt que de calculer leur espérance (algorithme EM). Contrairement à l'algorithme EM qui maximise l'espérance du log-vraisemblance, SEM simule les variables cachées à chaque itération.
On dispose :

— d'un estimateur $\hat{\theta}(x, y)$ des paramètres à partir des données complètes (X, Y) ;

— pour tout θ, d'une méthode de simulation de X selon la loi a posteriori $p(x|y, \theta)$.

Algorithme SEM :

1. On fixe une valeur initiale :

$$\theta^{(0)} = \left(\pi^{(0)}, \mu_1^{(0)}, \sigma_1^{(0)}, \mu_2^{(0)}, \sigma_2^{(0)}\right)$$

où :

— $\pi^{(0)}$ est la probabilité a priori d'appartenance à la classe ω_1 ;

— $\mu_i^{(0)}$ et $\sigma_i^{(0)}$ sont respectivement la moyenne et l'écart-type initiaux des classes.

2. Pour $k = 0, \ldots, K-1$, on effectue les étapes suivantes :

(a) Calcul de la loi a posteriori $p(x|y, \theta^{(k)})$.
Sous l'hypothèse d'indépendance conditionnelle, on a :

$$p(x|y, \theta^{(k)}) = \prod_{n=1}^{N} p(x_n|y_n, \theta^{(k)}),$$

et pour tout n :

$$p(x_n = \omega_1|y_n, \theta^{(k)}) = \frac{\pi^{(k)} f_1(y_n; \mu_1^{(k)}, \sigma_1^{(k)})}{\pi^{(k)} f_1(y_n; \mu_1^{(k)}, \sigma_1^{(k)}) + (1-\pi^{(k)}) f_2(y_n; \mu_2^{(k)}, \sigma_2^{(k)})},$$

$$p(x_n = \omega_2|y_n, \theta^{(k)}) = \frac{(1-\pi^{(k)}) f_2(y_n; \mu_2^{(k)}, \sigma_2^{(k)})}{\pi^{(k)} f_1(y_n; \mu_1^{(k)}, \sigma_1^{(k)}) + (1-\pi^{(k)}) f_2(y_n; \mu_2^{(k)}, \sigma_2^{(k)})},$$

où f_i désigne la densité gaussienne :

$$f_i(y_n; \mu_i, \sigma_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(y_n - \mu_i)^2}{2\sigma_i^2}\right).$$

(b) Simuler une séquence $X^{(k)} = (x_1^{(k)}, \ldots, x_N^{(k)})$ selon la loi $p(x|y, \theta^{(k)})$.
Cela revient à tirer chaque $x_n^{(k)}$ aléatoirement selon les probabilités :

$$x_n^{(k)} \sim \text{Bernoulli}\left(p(x_n = \omega_1|y_n, \theta^{(k)})\right).$$

(c) Mise à jour : en considèrant la séquence simulée $x^{(k)}$ on a :

$$\pi^{(k+1)} = \frac{1}{N} \sum_{n=1}^{N} p(x_n = \omega_i|y_n, \theta^k),$$

$$\mu_i^{(k+1)} = \frac{\sum_{n=1}^N p(x_n = \omega_i | y_n, \theta^k) y_n}{\sum_{n=1}^N p(x_n = \omega_i | y_n, \theta^k)},$$

$$(\sigma_i^{(k+1)})^2 = \frac{\sum_{n=1}^N p(x_n = \omega_i | y_n, \theta^k)(y_n - \mu_i^{(k+1)})^2}{\sum_{n=1}^N p(x_n = \omega_i | y_n, \theta^k)}$$

3.1.2 Cas des Chaînes de Markov Cachées

Dans le cas d'un modèle de Chaîne de Markov Cachée, les variables cachées $X = (X_1, \ldots, X_N)$ ne sont plus supposées indépendantes conditionnellement, mais reliées entre elles par une structure markovienne.
On se donne une valeur initiale :

$$\theta^{(0)} = \left(\pi^{(0)}, A^{(0)}, \mu_1^{(0)}, \sigma_1^{(0)}, \mu_2^{(0)}, \sigma_2^{(0)}\right)$$

Ensuite, pour $k = 0, \ldots, K-1$:

1. **Calcul de la loi a posteriori** $p(x|y, \theta^{(k)})$.
   Ici, $p(x|y, \theta^{(k)})$ est une **loi markovienne**, et s'écrit sous la forme :

$$p(x|y, \theta^{(k)}) = p(x_1|y, \theta^{(k)}) \prod_{n=2}^N p(x_n|x_{n-1}, y, \theta^{(k)}).$$

$$p(x_1|y, \theta^{(k)}) = \frac{p(x_1)p(y_1|x_1)\beta_1(x_1)}{\sum_{x_1} p(x_1)p(y_1|x_1)\beta_1(x_1)},$$

   et pour tout $n = 2, \ldots, N$:

$$p(x_n|x_{n-1}, y, \theta^{(k)}) = \frac{p(x_n|x_{n-1})p(y_n|x_n)\beta_n(x_n)}{\beta_{n-1}(x_{n-1})}.$$

2. Simuler une séquence $X^{(k)} = (x_1^{(k)}, \ldots, x_N^{(k)})$ selon la loi $p(x|y, \theta^{(k)})$.
   — $x_1^{(k)} \sim p(x_1|y, \theta^{(k)})$,
   — Puis, pour $n = 2, \ldots, N$, tirer $x_n^{(k)} \sim p(x_n|x_{n-1}^{(k)}, y, \theta^{(k)})$.
3. Mise à jour des paramètres :

$$\pi_i^{(k+1)} = \frac{1}{N} \sum_{n=1}^N \mathbb{1}_{\{x_n^{(k)} = \omega_i\}},$$

$$a_{ij}^{(k+1)} = \frac{\sum_{n=2}^N \mathbb{1}_{\{x_{n-1}^{(k)} = \omega_i, x_n^{(k)} = \omega_j\}}}{\sum_{n=2}^N \mathbb{1}_{\{x_{n-1}^{(k)} = \omega_i\}}},$$

$$\mu_i^{(k+1)} = \frac{\sum_{n=1}^{N-1} \mathbb{1}_{x_n = \omega_i} y_n}{\sum_{n=1}^{N-1} \mathbb{1}_{x_n = \omega_i}},$$

$$(\sigma_i^{(k+1)})^2 = \frac{\sum_{n=1}^N \mathbb{1}_{x_n = \omega_i}(y_n - \mu_i^{(k+1)})^2}{\sum_{n=1}^N \mathbb{1}_{x_n = \omega_i}}$$

3.2 Résultats

3.2.1 Fonction `calc_param_SEM_mc`

$$p(x_1, \ldots, x_N) = p(x_1) \prod_{n=2}^{N} p(x_n|x_{n-1}),$$
$$p(y_n|X_n = \omega_i) = \mathcal{N}(y_n; m_i, \sigma_i^2).$$

Les paramètres à estimer sont :

$$\theta = \{p = [p_1, p_2],\ A = [a_{ij}],\ m_1, m_2,\ \sigma_1, \sigma_2\}.$$

Probabilités a posteriori Les probabilités marginales a posteriori sont obtenues à partir du produit normalisé :

$$\gamma_n(i) = \frac{\alpha_n(i)\,\beta_n(i)}{\sum_{k=1}^{2} \alpha_n(k)\beta_n(k)}.$$

"probabilité d'être dans l'état ω_i au temps n, sachant l'ensemble des observations Y" :

$$\gamma_n(i) = P(X_n = \omega_i|Y, \theta).$$

Probabilités conjointes

$$P(X_t = \omega_i, X_{t+1} = \omega_j|Y, \theta) = \frac{\alpha_t(i)\,a_{ij}\,f_j(y_{t+1})\,\beta_{t+1}(j)}{\sum_{i',j'} \alpha_t(i')a_{i'j'}f_{j'}(y_{t+1})\beta_{t+1}(j')}.$$

Estimation des paramètres

— **Distribution initiale :**

$$p_i^{\text{new}} = \gamma_1(i)$$

— **Matrice de transition :**

$$a_{ij}^{\text{new}} = \frac{\sum_{t=1}^{N-1} P(X_t = \omega_i, X_{t+1} = \omega_j|Y, \theta)}{\sum_{t=1}^{N-1} \gamma_t(i)}$$

— **Moyennes et variances conditionnelles :**

$$m_i^{\text{new}} = \frac{\sum_{n=1}^{N} \gamma_n(i) y_n}{\sum_{n=1}^{N} \gamma_n(i)}, \qquad (\sigma_i^{\text{new}})^2 = \frac{\sum_{n=1}^{N} \gamma_n(i)(y_n - m_i^{\text{new}})^2}{\sum_{n=1}^{N} \gamma_n(i)}.$$

3.2.2 Résultats des paramètres de segmentation

Dans cette section nous nous intéressons à 3 images :

1. alfa2.bmp
2. veau2.bmp
3. zebre2.bmp

Et pour ces 3 images nous utiliserons 3 bruits gaussiens différents, qui sont ceux vu précédemment et 2 méthodes de segmentation (Indépendant et Markov)
Nous étudions les taux d'erreur et les paramètres estimés après SEM. Chaque image a été normalisée, puis bruitée selon les trois configurations gaussiennes. Et le SEM exécuté sur 10 itérations. Même avec le rescaling nous avons rencontré des problèmes lorsque nous augmentions le nombre d'itération car les valeurs était trop faible. Peut être que le rescaling n'a pas été bien fait dans le code ou bien c'est un résultat attendu puisque en augmentant N nous diminuons alpha et beta. En augmentant le nombre d'itérations légèrment, il n'y avait pas de différence significative, et le temps de cacul d'une epoch est long (environ 1 min).

Image	μ_1	μ_2	σ_1	σ_2	**Err Markov**	**Err Ind**	μ_1^{mc}	μ_2^{mc}	μ_1^{gm}	μ_2^{gm}
alfa2.bmp	0	3	1	2	0.2071	0.2068	-0.1638	0.7827	-0.1655	0.7330
	1	1	1	5	0.3438	0.0203	0.0016	1.0249	0.1021	1.5588
	0	1	1	1	0.2213	0.2184	0.1633	-0.7028	0.1665	-0.6294
veau2.bmp	0	3	1	2	0.1009	0.1281	0.1006	-1.2934	0.0718	-0.7879
	1	1	1	5	0.2113	0.2115	0.0000	1.0063	0.0000	1.0082
	0	1	1	1	0.1021	0.1363	-1.2401	0.0997	-0.5843	0.0573
zebre2.bmp	0	3	1	2	0.1447	0.1513	-1.0666	0.1082	-0.8720	0.0992
	1	1	1	5	0.2559	0.2472	1.0038	0.0000	1.0264	0.0001
	0	1	1	1	0.1472	0.1589	-1.0277	0.1011	-0.6723	0.0823

TABLE 3 – Résultats de segmentation avec différents bruits gaussiens et les 3 images

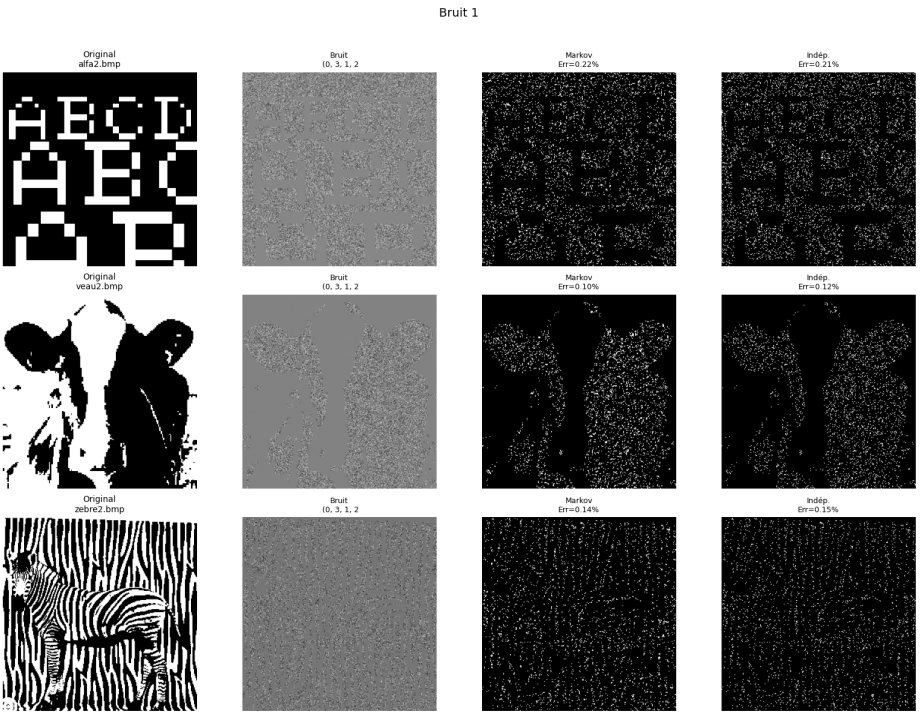


FIGURE 1 – Segmentation pour les 3 images et le bruit 1

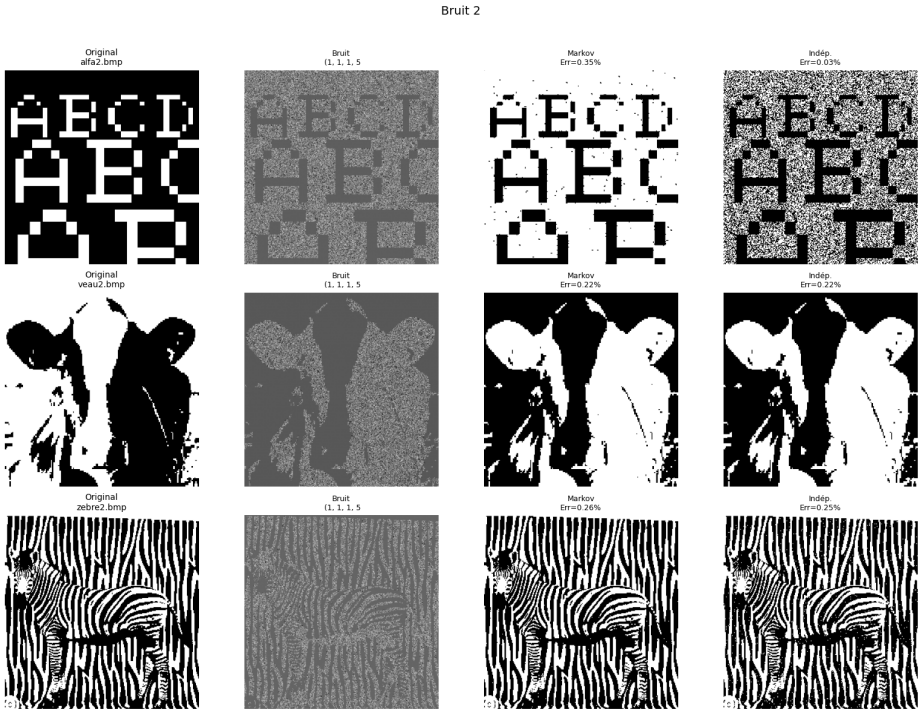


FIGURE 2 – Segmentation pour les 3 images et le bruit 2

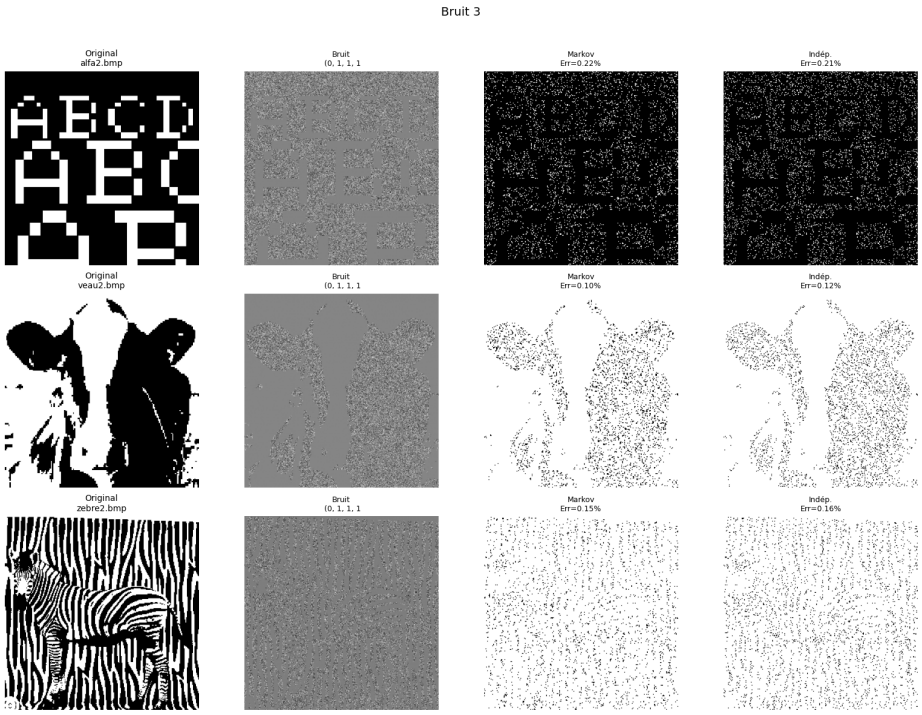


FIGURE 3 – Segmentation pour les 3 images et le bruit 3

Les valeurs estimées des moyennes sont proches de 0 et 1 (parfois inversées ou légèrement décalées).
Nous pouvons noter que pour le bruit (1,1,1,5), les deux classes ont les même moyennes, donc la séparation dépend uniquement de la variance. Et le modèle indépendant semble mieux gérer ce cas, surtout sur la figure alfa2.

Visuallement :
Bruit 1 : Les images sont globalement mal segmentées et difficiles à interpréter, aussi bien pour le modèle indépendant que pour le modèle markovien.
Bruit 2 : Les motifs de l'image zèbre sont presque parfaitement retrouvés, le modèle capturant efficacement les variations texturées. Et l'images des lettres est très bien retrouvé contrairement au modèle indépendant qui a plus de mal.
Bruit 3 : La segmentation de l'image veau est correcte mais inversée (classes échangées), ce qui explique les différences de signe dans les moyennes estimées.
Pour (0,1,3,2) les classes sont bien séparées, donc les deux modèles réussissent presque parfaitement avec un taux d'erreur très faible voire nul. Et pour (0,1,1,1) il y a un faible contraste entre les classes et Markov semble avoir un léger avantage.
Markov semble être meilleur quand les classes sont **peu contrastées** (en exploitant les relations locales entre pixels ?).
Et le cas **Indépendant** est meilleur lorsque les classes sont bien séparées, et robuste avec des bruits de même moyenne.

On remarque que pour l'image alfa2 et pour le bruit 2 avec le modèle indépendant, l'erreur est très faible, 0.02%. Or en observant l'image elle n'est pas reconstitué à la perfection, du bruit persiste autour des lettres qui sont bien retrouvé. Les lettres sont même plus lisible avec Markov. Nous pouvons noter également que l'image alfa2 est formée de lettres noires sur fond blanc où les contours sont très nets et droit, sans "texture" comme on aurait sur une image plus réaliste. Les traits des lettres sont net et démarqués du fond. Les changements de classe sont plus "abrupts". Les images “zèbre” et “veau” fonctionnent bien malgré leur structures complexe avec markov et le cas indépendant. Cependant **"alfa" qui a une structure plus discontinue est mieux géré avec Markov** qui prend plus en compte les pixel voisin et donc **génère moins d'erreur lorsque l'image est toute blanche ou noire sur un voisinage**.

En conclusion, ce TP a permis de comprendre et d'implémenter les méthodes de segmentation basées sur les chaînes de Markov cachées et les modèles gaussiens. Nous avons observé que le modèle le modèle markovien améliore très légèrement la segmentation pour les images texturées, mais dégrade parfois les images à fortes discontinuités spatiales. Cependant, pour les images à contours nets, le modèle indépendant reste plus adapté.