Décodage conjoint source-canal avec estimation en ligne de la source

C. Weidmann* P. Siohan

IRISA-INRIA

Campus universitaire de Beaulieu, 35042 Rennes cedex, France

{claudio.weidmann, pierre.siohan}@irisa.fr

Résumé

Le décodage conjoint source-canal permet de réduire le taux d'erreur au décodeur en exploitant la redondance résiduelle encore présente après compression de la source. Pour cela, le décodeur a besoin d'informations a priori *sur la statistique de la source ; de nombreux travaux dans le domaine font l'hypothèse qu'elle soit connue au décodeur. Cette hypothèse étant souvent peu réaliste en pratique, on étudie des méthodes qui intègrent l'estimation de la source au décodeur. Comme exemple on applique une des méthodes proposées au décodage des vecteurs-mouvement de séquences vidéo codées avec la norme H.263++ et transmises sur un canal à évanouissement.*

Mots clefs

Décodage conjoint source-canal, décodage itératif, estimation en ligne.

1 Introduction

La plupart des normes de compression pour images, audio ou vidéo utilisent des codes à longueur variable (VLC) pour comprimer des coefficients obtenus par une transformation du signal. Comme modèle pour la transmission de sources codées VLC on considère donc un système composé d'une source discrète, d'un codeur source spécifié par une table de mots de code binaires de longueur variable, d'un entrelaceur π à niveau bit et d'un codeur canal de type convolutif. Un élément clef du décodeur itératif (voir figure 1) est le décodeur VLC à entrées et sorties souples (*soft-in, soft-out :* SISO), qui utilise les informations *a priori* sur la statistique de la source pour maximiser la probabilité postérieure de la séquence décodée.

On va d'abord décrire la méthode de décodage VLC-SISO dans un cadre BCJR [1] (le même algorithme peut aussi être obtenu dans un cadre de réseaux Bayésiens [2]) et les différentes approches pour l'estimation en ligne de la source. Ensuite on adressera les détails de l'application à la norme H.263++ et on présentera des premiers résultats de simulation.

*Ce projet a bénéficié d'un financement du ministère de la recherche Français sous le contrat COSOCATI (http ://www.telecom.gouv.fr/rnrt/index_net.htm)

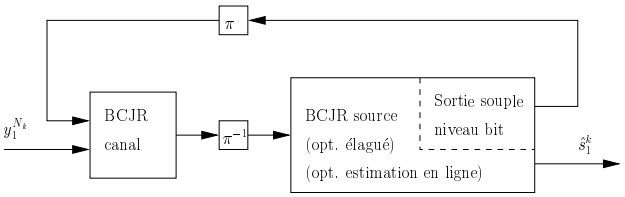

Figure 1 – *Décodage source-canal itératif.*

2 Décodage souple des VLC

On considère une séquence de k symboles $S_1, S_2, \ldots, S_k$ émis par une source discrète Markovienne d'ordre 1 spécifiée par les probabilités conditionnelles $P(s_t|s_{t-1})$ et marginales $P(s)$. La séquence de symboles est codée dans un train binaire $X_1^{N_k}$ en utilisant un code à longueur variable (VLC) qui représente le symbole s par le mot binaire $c_{s,1}, c_{s,2}, \ldots, c_{s,\ell(s)}$, dont $\ell(s)$ est la longueur. Enfin le train binaire est transmis à travers un canal à entrée binaire sans mémoire avec probabilité de transition $p(Y|X)$. Dans le cadre du décodage concaténé on utilisera la sortie souple du décodeur canal, soit les mesures $P(X_n=0,1|Y_n=y_n)$. Notons que la longueur N_k du train binaire est une variable aléatoire. Pour le décodage il est avantageux de définir tous les points de segmentation $N_t = \sum_{j=1}^{t} \ell(S_j)$, $t = 1, \ldots, k$, de manière que N_t pointe à la fin du t-ème mot de code. Si on fait l'hypothèse que le décodeur connaît k et N_k, il est possible de construire un treillis de segmentation, comme le montre la figure 2 pour le VLC $\mathcal{C} = \{0, 10, 11\}$ et $k = 6$, $N_k = 9$. Cette hypothèse est applicable dans le cas du *data partitioning* du train binaire de H.263++ (voir section 3). Chaque chemin sur le treillis correspond à une séquence que la source aurait pu émettre (il y a des transitions parallèles pour les mots de code de la même longueur). Pour tenir compte du caractère Markovien de la source, on *augmente* le treillis en associant le dernier symbole S_t émis à chaque état $N_t = n$ dans le treillis de segmentation. Ceci est indiqué par les nœuds superposés dans la figure 2, qui correspondent aux différents derniers symboles possibles.

Pour décoder les k symboles source on peut soit maximiser la probabilité postérieure (APP) de la séquence (critère MAP ou MAP-séquence), $\hat{s}_1^k = \arg\max_{s_1^k} P(s_1^k|y_1^{N_k})$,

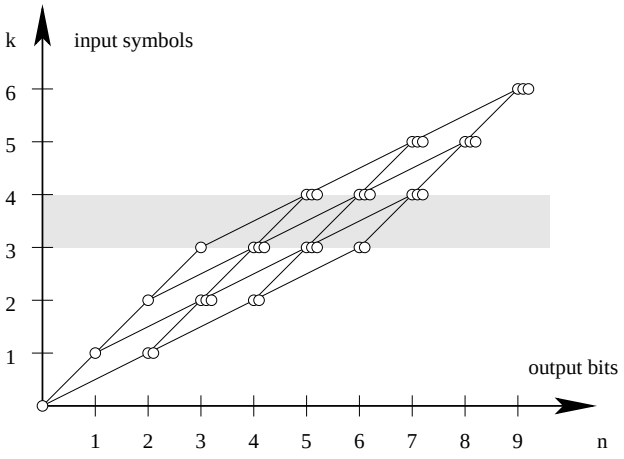

Figure 2 – *Treillis de segmentation.*

soit l'APP symbole, $\hat{s}_t = \arg\max_{s_t} P(s_t|y_1^{N_k})$ (critère *maximum posterior marginals*, MPM ou MAP-symbole).
Même si en général la séquence des symboles décodés au MPM va produire un train binaire codé VLC de longueur non égale à N_k, ici on se concentre sur cette méthode, car elle donne des meilleurs résultats en terme de taux d'erreur symbole (SER).
La méthode de référence pour le décodage MPM est le procédé *forward-backward* ou algorithme BCJR [3] appliqué au treillis de segmentation augmenté. La différence principale par rapport à un BCJR classique est que les points de segmentation N_t sont des variables d'état et donc les transitions d'état à état dépendent d'un nombre variable de symboles du canal. Mais ceci ne pose pas de problèmes particuliers, car seulement les chemins de la même longueur peuvent se joindre dans un état. On détaille brièvement les fonctions de probabilité qui sont à la base du BCJR :

$$\alpha_t(n,s) = \Pr\{N_t{=}n, S_t{=}s, y_1^n\} \quad (1)$$

$$\beta_t(n,s) = \Pr\{y_{n+1}^{N_k} \mid N_t{=}n, S_t{=}s\} \quad (2)$$

$$\begin{aligned}\gamma_t(n',s';n,s) &= \Pr\{S_t{=}s, N_t{=}n, y_{n'+1}^{n} \mid \\ &\qquad N_{t-1}{=}n', S_{t-1}{=}s'\} \quad (3)\\ &= \delta_{n-n'-\ell(s)}\, P(s|s') \textstyle\prod_{i=1}^{\ell(s)} p(y_{n'+i}|c_{s,i}).\end{aligned}$$

La passe avant du BCJR calcule les $\alpha_t(n,s)$ en utilisant la récursion $\alpha_t(n,s) = \sum_{n',s'} \alpha_{t-1}(n',s')\gamma_t(n',s';n,s)$, tandis que la passe arrière obtient les $\beta_t(n,s)$ par la récursion $\beta_t(n,s) = \sum_{n',s'} \beta_{t+1}(n',s')\gamma_{t+1}(n,s;n',s')$. $\alpha_0(0,s)$ sera initialisé avec la loi stationnaire de la source et $\beta_k(N_k,s)$ avec une loi uniforme. A chaque étage t du treillis les α_t et β_t peuvent être normalisés pour une meilleure stabilité numérique de l'algorithme. Le décodeur MPM procède en calculant les APP

$$\begin{aligned}&\Pr\{N_t{=}n, S_t{=}s \mid y_1^{N_k}\} \propto \\ &\qquad \lambda_t(n,s) = \alpha_t(n,s)\beta_t(n,s) \quad (4)\end{aligned}$$

pour décoder $\hat{s}_t$ de $(\hat{n}_t, \hat{s}_t) = \arg\max_{n,s} \lambda_t(n,s)$. (Les simulations montrent qu'il est mieux de décoder la paire (n_t, s_t) plutôt que de marginaliser sur n avant de maximiser.)
On remarque encore qu'un décodage MAP-séquence avec l'algorithme de Viterbi revient à calculer la récursion des α_t en substituant l'opération $\max$ à l'addition.
Un élément essentiel du décodage itéré canal-source est la sortie d'information souple du décodeur source. Celle-ci doit être à niveau bit et non pas symbole comme (4). Le passage au niveau bit s'avère très simple si l'on considère que (4) donne la probabilité que le mot de code pour s ait été émis aux positions $n-\ell(s)+1$ à n du train binaire. Il suffit donc d'accumuler ces probabilités au niveau bit en distinguant les positions des 1 et 0 dans les mots de code [2]. L'information extrinsèque réinjectée au décodeur canal est obtenue en divisant ces probabilités bit par les probabilités *a priori* à l'entrée du décodeur source.

2.1 Réduction de la complexité

La figure 2 montre que la taille du treillis et donc la complexité de l'algorithme est de l'ordre de $O(k^2)$, un coût qui pour des applications avec k dans les centaines devient vite prohibitif. Dans l'étude [4] on a proposé plusieurs approches pour réduire la complexité. Pour le présent article on utilise un élagage "horizontal" du treillis qui consiste à limiter la valeur de N_t à chaque instant t.

2.2 Estimation de la source

L'estimation de la loi régissant la source peut être partie intégrante du décodeur VLC-SISO ou n'intervenir qu'après décodage dur de la séquence. Cette dernière approche, généralement moins complexe, s'est révélée assez efficace pour l'estimation paramétrique d'une source AR-1 Gauss-Markov avec corrélation ρ (quantifiée et codée VLC), qu'on a étudié à titre d'exemple théorique [4]. Si on ne dispose pas d'un modèle paramétrique de la source, on peut utiliser la méthode de Baum-Welch qui estime directement les probabilités de transition par un histogramme "souple" (voir le tutoriel [5]). Les calculs se font directement à partir des variables du BCJR source, sans en augmenter sensiblement la complexité :

$$\hat{P}(s'|s) = \frac{\sum_t \xi_t(s;s')}{\sum_t \sum_{s'} \xi_t(s;s')} \quad (5)$$

$$\text{où} \quad \xi_t(s;s') = \frac{\sum_n \alpha_t \gamma_{t+1} \beta_{t+1}}{\sum_n \sum_s \sum_{s'} \alpha_t \gamma_{t+1} \beta_{t+1}}. \quad (6)$$

La loi estimée peut être utilisée pour itérer le décodage source seul ou l'ensemble source-canal. Selon le degré de stationnarité de la source on peut utiliser une fenêtre glissante ou un facteur d'oubli pour adapter et/ou affiner la loi estimée au cours de plusieures séquences.

3 Application à la norme H.263++

On a choisi d'appliquer le décodage VLC-SISO aux vecteurs-mouvement (VM) de la norme H.263++. Le mode

dit *data partitioning* spécifié dans l'annexe V [6] de la norme permet d'isoler les zones codant les VM du reste du train binaire, c.a.d. que le décodeur connaît effectivement et le nombre de symboles, k, et le nombre de bits, N_k, et donc la méthode VLC-SISO est applicable.

Le mode *data partitioning* stocke les données pour un nombre variable de macroblocs de 16×16 pixels dans des paquets ne dépassant pas une taille maximale fixée au moment du codage. Les VM $[v_1, v_2, \ldots, v_k]$ d'un paquet sont d'abord remplacés par la suite des différences $[v_1, v_2 - v_1, \ldots, v_k - v_{k-1}]$ et ensuite codés avec un VLC réversible (RVLC), ce qui permet de décoder à partir des deux extrémités du train binaire. Pour pouvoir reconstruire les VM à partir des différences en décodant par l'arrière, on rajoute le code RVLC pour le dernier VM, v_k, après celui de la dernière différence $v_k - v_{k-1}$.

Pour éviter que certains mots de synchronisation apparaissent dans la zone VM, la norme prévoit que deux codes pour "+1" soient toujours suivis d'un mot pour "0" qui doit être enlevé au décodeur. Dans nos simulations on n'a pas pris en considération cette contrainte, qui en principe devrait légèrement améliorer les performances. On remarque simplement que ce type de contrainte peut très bien être pris en compte par un modèle de source Markovien.

Les codes RVLC rajoutent généralement de la redondance par rapport à un VLC compact décodable dans un seul sens construit pour la même loi de source. Ils ont été introduits dans la norme H.263++ pour combattre la propagation d'erreurs typique des VLC, qui se manifeste surtout comme désynchronisation de l'horloge symbole par rapport au train binaire. Le cas de figure envisagée est une erreur binaire isolée, dont on espère limiter les conséquences en décodant des deux côtés. A des taux d'erreur plus importants, la redondance RVLC ne suffit plus pour éviter la propagation d'erreurs et, qui plus est, le phénomène est même amplifié par l'emploi du codage différentiel. Pour cette raison on a aussi étudié un système hors norme qu'on appellera *non-différentiel*, qui code directement les VM et non leurs différences. Dans ce cas on n'a pas besoin de rajouter le code du dernier VM pour permettre le décodage en arrière. Par l'absence du codage différentiel on s'attend à un moindre taux de compression, étonnamment ce n'est pas toujours le cas. Dans le mode non-différentiel on va d'ailleurs coder d'abord toutes les composantes horizontales des VM et ensuite les verticales, plutôt que de les entrelacer comme dans la norme. Cette approche permet l'utilisation de deux modèles Markovien pour ces composantes, alors que dans le cas normalisé la taille du modèle Markovien pour les vecteurs serait trop importante.

Le codage non-différentiel a un autre avantage de poids, qui est le fait que le décodeur va estimer directement les VM et non leur différences. Seulement dans ce cadre on peut récupérer les gains potentiels de la méthode MPM, car pour le codage différentiel, toute erreur d'estimation (MPM ou MAP) des différences va se propager de toute façon.

4 Résultats

Pour tester les méthodes proposées on a isolé les zones VM à la sortie d'un codeur H.263++ (taille maximale de paquet 512 octets, paramètre de quantification QP=13). Les VM (ou leurs différences) sont codés en RVLC et ensuite avec un code convolutif de rendement 1/2 (générateur $(35, 23)_8$). Le train binaire résultant est transmis par modulation binaire antipodale (BPSK) sur un canal de Rayleigh entrelacé avec estimation parfaite du canal au décodeur. L'élagage du treillis de décodage source limite le compteur bit à chaque instant symbole t, $\max\{N_t\} - \min\{N_t\} \leq 100$. L'estimation de la source utilise la méthode Baum-Welch avec deux itérations de décodage canal-source-estimation pour chaque paquet (on utilise *paquet* pour distinguer la séquence VM de la séquence vidéo). Deux modèles séparés sont estimés pour les composantes horizontales et verticales. La loi de la source est initialisée en source indépendante avec $P(s) \propto 2^{-\ell(s)}$ à chaque paquet, donc il n'y a pas de lissage de l'estimation au travers des paquets (mais pas non plus de propagation d'une éventuelle erreur).

On compare avec le décodage robuste des RVLC comme il est proposé pour la norme H.263++. Après un décodage dur du train binaire (par seuillage), le décodage RVLC procède des deux extrémités et s'arrête dès qu'il tombe sur un préfixe (ou suffixe) qui n'est pas dans la table de codes ; les positions non-décodées sont remplies avec des zéros.

La figure 4 montre les résultats pour la séquence "News" (QCIF, 299 trames). Etonnamment le codage standard (différentiel) utilise 5.22 bits/VM comparé à 4.26 bits/VM pour le non-différentiel. Donc la comparaison pénalise ce dernier, qui pourtant en sort gagnant (car la courbe "RVLC différentiel" devrait être déplacée de $10 \times \log_{10}(5.22/4.26) = 0.88$ dB à droite). La source VM de "News" est assez stationnaire, ce qui se traduit par des bonnes performances si on estime la loi par un histogramme sur toute la séquence et qu'on la suppose connue au décodeur. Dans la figure 3 on voit que l'estimation Baum-Welch par paquet reste entre la connaissance de la loi marginale et la connaissance de la loi de transition.

Les résultats pour "Foreman" (QCIF, 399 trames ; figure 5) montrent que pour une séquence avec mouvements plus complexes et moins stationnaires, il devient plus difficile d'obtenir un gain par rapport au décodage RVLC robuste (les débits sont comparables, différentiel 8.08 bits/VM et non-différentiel 8.17 bits/VM). A cause de la non-stationnarité, l'utilisation d'une loi estimée sur toute la séquence donne ici de très mauvais résultats (non montrés).

Finalement, les résultats pour "Flower" (CIF, 249 trames ; figure 6) sont dans la lignée de "News", sauf que le décodage robuste en non-différentiel est largement battu par le différentiel. Pour obtenir un débit global comparable, ici on a choisi de poinçonner le code convolutif à un taux de 2/3 pour le codage non-différentiel. Les débits sont $5.45/(1/2) = 10.9$ bits/VM en différentiel et $7.34/(2/3) = 11.0$ bits/VM en non-différentiel.

5 Conclusion

La méthode proposée permet d'obtenir des gains de performance respectables dans un cadre pratique. Des améliorations semblent possibles surtout en exploitant mieux la stationnarité locale (temporelle) pour affiner l'estimation de la source. L'utilisation de modèles paramétriques est aussi attractive, mais comporte le risque que la méthode devienne moins robuste.

Références

[1] R. Bauer et J. Hagenauer. Symbol-by-symbol MAP decoding of variable length codes. Dans *3rd ITG Conf. on Source and Channel Coding*, pages 111–116, Munich, Germany, Janvier 2000.

[2] A. Guyader, E. Fabre, C. Guillemot, et M. Robert. Joint source-channel turbo decoding of entropy-coded sources. *IEEE JSAC*, 19(9) :1680–1696, Septembre 2001.

[3] L. R. Bahl, J. Cocke, F. Jelinek, et J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. Inform. Theory*, IT-20 :284–287, Mars 1974.

[4] C. Weidmann. Reduced-complexity soft-in-soft-out decoding of variable-length codes. Submitted to ISIT 2003, Octobre 2002.

[5] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77 :257–286, Février 1989.

[6] Adam H. Li, Surin Kittitornkun, Yu-Hen Hu, Dong-Seek Park, et John Villasenor. Data partitioning and reversible variable length codes for robust video communications. Dans *Proc. Data Compression Conference*, pages 460–469, Snowbird, Utah, Mars 2000.

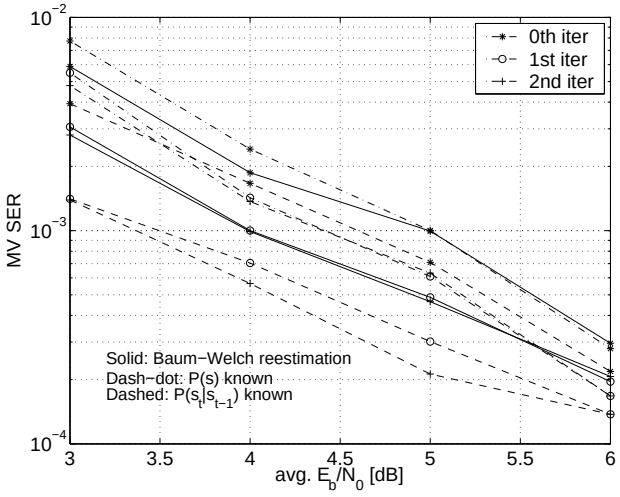

Figure 3 – *Comparaison entre décodage avec lois stationnaires connues et décodage avec réestimation Baum-Welch pour la séquence "News".*

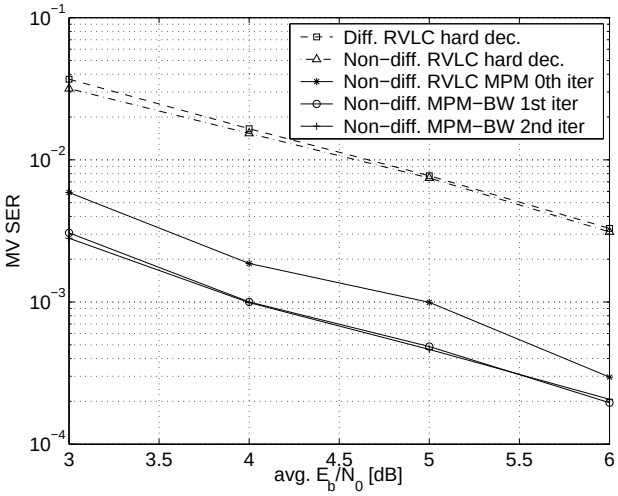

Figure 4 – *Résultats pour la séquence "News".*

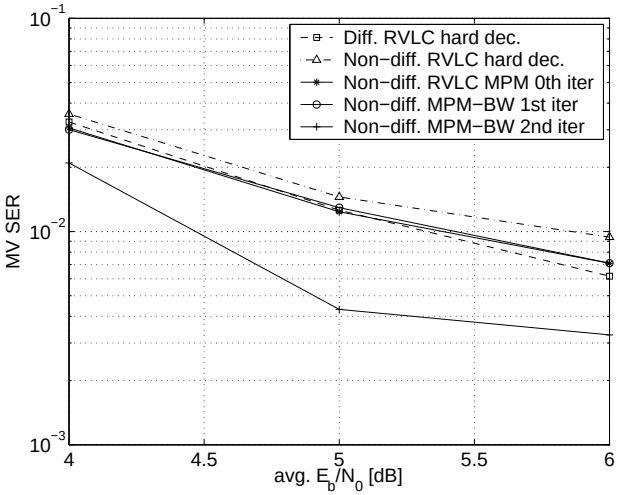

Figure 5 – *Résultats pour la séquence "Foreman".*

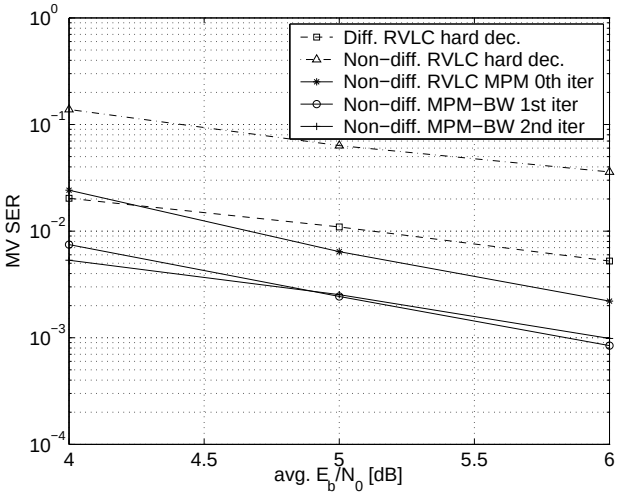

Figure 6 – *Résultats pour la séquence "Flower".*