Treillis à complexité réduite pour le décodage de codes à longueur variable

Reduced-complexity trellises for the joint source-channel decoding of variable-length encoded data

Gholam-Reza Mohammad-Khani, Chang-Ming Lee,
Michel Kieffer et Pierre Duhamel

LSS – CNRS – Supélec – Univ Paris-Sud,
Plateau de Moulon - F-91192 Gif-sur-Yvette Cedex
Tél. : 00 33 1 69 85 17 32 — Fax : 00 33 1 69 85 17 65
{khani,lee,kieffer,Pierre.Duhamel}@lss.supelec.fr

Manuscrit reçu le 24 octobre 2005

Résumé et mots clés

De nombreux algorithmes ont été proposés pour le décodage souple de données codées à l'aide de codes à longueur variable (CLV), la plupart travaillant avec des treillis. Pour un code réaliste, ces treillis sont très complexes à cause du nombre de mots de code à considérer. Cet article présente le principe d'un algorithme de regroupement de mots de CLV en un nombre minimal de classes, ce qui permet de réduire significativement la complexité des treillis utilisés pour le décodage souple de CLV. L'adaptation des algorithmes de décodage tels que SOVA ou BCJR à ce type de treillis est détaillée. Une illustration sur les CLV de la norme H.263+ est proposée ainsi qu'un exemple d'application à la localisation des frontières de blocs de texture H.263+.

Codes à longueur variable, décodage source-canal conjoint, estimation au sens du MAP, estimation au sens du maximum de vraisemblance.

Abstract and key words

Many trellis-based soft decoding techniques have been proposed for data encoded using variable-length codes (VLC). However, for actual VLC tables, these trellises are too complex to allow real-time soft decoding. This paper presents the principle of an algorithm for grouping VLC codewords into classes, which allows significant reductions of the complexity of the resulting trellises and of the soft decoding techniques. The adapation of decoding algorithms such as SOVA or BCJR to the reduced-complexity trellises is detailed. Illustrations are provided on the VLC table used for texture encoding in H.263+. The performance of a decoding technique for the localization of block frontiers in a bitstream generated by an H.263+ coder is also presented.

Decoding, joint source-channel decoding, MAP estimation, maximum likelihood decoding, maximum likelihood estimation, variable length codes.

Ce travail a été en partie financé par le Réseaux d'Excellence européen NEWCOM.

1. Introduction

Dans de nombreux standards de compression, tel que JPEG, H.263+ ou le mode de base de H264, une étape de compression à l'aide d'un code à longueur variable (CLV) des données est effectuée. Les trains binaires générés à l'aide d'un CLV sont particulièrement sensibles à l'égard d'erreurs de transmission, principalement à cause de la perte de synchronisation qui peut résulter de certaines erreurs introduites par le canal.

Une solution classique consiste à faire appel à un codage canal puissant ou à insérer dans le train binaire des marques de synchronisation. Cependant, ceci conduit à une augmentation significative du débit binaire sur le canal. Des techniques plus récentes ont exploité la structure particulière des CLV afin de développer des algorithmes de décodage souple de trains binaires générés à l'aide d'un CLV [BF95], [Bal97], [DS98], [BH00], [SOD00]. L'ensemble des successions de mots de code possibles est décrite à l'aide de treillis exploitant certaines informations supposés disponibles *a priori* au niveau du décodeur (nombre de mots de code, longueur du train binaire à décoder, etc.). Les performances obtenues sont bien meilleures que celles atteintes par des décodeurs classiques. L'ajout d'information sur la source, par exemple d'un modèle Markovien [TK03], [JCS05] ou la prise en compte de la syntaxe que doit respecter le train binaire généré par le codeur source [ND03], [LKD05], améliorent encore les performances du décodage. Enfin, le décodeur peut mettre en œuvre un estimateur au sens du maximum de vraisemblance (MV) ou du sens du maximum *a posteriori* (MAP) du train binaire émis. Dans le second cas, les probabilités *a priori* des mots de code doivent être disponibles au décodeur ; elles peuvent également être estimées comme le suggère [MSJ06].

Toutes ces méthodes ont en commun la construction et l'élagage de treillis intégrant plus ou moins d'information *a priori*. La complexité de ces treillis peut devenir rédhibitoire lorsque des CLV comportant un nombre important de mots de code sont considérés. Une technique d'agrégation d'état a été proposée récemment par [JMG05]. Dans cette approche, le treillis de décodage est périodiquement *replié*, ce qui permet d'obtenir un treillis de complexité réduite. Les résultats que nous proposons ont le même objectif de réduction de complexité du décodage, mais l'approche est différente et complémentaire des travaux présentés dans [JMG05]. En effet, cet article présente un algorithme de construction de tables simplifiées pour le décodage de données générées par un CLV. Les mots de code sont regroupés en un nombre minimal de classes. Au lieu de travailler sur l'ensemble des mots de code, les algorithmes de décodage souple peuvent utiliser ces classes en nombre réduit, ce qui diminue la complexité des treillis et les temps de calcul.

Le paragraphe 2, présente brièvement un exemple de treillis utilisé par un algorithme de décodage au sens du MAP de données produites à l'aide d'un CLV afin de mettre l'accent sur la complexité du treillis à considérer. Le paragraphe 3 décrit le principe de l'algorithme de simplification des mots de code utilisés par un CLV donné, les preuves d'optimalité étant données dans [MKKD06]. L'adaptation des algorithmes de décodage tels que SOVA et BCJR à des treillis exploitant ces CLV groupés en classes est proposée au paragraphe 4. Un exemple de simplification des tables utilisées pour le codage de la texture par H.263+ est présenté au paragraphe 5. Ce paragraphe se termine par l'étude des performances d'un algorithme de décodage à entrées souples utilisant une table normale puis une table compactée.

2. Décodage souple de CLV

On considère une source X générant des symboles appartenant à un alphabet $\mathcal{X} = \{X_1, \ldots, X_K\}$. À chaque symbole X_k est associée une probabilité d'occurrence p_k. Un CLV associe à chaque symbole un mot de code $\mathbf{x}_k$ de $\ell_k = \ell(\mathbf{x}_k)$ bits, tel que $\ell_{\min} \leqslant \ell_k \leqslant \ell_{\max}$ pour tout $k \in \{1, \ldots, K\}$. À une suite de N symboles de la source peut donc être associé un vecteur de N mots de code appartenant au CLV

$$\mathbf{x}_1^N = (\mathbf{x}(1), \ldots, \mathbf{x}(N)), \text{ avec } \sum_{n=1}^{N} \ell(\mathbf{x}(n)) = L.$$

Cette suite peut aussi être vue comme un vecteur de L bits $\mathbf{b}_1^L = (b_1, \ldots, b_L)$. Lorsque $\mathbf{b}_1^L$ est envoyé sur un canal de transmission, le vecteur $\mathbf{y}_1^L = (y_1, \ldots, y_L)$ est obtenu au récepteur.

Au niveau du décodeur, lorsque seuls L et la table des mots de code du CLV sont connus, l'estimée $\hat{\mathbf{x}}$ au sens du MAP de $\mathbf{x}_1^N$ est

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x} \in \mathcal{S}_L} p\left(\mathbf{x} | \mathbf{y}_1^L\right), \tag{1}$$

où $\mathcal{S}_L$ est l'ensemble de toutes les suites de mots de code de L bits pouvant être générées par le CLV considéré. Lorsque le canal est gaussien sans mémoire et que les probabilités *a priori* des mots de code sont toutes identiques, (1) se traduit par la minimisation sur $\mathcal{S}_L$ d'une norme euclidienne.

Dans le cas général, (1) nécessite la résolution d'un problème de minimisation sous contraintes. Pour cela, [KB00] propose la construction d'un treillis représentant l'ensemble des suites de mots de code de longueur totale L pouvant être générées par un CLV. Une fois le treillis disponible, des algorithmes tels que SOVA [HH89] ou BCJR [BCJR74] peuvent être employés. Dans le treillis proposé, chaque nœud (a) représente la fin d'une suite de mots de code dont la longueur cumulée est a bits. Chaque branche correspond à un mot de code. Ainsi, un groupe de branches parallèles reliant les nœuds (a) et $(a + \ell)$ représente tous les mots de code de ℓ bits. La figure 1 montre

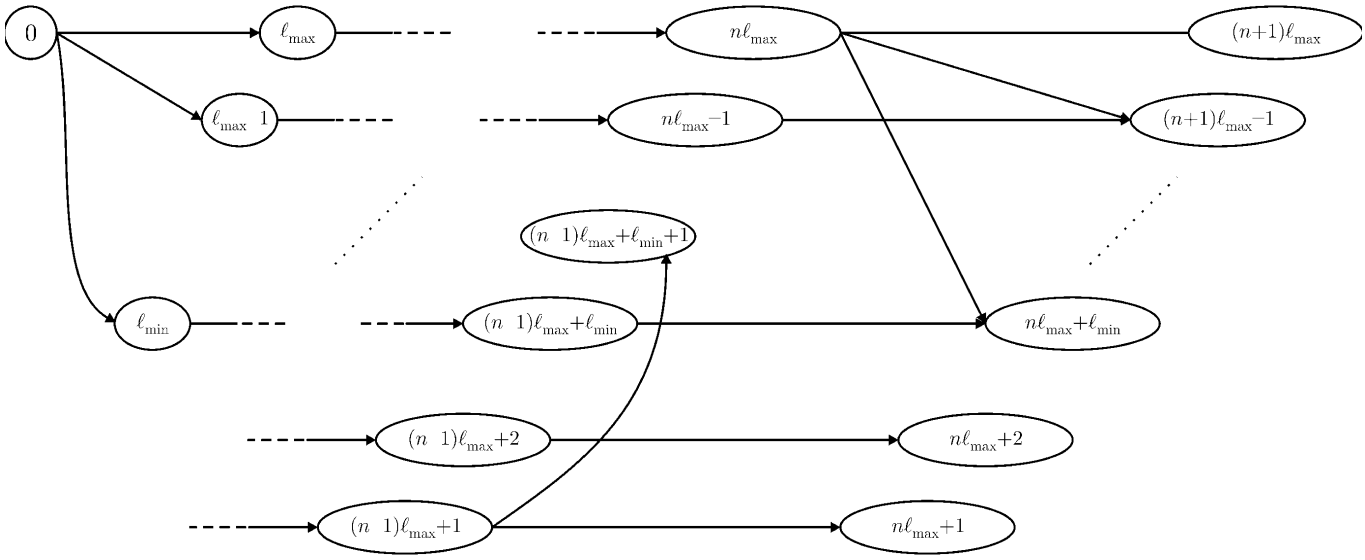

Figure 1. Structure de treillis pour des CLV de longueur comprise entre $\ell_{\min}$ et $\ell_{\max}$.

qu'après un certain temps, la structure devient périodique. Chaque période du treillis contient au plus $\ell_{\max}$ nœuds. Le nombre total de nœuds dépend de $\ell_{\min}$ et est au maximum de L. Ainsi, le nombre de branches arrivant et émergeant d'un nœud donné peut être très grand. Si l'on considère par exemple le CLV utilisé pour la texture dans la norme H.263+, pour les mots de code de 10 bits, 42 branches parallèles relient $(a-10)$ et (a). Pour le décodage, le nombre de métriques de branche à évaluer est donc très important.

3. Simplification de tables de CLV

Une méthode permettant de réduire la complexité de décodage consiste à utiliser une table de mots de code de taille réduite. Cet article présente un algorithme de regroupement des mots de code de même longueur en classes. Ainsi, par exemple, les 16 mots de code de 8 bits utilisés pour le codage de la texture dans la norme H.263+

$$\mathcal{A}_8 = \{00100000, 00100001 \ldots 00101110, 00101111\}$$

sont formés du préfixe 0010 et de tous les suffixes possibles de 4 bits. $\mathcal{A}_8$ peut ainsi être décrit de manière compacte par 0010\$\$\$\$, où \$ représente soit 0, soit 1. À l'aide de cette simplification, dans le treillis présenté au paragraphe 2, 16 branches parallèles entre les nœuds (a) et $(a+8)$ peuvent être remplacées par une branche unique. Le principe de l'algorithme proposé généralise la technique de réduction présentée précédemment.

3.1. Notations et définitions

On considère l'ensemble des K_ℓ mots de code de ℓ bits, $\mathcal{A}_\ell = \{\mathbf{a}_1, \ldots, \mathbf{a}_{K_\ell}\}$, utilisé par un CLV donné. Le cardinal de $\mathcal{A}_\ell$ est noté $|\mathcal{A}_\ell|$. Il s'agit de trouver un regroupement des éléments de $\mathcal{A}_\ell$ de manière à obtenir un nombre minimal de représentants pour ces mots de code.

Soit $\mathbb{B} = \{0,1\}$, l'ensemble des symboles binaires, qui peut être étendu en $\overline{\mathbb{B}} = \{0,1,\$\}$ avec l'élément *indéterminé* \$, correspondant à $\{0,1\}$. $\mathbb{B}^\ell$ et $\overline{\mathbb{B}}^\ell$ dénotent respectivement l'ensemble des vecteurs de dimension ℓ des mots binaires et binaires étendus.

Définition 1 : *Une* classe $\mathbf{c}$ *est un élément de* $\overline{\mathbb{B}}^\ell$.

Soit $\mathcal{S}(\mathbf{c})$ l'ensemble des éléments de $\mathbb{B}^\ell$ correspondant à $\mathbf{c}$. Ainsi, si $\mathbf{c} = (c_1, \ldots, c_\ell) \in \overline{\mathbb{B}}^\ell$, alors

$$\mathcal{S}(\mathbf{c}) = \left\{(x_1, \ldots, x_\ell) \in \mathbb{B}^\ell \text{ tq } x_i = c_i \text{ ou } c_i = \$\right\}.$$

Définition 2 : *Une* classification $\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_M\}$ *de* $\mathcal{A}_\ell$ *est un ensemble de classes tel que*

$$\begin{cases} \forall \mathbf{x} \in \mathcal{A}_\ell, \exists i \text{ tel que } \mathbf{x} \in \mathcal{S}(\mathbf{c}_i), \\ \mathcal{S}(\mathbf{c}_i) \subset \mathcal{A}_\ell,\ i = 1, \ldots, M. \end{cases}$$

La méthode de simplification proposée consiste à représenter $\mathcal{A}_\ell$ à l'aide d'une *classification minimale*, c'est-à-dire une classification comportant un nombre minimal de classes.

Avant de présenter l'algorithme de construction de la classification minimale, quelques notions complémentaires doivent être introduites. La distance de Hamming $d(\cdot,\cdot)$ se généralise de $\mathbb{B}$ à $\overline{\mathbb{B}}$. Ainsi, si $(x,y) \in \overline{\mathbb{B}}$, $d(x,y) = 1$ si $x \neq y$ et $d(x,y) = 0$ sinon. La distance de Hamming entre deux classes est la somme des distances entre les composantes de ces classes. L'*ordre o* $(\mathbf{c})$ d'une classe $\mathbf{c}$ correspond au nombre d'éléments indéterminés

de $\mathbf{c}$. Il est tel que $o(\mathbf{c}) = \log_2(|\mathcal{S}(\mathbf{c})|)$, où $|\mathcal{S}(\mathbf{c})|$ est le cardinal de $\mathcal{S}(\mathbf{c})$. Deux classes de même ordre sont *adjacentes* lorsque leur distance de Hamming est de un.
Deux classes adjacentes $\mathbf{c}_1$ et $\mathbf{c}_2$ peuvent être *réunies* pour former une nouvelle classe $\mathbf{c} = \mathbf{c}_1 \sqcup \mathbf{c}_2$, on a alors $\mathcal{S}(\mathbf{c}) = \mathcal{S}(\mathbf{c}_1) \cup \mathcal{S}(\mathbf{c}_2)$ et $o(\mathbf{c}) = o(\mathbf{c}_1) + 1$.

3.2. Construction d'une classification minimale

Dans cet article, seul le principe de l'algorithme de construction d'une classification minimale est présenté. Les détails de l'algorithme ainsi qu'une preuve d'optimalité sont donnés dans [MKKD06].
L'algorithme a pour entrée un ensemble $\mathcal{A}_\ell$ de mots de code de longueur ℓ. Chaque élément de $\mathcal{A}_\ell$ est également une classe d'ordre 0. Une première classification $\mathcal{C}_0$ de $\mathcal{A}_\ell$ est formée à l'aide des éléments de $\mathcal{A}_\ell$. Un ensemble de classes $\mathcal{C}_1$ est obtenu avec toutes les classes adjacentes appartenant à $\mathcal{C}_0$ qu'il est possible de réunir. $\mathcal{C}_1$ n'est plus nécessairement une classification de $\mathcal{A}_\ell$ (par exemple, s'il existe un élément $\mathbf{c} \in \mathcal{C}_0$ n'ayant pas de classe adjacente, alors $\mathbf{c} \notin \mathcal{C}_1$). La procédure est itérée jusqu'à l'obtention d'une classe $\mathcal{C}_{j_{\max}}$ ne comportant plus aucune classe adjacente.
L'union de tous les ensembles ainsi obtenus $\overline{\mathcal{C}} = \bigcup_{j=0}^{j_{\max}} \mathcal{C}_j$ est une classification *maximale* de $\mathcal{A}_\ell$, c'est-à-dire qu'elle contient toutes les classes qu'il est possible de former par réunion de classes adjacentes obtenues à partir des éléments de $\mathcal{A}_\ell$. Une classification minimale $\underline{\mathcal{C}}$ de $\mathcal{A}_\ell$ est alors un sous-ensemble de $\overline{\mathcal{C}}$ qui reste une classification de $\mathcal{A}_\ell$ mais contenant le nombre minimum de classes. L'obtention de $\underline{\mathcal{C}}$ se fait en deux étapes. La première consiste à éliminer de $\overline{\mathcal{C}}$ tous les éléments des ensembles intermédiaires $\mathcal{C}_j$ qui ont pu être regroupés en classes appartenant à $\mathcal{C}_{j+1}$ (cette procédure peut être effectuée lors de la construction de $\mathcal{C}_{j+1}$). La classification $\mathcal{C}$ est alors obtenue. Pour former la classification minimale $\underline{\mathcal{C}}$, une méthode combinatoire consistant à éliminer un maximum de classes de $\mathcal{C}$, tout en vérifiant que le sous-ensemble obtenu reste une classification de $\mathcal{A}_\ell$, est ensuite appliquée.
L'annexe fournit un exemple de construction d'une classification minimale.
Cette technique peut fournir une classification minimale avec certains mots de code appartenant à plusieurs classes. Cette situation risque de soulever des difficultés lors du décodage à l'aide d'algorithmes de type MAP, la somme des probabilités *a priori* des classes n'étant plus égale à 1. Une solution à ce problème nécessite une adaptation de l'algorithme précédent présentée dans [MKKD06].

4. Propriétés

Les tables de CLV obtenues par la procédure de simplification décrite au paragraphe 3 contiennent légèrement moins d'information que les tables initiales. Ceci est dû au regroupement des mots de code en classes. Cependant, l'impact de cette perte d'information peut être très limité, selon l'algorithme de décodage utilisé.
Si un décodage au sens du MV est utilisé, l'algorithme utilisant les tables simplifiées fournira les mêmes résultats que s'il avait utilisé les tables complètes, en effet, les probabilités *a priori* des symboles ne sont pas utilisées.
Lorsque l'algorithme de décodage cherche à fournir une estimée au sens du MV et fournit des sorties souples (en utilisant par exemple SOVA [HH89], [VY00]), on montre au paragraphe 4.1. qu'à nouveau les résultats sont identiques à ceux obtenus avec une table complète. Rappelons que ces sorties souples permettent de réaliser un décodage itératif avec un décodeur canal, lui aussi à sorties souples, voir [BH00].
Pour un décodeur optimal utilisant le critère du MAP, tel que celui présenté dans [BH00], le résultat restera optimal si les probabilités *a priori* de chacun des mots de code regroupés au sein d'une même classe sont les mêmes. Comme les mots de code qui peuvent être regroupés doivent avoir la même longueur, il est raisonnable de penser que cette condition sera vérifiée. Cependant, dans le cas d'une utilisation des tables simplifiées dans le cadre d'une procédure itérative, les différences entre les probabilités *a priori* des mots de code formant une classe pourront être à l'origine d'une sous-optimalité, voir le paragraphe 4.2.
Nous avons adapté les deux algorithmes de décodage SOVA et BCJR afin d'utiliser les tables de CLV simplifiées sur un treillis tel que celui représenté sur la figure 1. La suite de ce paragraphe présente plus spécifiquement les adaptations nécessaires de ces deux algorithmes de décodage.

4.1. Décodage au sens du MV à sorties souples

Considérons un décodeur de type SOVA permettant de réaliser un décodage au sens du MV. Il est appliqué sur le treillis de la figure 1 en utilisant une table de CLV simplifiée. La séquence reçue a une longueur connue de L bits. À un instant bit donné t, $(1 \leqslant t \leqslant L)$, le rapport de vraisemblance associé au bit émis b_t se calcule de la manière suivante.
Un décodeur MV à sorties dures est d'abord employé pour évaluer les métriques des états dans le sens avant $\mu^f(t-i)$, $i = 1,\ldots,\min\{\ell_{\max}, t\}$, avec

$$\mu^f(0) = 0,\ \mu^f(1) = \infty,\ldots,\mu^f(\ell_{\min} - 1) = \infty$$

Un décodeur MV à sorties dures évalue ensuite les métriques des états dans le sens arrière $\mu^b(t+i)$, $i = 0,\ldots,\min\{\ell_{\max} - 1, L - t\}$, avec

$\mu^b(L)=0,\ \mu^b(L-1)=\infty,\ldots,\mu^b(L-\ell_{\min}+1)=\infty$

Les deux décodeurs bénéficient du treillis à complexité réduite donné par la table CLV simplifiée, en effet, le nombre de métriques de branches à évaluer et à comparer est bien plus faible avec une table simplifiée. De plus, les décisions dures effectuées sur chaque bit fournissent les mêmes résultats que ceux qu'obtiendrait un algorithme utilisant la table initiale. Le fait qu'une classification minimale produit des classes pouvant intersecter conduit à plusieurs trajets possibles pour la même séquence décodée. Cela n'a pas d'importance ici, car une décision dure est prise sur les bits reçus.
Supposons, sans perdre la généralité, que la métrique optimale $\mu^{opt}(L)$ correspond à une estimation dure de $\widehat{b}_t=1$. La sortie souple de SOVA associé à $\widehat{b}_t$ est

$$\Lambda(b_t)=\mu_t^0-\mu_t^1,$$

où $\mu_t^1=\mu^{opt}(L)$ et où μ_t^0 est la métrique minimale parmi les métriques de tous les trajets tels que $b_t=0$. Cette dernière métrique est évaluée comme suit [VY00]

$$\mu_t^0=\min_{i,\ell}\left(\mu^f(t-i)+\nu_\ell^0(t-i,t-i+\ell)+\mu^b(t-i+\ell)\right),$$

$$i=1,\ldots,\min\{\ell_{\max},t\}\text{ et}$$

$$\ell=\max\{\ell_{\min},i\},\ldots,\min\{\ell_{\max},L-t+i\}$$

où $\nu_\ell^0(t-i,t-i+\ell)$ est la métrique de la branche connectant l'état $(t-i)$ à l'état $(t-i+\ell)$ évaluée pour une classe ayant un bit 0 ou \$ à la i-ème position.
Si l'on considére d'une part une modulation de type *binary phase shift keying* (BPSK) utilisée sur un canal de transmission sans mémoire, perturbé par un bruit blanc additif gaussien (BBAG) de variance σ^2 et d'autre part une classification $\mathcal{C}_\ell=\left\{\mathbf{c}_{\ell,1},\ldots,\mathbf{c}_{\ell,M_\ell}\right\}$ des K_ℓ mots de ℓ bits du CLV, la métrique de branche s'écrit alors

$$\nu_j^0(t-i,t-i+j)=$$
$$\max_{\mathbf{c}\in\mathcal{C}_\ell\ \text{tq}\ c_i=0\ \text{ou}\ c_i=\$}\bar{\nu}_j^0\left(t-i,t-i+j,\widehat{\mathbf{c}}^{0,i}\left(\mathbf{y}_{t-i}^{t-i+j},\mathbf{c}\right)\right),\qquad(2)$$

où $\widehat{\mathbf{c}}^{0,i}=\widehat{\mathbf{c}}^{0,i}\left(\mathbf{y}_{t-i}^{t-i+j},\mathbf{c}\right)$ est le vecteur dont la n-ième composante est définie par

$$\widehat{c}_n^{0,i}=\begin{cases}c_n & \text{si } c_n\neq\$ \text{ et } n\neq i,\\ 0 & \text{si } n=i,\\ 1 & \text{si } c_n=\$ \text{ et } y_{t-i+n}>0,\\ 0 & \text{si } c_n=\$ \text{ et } y_{t-i+n}\leqslant 0.\end{cases}$$

Le vecteur $\widehat{\mathbf{c}}^{0,i}$ correspond à une décision dure faite à partir de $\mathbf{y}_{t-i}^{t-i+j}$, sachant que le i-ème bit de cette décision est 0 et que le résultat doit appartenir à $\mathcal{S}(\mathbf{c})$. De plus, dans (2)

$$\bar{\nu}_j^0\left(t-i,t-i+j,\widehat{\mathbf{c}}^{0,i}\right)=$$
$$\prod_{n=-i+1}^{\ell-i}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\left(y_{t+n}-m\left(\widehat{c}_n^{0,i}\right)\right)^2}{2\sigma^2}\right),$$

avec $m(0)=-1$ et $m(1)=1$. Cette seconde étape bénéficie aussi de la réduction de la table de CLV, et donne (à cause de la décision dure utilisée pour calculer $\widehat{\mathbf{c}}^{0,i}$ et $\bar{\nu}_j^0$) le même résultat qu'un algorithme exploitant la table de CLV initiale, avec une complexité moindre.
Ainsi, l'utilisation d'une table CLV combinée avec un SOVA effectuant une estimation du type MV aboutit à une réduction significative du nombre de branches sur le treillis associé à la table CLV mais avec des résultats similaires à ceux obtenus avec la table de CLV initiale.

4.2. Décodage au sens du MAP à sorties souples

Considérons à nouveau le treillis représenté dans la figure 1. La longueur L de la séquence reçue $\mathbf{y}_1^L$ est toujours supposée connue. Cette fois, un algorithme de type BCJR [BCJR74] est considéré. L'utilisation d'une table simplifiée nécessite une modification des métriques α, β et γ.
Les métriques avant $\alpha(t)$ et arrière $\beta(t)$ sont données par

$$\alpha(t)=\sum_{i=\ell_{\min}}^{\min\{\ell_{\max},t\}}\alpha(t-i)\,\gamma(t-i,t)\qquad\text{et}$$

$$\beta(t)=\sum_{i=\ell_{\min}}^{\min\{\ell_{\max},L-t\}}\beta(t+i)\,\gamma(t,t+i)\qquad(3)$$

avec $\alpha(0)=1$, $\alpha(t)=0$, $t=1,\ldots,\ell_{\min}$ et
$\beta(L)=1$, $\beta(L-t)=0$, $t=1,\ldots,\ell_{\min}-1$.
Le calcul de la métrique associée aux branches liant les états (t) et $(t+\ell)$ utilise une classification $\mathcal{C}_\ell=\left\{\mathbf{c}_{\ell,1},\ldots,\mathbf{c}_{\ell,M_\ell}\right\}$ des K_ℓ mots de ℓ bits du CLV. La métrique de branche s'écrit

$$\gamma(t,t+\ell)=\sum_{\mathbf{c}\in\mathcal{C}_\ell}p(\mathbf{c})\,\overline{\gamma}(t,t+\ell,\mathbf{c}).\qquad(4)$$

Elle correspond à la moyenne pondérée des métriques associées à chaque classe, où $p(\mathbf{c})$ est la somme des probabilités *a priori* des éléments de $\mathcal{S}(\mathbf{c})$. Dans (4)

$$\overline{\gamma}(t,t+\ell,\mathbf{c})=\prod_{j=1}^{\ell}d\left(y_{t+j},c_j\right),\qquad(5)$$

où c_j est la j-ème composante de $\mathbf{c}$. Pour les mêmes conditions de transmission qu'au paragraphe 4.1., $d(y,c)$ peut facilement se calculer pour $t=1,\ldots,L$

$$d(y,c) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-m(c))^2}{2\sigma^2}\right) & \text{si } c \in \{0,1\}, \\ d(y,0) + d(y,1) = & \\ \frac{2}{\sqrt{2\pi\sigma^2}} \cosh\left(\frac{y}{\sigma^2}\right) \exp\left(-\frac{(y^2+1)}{2\sigma^2}\right) & \text{si } c = \$, \end{cases} \tag{6}$$

avec $m(0) = -1$ et $m(1) = 1$. Une fois les métriques évaluées, le rapport des log-vraisemblances pour le bit émis b_t à l'instant t est donné par

$$\Lambda(b_t) = \log \frac{\Pr(b_t=1|\mathbf{y})}{\Pr(b_t=0|\mathbf{y})} = \log \frac{\sum_{i=1}^{\min\{\ell_{\max},t\}} \sum_{j=\max\{\ell_{\min},i\}}^{\min\{\ell_{\max},L-t+i\}} \alpha(t-i)\gamma^1(t-i,t-i+j,t)\beta(t-i+j)}{\sum_{i=1}^{\min\{\ell_{\max},t\}} \sum_{j=\max\{\ell_{\min},i\}}^{\min\{\ell_{\max},L-t+i\}} \alpha(t-i)\gamma^0(t-i,t-i+j,t)\beta(t-i+j)}. \tag{7}$$

où $\gamma^1(t-i,t-i+j,t)$ et $\gamma^0(t-i,t-i+j,t)$ sont les métriques des branches liant les états $(t-i)$ et $(t-i+j)$ qui correspondent respectivement aux classes pour lesquelles $b_t = 1$ et $b_t = 0$. Ces métriques peuvent facilement être déduites à partir de (4).

Un algorithme de décodage utilisant l'ensemble $\mathcal{A}_\ell = \{\mathbf{a}_1, \ldots, \mathbf{a}_{K_\ell}\}$ des mots de code de ℓ bits de la table de CLV initiale, aurait évalué

$$\gamma(t,t+\ell) = \sum_{\mathbf{a}\in\mathcal{A}_\ell} p(\mathbf{a})\, \overline{\gamma}(t,t+\ell,\mathbf{a}), \tag{8}$$

où $p(\mathbf{a})$ est la probabilité *a priori* de $\mathbf{a}$. Par conséquent, un décodeur MAP appliqué à une table de CLV simplifiée reste optimal seulement lorsque les mots de code fusionnés dans une classe donnée ont la même probabilité *a priori* et quand il n'y a aucun mot de code appartenant simultanément à plusieurs classes. Dans ce cas, la classification minimale sans intersection doit être utilisée. En outre, parmi les mots de code de même longueur seuls ceux ayant des probabilités *a priori* très proches peuvent être regroupés.

4.3. Brève étude de complexité

L'essentiel du gain en terme de réduction de complexité est obtenu lors de l'évaluation des métriques de branches. Avec une table de CLV initiale, le calcul de $\gamma(t,t+\ell)$ requiert l'évaluation de $\gamma(t,t+\ell,\mathbf{a})$ pour K_ℓ mots de CLV. D'après (5), l'évaluation de $\overline{\gamma}(t,t+\ell,\mathbf{a})$ nécessite ℓ comparaisons pour sélectionner la bonne métrique bit et $\ell - 1$ multiplications. Puis, d'après (8), le calcul de $\gamma(t,t+\ell)$ nécessite K_ℓ multiplications (multiplication de $\overline{\gamma}(t,t+\ell,\mathbf{a})$ par $p(\mathbf{a})$) et $K_\ell - 1$ additions. En résumé, l'évaluation complète de $\gamma(t,t+\ell)$ requiert $K_\ell + K_\ell(\ell-1) = K_\ell\ell$ multiplications, $K_\ell - 1$ additions et $K_\ell\ell$ comparaisons.

Avec une table de CLV simplifiée, l'évaluation de $\overline{\gamma}(t,t+\ell,\mathbf{c})$ requiert moins de 2ℓ comparaisons pour choisir la bonne métrique bit et $\ell - 1$ multiplications. Ces chiffres doivent être multipliés par M_ℓ pour obtenir le coût de calcul pour toutes les classes. L'évaluation de $\gamma(t,t+\ell)$ à partir de $\overline{\gamma}(t,t+\ell,\mathbf{c})$ et de $p(\mathbf{c})$ à l'aide de (4) requiert alors M_ℓ multiplications et $M_\ell - 1$ additions. En résumé, l'évaluation complète de $\gamma(t,t+\ell)$ requiert $M_\ell\ell$ multiplications, $M_\ell - 1$ additions et au plus $2M_\ell\ell$ comparaisons.

Pour M_ℓ très inférieur à K_ℓ, un gain en terme de complexité de calcul est donc obtenu avec une table simplifiée.

5. Application

Dans [LKD05], un algorithme de décodage des informations de texture générées par un codeur H.263+ est considéré. Chaque bloc de texture de 8×8 pixels subit une transformation en cosinus discrète, une quantification, un balayage zig-zag, un codage par plages de zéros puis un codage entropique à l'aide d'un CLV. Les mots de code ainsi générés sont placés dans des paquets qui sont ensuite envoyés sur le canal. Un paquet contient en général plusieurs blocs de texture. Le décodeur proposé par [LKD05] comprend deux étapes. La première consiste à localiser les blocs de texture dans un paquet. La seconde consiste à décoder chaque bloc ainsi localisé en exploitant certaines contraintes liées à la syntaxe du train binaire que peut générer un codeur H.263+. Par exemple, après décodage, le nombre de coefficients transformés d'un bloc doit être de 64, voir [ND03] pour plus de détails.

Pour la première étape, seul le nombre de blocs que comporte un paquet est connu *a priori*. Pour localiser la position des blocs, le treillis présenté au paragraphe 2 doit être modifié pour faire apparaître comme propriété de chaque nœud le nombre de blocs accumulés. Ceci revient à ajouter une dimension au treillis. Pour chaque mot de code, il convient également de distinguer ceux qui marquent la fin d'un bloc.

Ainsi, les 204 mots du CLV dédiés au codage de la texture dans le standard H.263+ sont partitionnés en deux sous-ensembles $\mathcal{A}$ et $\mathcal{E}$ contenant respectivement les mots de code ne marquant pas et marquant la fin d'un bloc. La table 1 rassemble les résultats obtenus par la procédure de simplification des tables de CLV appliquée aux mots de code de même longueur de $\mathcal{A}$ et de $\mathcal{E}$. Les 204 mots de code peuvent être regroupés en 34 classes.

Pour tester l'algorithme de localisation de frontières de blocs de texture exploitant les tables de CLV simplifiées, la séquence vidéo `foreman` au format QCIF a été codée en H.263+ à 256 kb/s et 30 images/s. Une image codée en INTRA est placée toutes les 10 images. Les paquets de texture ont une longueur maximale de 1 024 bits. Chaque paquet contient environ 10 blocs de texture lorsqu'ils sont codés en INTRA et environs 30 blocs, s'ils sont codés en INTER. Ces paquets sont ensuite envoyés sur un canal BBAG.

Un algorithme de Viterbi est employé pour la localisation des frontières de blocs dans des paquets de texture. Les performances sont données en terme de taux d'erreur bloc (TEB), correspondant à la proportion de blocs de texture de 8×8 pixels

Tableau 1. Extrait de la classification minimale des mots de codes utilisés pour le codage de la texture dans la norme H.263+. Les mots marquant la fin d'un bloc (EOB) sont distingués des autres.

Longueur	EOB	Nb de mots de code	Classification minimale	Nb de classes
3	0	2	10$	1
4	0	2	110$	1
5	0	4	111$$	1
	1	2	0111$	1
⋮	⋮	⋮	⋮	⋮
10	0	24	00010010$$, 0001000$$$, 0000111$$$, 00001101$$	4
	1	18	00001100$$, 0000101$$$, 00001001$$, 000010001$	4
11	0	20	000010000$$, 0000001$$$$	2
	1	8	00000001$$$	1
12	0	12	0000000011$$, 000001000$$$	2
	1	12	0000000010$$, 000001001$$$	2
13	0	16	000001010$$$$	1
	1	16	000001011$$$$	1
Total		204		34

erronés, pour des images codées INTER et INTRA sur la figure 2 et en terme de temps de décodage moyen par bloc dans le tableau 2.

Tableau 2. Temps de décodage moyen (en ms/bloc).

	ML, table CLV		MAP, table CLV	
	initiale	compacte	initiale	compacte
INTRA	165	62 (−62%)	242	92 (−62%)
INTER	356	108 (−70%)	445	131 (−71%)

Avec des temps de calcul presque trois fois moindres, les performances sont la plupart du temps très semblables, excepté pour le décodage au sens du MAP des blocs de type INTRA, où une perte de l'ordre de 0.2 dB peut être notée. La diminution de performances dans le cas des paquets de type INTRA est due au fait que les mots de code regroupés dans une même classe n'ont pas toujours des probabilités *a priori* identiques. Le regroupement se fait alors avec une perte d'information. Dans le cas de paquets de type INTER, les probabilités *a priori* des mots de code regroupés sont plus proches, ce qui explique d'une part des performances du décodeur MAP à peine meilleures que celles

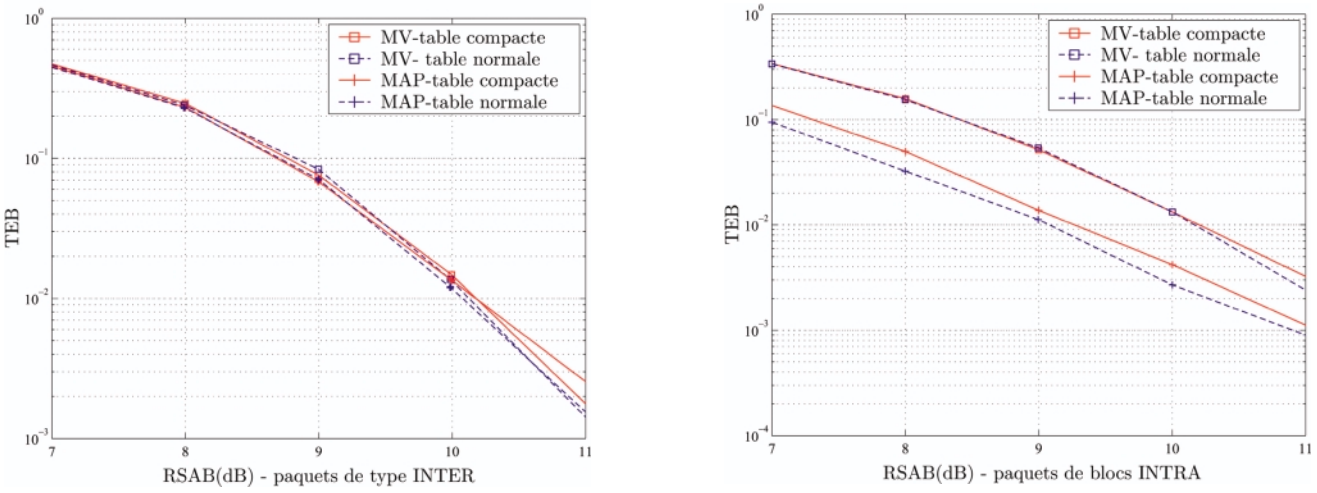

Figure 2. Décodage de paquets de blocs codés INTER et INTRA ; taux d'erreur blocs (TEB) en fonction du rapport signal-à-bruit (RSAB) sur le canal.

du ML et d'autre part, une dégradation moins importante lors de l'utilisation de la table CLV simplifiée.

6. Conclusion

Cet article décrit le principe d'un algorithme de simplification des tables de codes à longueur variable. Son principe consiste à regrouper des mots de code de même longueur en un nombre réduit de classes. Nous avons montré comment des algorithmes de type SOVA ou BCJR peuvent être adaptés à ces classes de manière à pouvoir réaliser un décodage robuste de données codées à l'aide de CLV. Nous avons également évalué la réduction de complexité apportée par ces regroupements en classes.
Dans l'exemple traité, la simplification a permis de représenter 204 mots de code par 34 classes, ce qui réduit largement la complexité des treillis représentant l'ensemble des succession de mots de code possibles pour une longueur de séquence codée. Les performances du décodeur au sens du MV restent inchangées lorsque des tables simplifiées lui sont fournies, mais le temps de calcul est presque trois fois moindre. Pour un décodeur utilisant un critère de MAP, une légère perte d'optimalité peut être notée lorsque les probabilités *a priori* des mots de code regroupés au sein d'une même classe sont différentes. Comme les mots de code qui peuvent être regroupés doivent avoir la même longueur, les différences restent faibles.
La combinaison des tables simplifiées que nous avons présentées avec les techniques d'agrégation d'état proposées par [JMG05] devraient permettre de réduire encore la complexité du décodage à sorties souples de données codées à l'aide de codes à longueur variable.

Annexe

Considérons l'ensemble $\mathcal{A}_7$ formé de 24 mots de code de 7 bits du CLV utilisé pour coder la texture dans la norme H.263+.

$$\begin{aligned}\mathcal{A}_7 = \{&\mathbf{000}0000,\mathbf{000}0001,\mathbf{000}0010,\mathbf{000}0011,\\ &\mathbf{000}0100,\mathbf{000}0101,\mathbf{000}0110,\mathbf{000}0111,\\ &\mathbf{000}1000,\mathbf{000}1001,\mathbf{000}1010,\mathbf{000}1011,\\ &\mathbf{0010}00,\mathbf{0010}01,\mathbf{0010}10,\mathbf{0010}11,\\ &\mathbf{0101}00,\mathbf{0101}01,\mathbf{0101}10,\mathbf{0101}11,\\ &\mathbf{1010}00,\mathbf{1010}01,\mathbf{1010}10,\mathbf{1010}11\}\,.\end{aligned}$$

Le calcul de la classification maximale et l'application de la première procédure d'élimination des classes inutiles permet d'obtenir la classification suivante de $\mathcal{A}_7$

$$\mathcal{C}'_7 = \{0000\$\$\$,000\$0\$\$,00\$01\$\$,0\$010\$\$,\$0101\$\$\}\,.$$

Nous constatons que $\mathcal{S}$ (\$0101\$\$) et $\mathcal{S}$ (0\$010\$\$) contiennent des mots de code n'appartenant à aucune autre classe de $\mathcal{C}'_7$. Une classification minimale contiendra donc nécessairement les classes correspondantes. Une méthode combinatoire sur les trois classes restantes permet ensuite de fabriquer une classification minimale

$$\underline{\mathcal{C}}_7 = \{0000\$\$\$,0\$010\$\$,\$0101\$\$\}\,.$$

On constate en effet qu'une quelconque des trois premières classes de $\mathcal{C}'_7$ peut être supprimée, l'ensemble de classes obtenues reste une classification de $\mathcal{A}_7$. Si l'on supprime les deux premières classes, 0000000 n'appartient plus à la classification obtenue ; si l'on supprime la première et la troisième, c'est 0000100 qui pose problème. Par contre, la suppression des seconde et troisième classes de $\mathcal{C}'_7$ permet encore de conserver une classification de $\mathcal{A}_7$, qui est la classification minimale.

Références

[Bal97] V. B. BALAKIRSKY. Joint source-channel coding with variable length codes. In *Proceedings of lEEE ISIT*, Ulm, Germany, 1997.

[BCJR74] L. R. BAHL, J. COCKE, F. JELINEK, and J. RAVIV. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. Info. Theory*, 20 :284–287, 1974.

[BF95] V. BUTTIGIEG and P.G. FARRELL. A MAP decoding algorithm for variable-length error-correcting codes. In *Codes and Cyphers : Cryptography and Coding IV*, pages 103–119, Essex, England, 1995. The Inst. of Mathematics and its Appl.

[BH00] R. BAUER and J. HAGENAUER. Turbo-FEC/VLC-decoding and its application to text compression. In *Proceedings of the Conference on Information Sciences and Systems (CISS'00),* Princeton University, USA, 2000.

[DS98] N. DEMIR and K. SAYOOD. Joint source/channel coding for variable length codes. In *IEEE Data Compression Conf.*, pages 139–148, Snowbird, UT, 1998.

[HH89] J. HAGENAUER and P. HOEHER. A Viterbi algorithm with soft-decision outputs and its applications. In *Proc. Globecom,* pages 1680–1686, Dallas, TX, 1989.

[JCS05] M. JEANNE, J.-C. CARLACH and P. SIOHAN. Joint source channel decoding of variable length codes for convolutional codes and turbo codes. *IEEE Trans. on Communications*, 53(1) :10–15, 2005.

[JMG05] H. JÉGOU, S. MALINOWSKI and C. GUILLEMOT. Trellis stage aggregation for soft decoding of variable-length codes. In *Proceedings of SPIS*, 2005.

[KB00] S. KAISER and M. BYSTROM. Soft decoding of variable-length codes. In *Proceedings of the IEEE International Conference on Communications*, volume 3, pages 1203–1207, New Orleans, 2000.

[LKD05] C.M. LEE, M. KIEFFER and P. DUHAMEL. Soft decoding of VLC encoded data for robust transmission of packetized video. In *Proceedings of ICASSP*, pages 737–740, 2005.

[MKKD06] G. R. MOHAMMAD-KHANI, M. KIEFFER and P. DUHAMEL. Simplification of VLC tables with application to ML and MAP decoding algorithms. *IEEE Transactions on Communications*, 54(10): 1835-1844, 2006.

[MSJ06] V. MANNONI, P. SIOHAN and M. JEANNE. A simple on-line turbo estimation of source statistics for joint sourcechannel VLC decoders : Application to MPEG-4 video. In *Proc., 4th International Symposium on Turbo Codes & Related Topics*, 2006.

[ND03] H. NGUYEN and P. DUHAMEL. Compressed image and video redundancy for joint source-channel decoding. In *Proc. Globecom,* 2003.

[SOD00] K. SAYOOD, H. H. OTU and N. DEMIR. Joint source/channel coding for variable length codes. *IEEE Trans. Commun*., 48 :787–794, 2000.

[TK03] R. THOBABEN and J. KLIEWER. On iterative source-channel decoding for variable-length encoded markov sources using a bit-level trellis. In *Proc. IV IEEE Signal Processing Workshop on Signal Processing Advances in Wireless Communications (SPAWC'03)*, Rome, 2003.

[VY00] B. VUCETIC and J. YUAN. *Turbo Codes – Principles and Applications*. Kluwer, Dordrecht, 2000.

Gholam-Reza **Mohammad-Khani**

Gholam-Reza Mohammad-Khani a obtenu les diplômes de Bachelor puis de Master in Science en électronique et systèmes de communication de l'Université de Technologie Sharif, Téhéran, Iran, en 1993 et en 1997. Il a obtenu sa thèse de doctorat en communication à l'Université de Limoges en 2002.

De novembre 2002 à octobre 2003, il a été post-doctorant au Geste, à l'ENSIL de l'université de Limoges. Depuis novembre 2003, il bénéficie d'une bourse de post-doctorant du CNRS au Laboratoire des Signaux et Systèmes, Gif-sur-Yvette. Ses principaux thèmes de recherche couvrent de nombreux aspects du traitement du signal pour les communications numériques (en particulier pour les systèmes de communication mobiles, la communication à accès multiple et les systèmes MIMO), le codage source-canal conjoint, le codage avec information adjacente, les canaux de diffusion et la théorie de l'information.

Michel **Kieffer**

Michel Kieffer est né à Sarreguemines en 1972. Il est normalien, agrégé de physique appliquée. Il a obtenu son doctorat en sciences et son habilitation à diriger des recherches de l'Université Paris-Sud, 11 en 1999 et en 2005.

Actuellement, il est maître de conférences en traitement du signal à l'Université Paris-Sud, 11. Son principal centre d'intérêt est le codage source-canal conjoint pour la transmission robuste de contenus multimédia. Il s'intéresse également aux techniques d'estimation ensemblistes utilisant l'analyse par intervalles. Avec L. Jaulin, O. Didrit et E. Walter, il a écrit Applied Interval Analysis paru en 2001 chez Springer-Verlag.

Chang-Ming **Lee**

Chang-Ming Lee est né à Taichung, Taiwan, en 1972. Il a reçu le diplôme de Bachelor in Science de la National Tsing Hua University, Taiwan, en 1994, le diplôme de Master in Science en ingénierie de l'information de la National Cheng Kung University, Taiwan, en 1997, et le diplôme de doctorat en automatique et traitement du signal de l'université Paris-Sud, 11, Orsay en 2004.

Depuis 2005, il est au département d'ingénierie de la communication de la National Chung Chen University, Chia-Yi, Taiwan. Ses activités de recherche actuelles concernent le codage source-canal conjoint et les systèmes de transmission de signaux multimédia.

Pierre **Duhamel**

Pierre Duhamel est ingénieur de l'INSA de Rennes. Il a reçu les diplômes de docteur ingénieur en 1978 puis de docteur en sciences en 1986 à l'Université Paris-Sud, 11.

De 1975 à 1980, il a travaillé chez Thomson-CSF, Paris, où il a travaillé sur la théorie des circuits pour le traitement du signal. En 1980, il a rejoint le CNET d'Issy les Moulineaux, où ses activités de recherche ont débuté par la sythèse de filtres récursifs pour les CCD. Plus tard, il a travaillé sur des algorithmes rapides de calculs de transformée de Fourier, de convolutions et a appliqué ces techniques au filtrage adaptatif, à l'analyse spectrale et à la transformée en ondelettes. De 1993 à 2000, il a été professeur à l'ENST, Paris avec une activité de recherche orientée vers le traitement du signal pour les télécoms. Il a dirigé le département Traitement du Signal et des Images de 1997 à 2000. Actuellement, il est directeur de recherches au CNRS, au LSS à Gif-sur-Yvette, où il développe des activités en traitement du signal pour les télécoms (égalisation, décodage itératif, systèmes multiporteuses) et en traitement du signal et des images pour le multimédia (codage de source, codage source-canal conjoint, tatouage, traitement audio).