Etude d'algorithmes de quantification vectorielle arborescente pour la compression d'images fixes

Patrick Fiche, Vincent Ricordel, Claude Labit

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE

Etude d'algorithmes de quantification vectorielle arborescente pour la compression d'images fixes

PATRICK FICHE, VINCENT RICORDEL, CLAUDE LABIT

N° 2241

janvier 1994

PROGRAMME 4

Robotique,

image

et vision

ISSN 0249-6399

Etude d'algorithmes de quantification vectorielle arborescente pour la compression d'images fixes

Patrick Fiche, Vincent Ricordel*, Claude Labit**

Programme 4 — Robotique, image et vision
Projet TEMIS

Rapport de recherche n°2241 — janvier 1994 — 39 pages

Résumé : Notre objectif est de construire un quantificateur à structure arborescente descendante élaguée pour le codage d'images fixes.
La quantification vectorielle est réalisée sur une image représentée non pas dans le domaine des luminances mais dans celui des fréquences (Une transformée DCT ou MLT est donc appliquée à l'image originale au préalable). Le changement d'espace de représentation a pour but de supprimer certaines redondances d'information présentes dans le domaine des luminances et de concentrer l'information sur quelques coefficients. Dans une première partie, nous présentons un quantificateur vectoriel (QV) à recherche exhaustive (c.a.d aucune contrainte de nature géométrique ou topologique n'est appliquée sur le dictionnaire des vecteurs représentants). L'algorithme de construction des dictionnaires utilisé est l'algorithme classique LBG (extension de l'algorithme de Lloyd-Max au cas vectoriel). Ce quantificateur a notamment pour but de comparer la quantification par blocs de pixels appartenant à une même bande fréquentielle (intra-bande) à la quantification où les blocs considérés correspondent spatialement aux blocs de l'image d'origine (inter-bande). Un des autres objectifs de ce quantificateur est de comparer l'efficacité des transformations MLT et DCT dans le cadre de la quantification vectorielle. Une étude porte également sur l'utilité de l'introduction d'une pondération psychovisuelle pour la reconstruction.
La complexité des QV à recherche exhaustive est coûteuse en termes opératoires (calculs de distorsion), ceci conduit à concevoir des quantificateurs à recherche arborescente. Un quantificateur de ce type a donc été réalisé et ses performances, bien que légèrement inférieures à celles du QV à recherche exhaustive (distorsion supérieure pour un même débit), demeurent acceptables vis à vis du gain de complexité (temps de calcul) réalisé. Enfin nous présentons la réalisation d'une structure arborescente élaguée permettant de construire des arbres non équilibrés afin de diminuer la distorsion pour un même débit.

(Abstract: pto)

Cette étude a été menée dans le cadre d'une coopération entre les laboratoires I3S (Sophia-Antipolis), L2S (ESE/Gif sur Yvette) et l'IRISA. Cette action, financée par le CNET/GDR TdSI "Traitement du Signal et de l'Image", vise à comparer differentes méthodes de compression d'images utilisant la quantification vectorielle.

Ce rapport présente particulièrement le travail de stage de fin d'études de Patrick Fiche réalisé en 1993 dans le cadre de la formation INSA (Institut National des Sciences Appliquées) de Rennes.

*IRISA e-mail ricordel@irisa.fr
**IRISA e-mail labit@irisa.fr

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

Tree-Structured VQ schemes for still images compression

Abstract: The purpose of this study is to design a pruned tree-structured vector quantizer for still images compression.
In order to exploit the intra-image redundancies and to obtain a frequency-based representation of the image information, a Discrete Cosine Transform (DCT) or a Modulated Lapped Transform (MLT) is first performed ; then a VQ-based scheme is used to code the coefficients.
In a first step, two vector design strategies are comparatively explored : vectors are composed by coefficients intraband or interband. The generalized Lloyd algorithm coupled with the "splitting" technique are applied to obtain the codebooks of a full search vector quantizer.
Experimental results show best performances for MLT transform in intraband case. Furthermore, we have tested a reconstruction scheme coding including psychovisual weighting.
The operating complexity presents a limitation on the applicability of an exhaustive codebook search. Hence, tree-structured VQ schemes are implemented ; suboptimal results are obtained but the degradation is quite reasonable compared to the complexity of an ordinary full search VQ. Finally a pruned tree structure is tested ; it provides better performances for a given low bitrate.

Table des matières

1 Introduction

La quantification vectorielle, méthode de compression de données, a pris une place très importante dans le domaine de la communication, que ce soit dans un but de transmission ou d'archivage d'informations.
Cette méthode s'applique essentiellement dans les deux domaines que sont l'image et la parole, ces deux formes de signaux contenant des informations redondantes.
La quantification vectorielle est un processus d'approximation d'un signal d'amplitude continue par un signal d'amplitude discrète. Le but de la compression étant d'extraire une information maximale tout en ne créant qu'un minimum de distorsion par rapport au signal original, notre objectif sera de créer une distorsion minimale pour un taux de compression donné ou autrement dit, de maximiser le taux de compression pour une qualité désirée.

Notre étude porte uniquement sur la compression d'images fixes (c-a-d sans tenir compte des redondances temporelles apparaissant dans une séquence).
Un des résultats fondamentaux des travaux de Shannon concernant la relation débit/distorsion montre que l'on obtient de meilleures performances en codant des vecteurs plutôt que des scalaires même dans le cas où la source vectorielle a k composantes statistiquement indépendantes [1] [2]. Aussi, nous n'avons traité que le cas de la quantification vectorielle tout en nous ramenant dans certains cas à de la quantification scalaire.

2 Présentation générale

2.1 Définition de la quantification vectorielle

La Quantification Vectorielle (QV) dans son sens le plus général est l'approximation d'un signal d'amplitude continue par un signal d'amplitude discrète.
Dans le cadre de notre application, elle consiste à représenter tout vecteur x de dimension k par un vecteur y de même dimension appartenant à un ensemble fini appelé dictionnaire.
D'un point de vue mathématique, nous pouvons définir notre quantificateur vectoriel de cette manière :
Soit I l'ensemble des indices correspondant au dictionnaire Y et $y_{i,i\in I}$ l'ensemble des vecteurs de Y.

$$I = \{1, \ldots, N\}$$

Le processus de codage est défini par :

$$\begin{array}{rccc} C & : & \mathbb{R} & \rightarrow & I \\ & & x & \rightarrow & C(x) \end{array}$$

Le processus de décodage étant quant à lui défini par :

$$\begin{array}{rccc} D & : & I & \rightarrow & Y \\ & & i & \rightarrow & y_i \end{array}$$

Cette opération, en comprimant les données perd de l'information et le signal original ne pourra plus être restitué. La quantification vectorielle est donc une opération `irréversible`.

2.2 Description d'un quantificateur vectoriel

Un Quantificateur Vectoriel (QV) se décompose donc généralement en deux applications :

- Un codeur
- Un décodeur

Dans la suite du rapport, l'abréviation QV sera utilisée pour désigner un Quantificateur Vectoriel ou la Quantification Vectorielle (appliqués au codage et décodage d'images) selon le contexte dans un but évident d'alléger les notations.

2.2.1 Le codeur

Le rôle du codeur consiste, pour tout vecteur x du signal en entrée (bloc de l'image), à rechercher dans le dictionnaire Y le code-vecteur y le plus "proche" du vecteur source x. La notion de proximité a été modélisée dans notre mise en oeuvre par la distance euclidienne entre vecteurs.
C'est uniquement l'`adresse` du code-vecteur y ainsi sélectionné qui sera transmise, c'est donc à ce niveau que s'effectue la compression.
Dans le cadre de notre étude, le dictionnaire est tout simplement représenté par un tableau et l'adresse du code-vecteur correspond à son indice dans le tableau.

2.2.2 Le décodeur

Le décodeur dispose d'une réplique du dictionnaire et consulte celui-ci pour fournir le `code-vecteur` d'indice correspondant à l'adresse reçue. Le décodeur réalise donc l'opération de décompression.

2.2.3 Schéma général d'un QV

Un QV peut est représenté par la Figure 1.

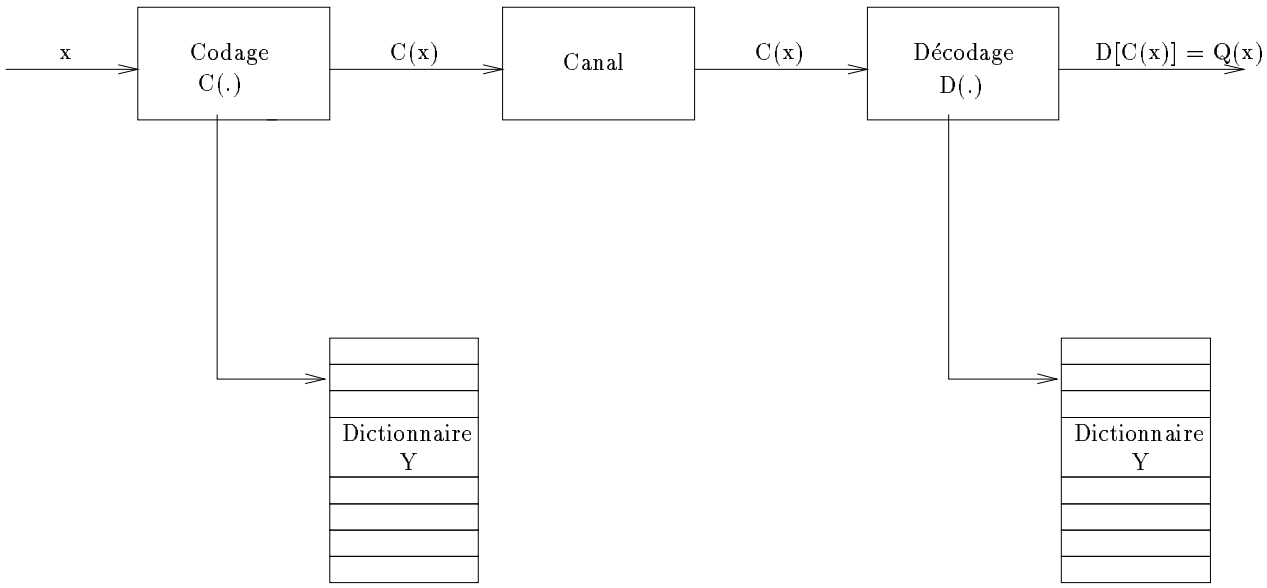

Figure 1 : schéma général d'un QV

2.2.4 Importance du dictionnaire

A travers ce schéma d'un QV, il est aisé de se rendre compte de l'importance primordiale du dictionnaire utilisé lors des opérations de codage et de décodage.

2.2.4.1 Caractéristiques d'un dictionnaire :

Un dictionnaire, indépendamment de son mode de construction et du système de recherche employé est défini par deux paramètres principaux :

- son nombre de vecteurs représentants : N.
- la dimension de l'espace vectoriel $\rightarrow$ dimension des vecteurs représentants : k.

Comme nous le verrons par la suite, nous travaillerons toujours dans un espace transformé et notre espace vectoriel sera un sous-ensemble de $\mathbb{R}^k$ avec $k \in \{1, 4, 16\}$. Les vecteurs de cet espace représenteront des blocs de l'image du domaine transformé.

Ces deux paramètres apparaissent étroitement liés et c'est en essayant d'ajuster au mieux la valeur de ces paramètres qu'il sera possible de construire un bon dictionnaire.
Avant de chercher quelle paraît être la meilleure composition de ces paramètres, il apparaît primordial de se fixer un ordre de grandeur quant à leur taille.

- La taille des vecteurs du dictionnaire a été fixée entre 1 et 16. En effet, en fixant une taille supérieure à 16, les effets de blocs deviennent trop importants et le dictionnaire est complètement conditionné par l'image à partir de laquelle il a été construit. Dans un cas extrême, on pourrait en arriver à prendre un vecteur de la taille de la séquence d'apprentissage avec donc un seul représentant, situation qui apparaît bien sûr non-envisageable.

- Le nombre de représentants du dictionnaire est quant à lui limité par deux contraintes bien distinctes :

  - Comme nous le verrons par la suite, le débit d'un QV est égal à $\frac{log_2 N}{k}$ bits/pixel où N est le nombre de représentants du dictionnaire et k la taille des vecteurs.
    Pour que le quantificateur ait un intérêt quelconque, il faut bien sûr que le débit soit inférieur au débit du signal original (en pratique, on se fixera un débit maximum bien inférieur au débit original).
    Aussi, si on fait de la quantification scalaire ($k = 1$), en appliquant la formule précédente et en prenant comme débit initial 8 bits/pixel, le nombre maximum de représentants est 256 ; par contre si on a utilisé des blocs de taille 16, le nombre maximum de représentants est $2^{16*8} = 2^{128}$.
    On voit déjà apparaître la nature du deuxième problème qui est l'espace occupé par le dictionnaire.

  - On essaiera donc, selon les applications, de limiter la taille des dictionnaires, il apparaît de plus inconcevable que les dictionnaires contiennent plus de blocs que l'image originale. Ainsi, pour une image $512*512$, le nombre de représentants du dictionnaire doit être majoré par 16384 dans le cas de blocs de taille 16.

Il apparaît que pour les parties de l'image exigeant une restitution du signal de haute qualité ou contenant une énergie importante (c-a-d nécessitant un débit important), on sera obligé d'utiliser de la quantification scalaire ou du moins des blocs de petite taille afin de ne pas construire un dictionnaire de taille gigantesque. Par contre, pour les parties de l'image n'exigeant pas une grande qualité de restitution ou recelant une faible énergie, il sera possible d'utiliser des blocs de plus grande taille sans connaître une explosion du nombre de représentants et en utilisant le fait que pour un débit donné, la quantification vectorielle est supérieure à la quantification scalaire.
Cette remarque nous montre immédiatement l'intérêt d'un système à débit variable et par la même occasion, l'intérêt de regrouper les informations importantes ou recelant un haut niveau d'énergie. Ceci justifiera la notion de changement d'espace de représentation introduite ultérieurement.

2.2.4.2 Universalité d'un dictionnaire :

La structure du dictionnaire à construire va aussi dépendre du cadre d'utilisation du QV :

- Si le QV est utilisé uniquement pour faire de la transmission de données, la taille du dictionnaire utilisé peut être très importante sans que ceci ne soit un réel handicap, le facteur à prendre en compte étant alors uniquement le débit engendré par le codage grâce à ce dictionnaire. L'intérêt d'un tel dictionnaire apparaît limité si celui-ci permet de restituer avec une bonne qualité uniquement la séquence d'apprentissage à partir de laquelle il a été élaboré mais ne peut restituer de nouvelles images avec une qualité acceptable. Se pose donc dans ce cas le problème de l'existence d'un dictionnaire universel permettant de restituer n'importe quelle séquence avec une bonne qualité.

- Par contre, dans le cas d'archivage de données sur un site unique (pas de transmission), la taille du dictionnaire et le débit (taille de l'image codée) sont à prendre en compte vu que l'objectif recherché est juste une compression des données pour diminuer l'espace occupé par les données.

Si la base de données est fixée, la séquence d'apprentissage utilisée pour la construction du dictionnaire est formée par l'ensemble des images de la base et il est possible d'obtenir de bons résultats avec un dictionnaire unique ; par contre si la base de données est évolutive, revient le problème du dictionnaire universel.

Comme nous pouvons le constater, la construction d'un QV performant en toutes circonstances est un problème très complexe qui ne sera pas traité lors de notre étude. Seuls des tests effectués en cours de projet apporteront des éléments de réponse à certaines de ces questions.
Le chapitre suivant va donc consister en l'élaboration du dictionnaire, en prenant bien conscience que la qualité du QV est fortement conditionnée par l'algorithme de construction du dictionnaire.

3 Elaboration du dictionnaire

L'élaboration du dictionnaire se fait à partir d'une séquence d'apprentissage. Les vecteurs constituant cette séquence correspondent à des blocs de l'image ou à un ensemble de coefficients si l'on se situe dans un espace transformé comme nous le verrons par la suite.

3.1 Algorithme de LBG

Cet algorithme proposé par Linde, Buzo et Gray [3] correspond à une extension de l'algorithme de Lloyd-Max [4] utilisé pour l'élaboration de dictionnaires dans le cas de la quantification scalaire. Son rôle est pour un dictionnaire initial donné d'essayer d'optimiser le codeur et le décodeur.

étape 0 : initialisation

- Fixer
  - un dictionnaire initial $Y^{(0)}$ de taille N
  - un seuil $\delta > 0$
  - un nombre maximum d'itérations p
  - une source de vecteurs x de distribution connue
- Initialiser
  - $D^{(-1)} = \infty$
  - $m = 0$

étape 1 :

- Pour le dictionnaire courant $Y^{(m)} = \{y_i^{(m)}\}_{\{i=1,\ldots,N\}}$, trouver la partition optimale $R^{(m)} = \{R_i^{(m)}\}_{\{i=1,\ldots,N\}}$ qui minimise la distorsion moyenne soit :

$$x \in R_i^{(m)} \text{si } d(x, y_i^{(m)}) < d(x, y_j^{(m)}), \forall j \neq i$$

- Calculer la distorsion moyenne

$$D^{(m)} = \sum_{i=1}^{N} E[d(x, y_i^m) | x \in R_i^{(m)}]$$

étape 2 :

- Si ($\frac{D^{(m-1)}-D^{(m)}}{D^{(m)}} \leq \delta$ ou $m = p$)
  - stopper : le QV final est décrit par $R^{(m)}$ et $Y^{(m)}$.
- Sinon
  - continuer

étape 3 :

- Pour la partition $R^{(m)}$, calculer le dictionnaire optimal $Y^{(m+1)} = y_i^{(m+1)}{}_{\{i=1,\ldots,N\}}$, avec $y_i^{(m+1)} = cent(R_i^{(m)})$.
- Incrémenter m.
- Retourner à l'étape 1.

L'abréviation *cent* utilisée dans l'algorithme correspond au `centroïde`. Le centroïde y_i d'une région R_i est dans le cas de la distance euclidienne défini par :

$$y_i = cent(R_i) = E[x|x \in R_i].$$

Cet algorithme d'optimisation itératif travaille à partir d'un dictionnaire initial. Le problème est donc reporté sur le choix de ce dictionnaire initial dont l'expérience a prouvé qu'il contribuait fortement aux performances de l'algorithme LBG. En effet, chaque itération ne provoquant qu'un changement local du dictionnaire, l'algorithme converge vers le minimum local le plus proche du dictionnaire initial.
Le nombre d'itérations maximum a été introduit lors de la mise en oeuvre du QV afin de réduire le temps de conception des dictionnaires, cependant, le nombre maximum d'itérations ayant été fixé à 20, celui-ci n'intervient que dans de rares cas et n'entraîne pas de hausse signifiante de la distorsion. Le problème est donc reporté sur la construction du dictionnaire initial.

3.2 Choix du dictionnaire initial

3.2.1 Méthodes "naïves"

Un certain nombre de méthodes ont été proposées pour l'initialisation du dictionnaire.

3.2.1.1 Initialisation aléatoire :

Le dictionnaire le plus simple est celui qui contient les N premiers vecteurs de la suite d'apprentissage ou N vecteurs extraits aléatoirement de cette suite. Ces vecteurs peuvent bien sûr ne pas être du tout représentatifs de la suite d'apprentissage et on aboutit à des résultats très médiocres.

3.2.1.2 L'algorithme à seuil :

Au lieu de prendre N vecteurs aléatoirement, on fixe une distance minimale entre les éléments du dictionnaire initial. Cette méthode permet d'obtenir une meilleure représentativité que dans le cas précédent mais n'est toujours pas satisfaisante, le seuil étant souvent difficile à déterminer puisque dépendant de la complexité de la séquence d'apprentissage.

3.2.1.3 Méthode des "vecteurs produits" :

Cette méthode nécessite de quantifier scalairement les k composantes des vecteurs de la séquence d'apprentissage sur P_k niveaux (avec $P_1.P_2 \ldots P_K = N$) et d'effectuer un produit cartésien entre les dictionnaires de base pour obtenir les N représentants initiaux. Le traitement des composantes de manière indépendante ne permet pas d'obtenir à coup sûr un dictionnaire optimal. On peut même, si l'on n'y prend pas garde, obtenir des représentants initiaux ne représentant aucun vecteur de la séquence d'apprentissage.

3.2.2 Méthode par dichotomie vectorielle

Cette méthode a été proposée dans la version initiale de l'algorithme de LBG sous le nom de "splitting" [3].
Un débit $R = log_2(N)$ entier est alors exigé. Dans ce cas, une succession de QV ayant des débits croissants de 0 à R est générée.
Le procédé de construction est le suivant :

étape 0 : initialisation

- Initialiser le dictionnaire Y en prenant $Y[0] = y_0$ où y_0 est le centroïde de la séquence d'apprentissage prise dans son ensemble.
- Noter u le vecteur de $\mathbb{R}^k$ ayant toutes ses composantes égales à 1.
- Fixer une perturbation scalaire ε.
- Fixer le débit R.
- Fixer un seuil de distorsion Δ.
- Prendre $r = 0$.

étape 1 :

- Perturber le dictionnaire courant $Y = \{y_i\}_{\{i=0,\ldots,2^r-1\}}$ selon :

$$\forall i = 0, \ldots, 2^r - 1 \begin{cases} Y[2i+1] & = & Y[i] - \varepsilon u \\ Y[2i] & = & Y[i] + \varepsilon u \end{cases}$$

- Incrémenter r

étape 2 :

- Considérer le nouveau dictionnaire $Y = \{y_i\}_{\{i=0,\ldots,2^r-1\}}$.
- Exécuter l'algorithme de LBG sur le dictionnaire Y.
- Si ($\frac{r}{k} = R$ ou $D^{(r)} < \Delta$)
  - Arrêter en retenant $Y^{(r)}$.
- Sinon
  - Retourner à l'étape 1.

Cet algorithme a été mis en oeuvre lors de la conception de notre QV.
Un organigramme simple de cet algorithme est donné par la Figure 2. Un exemple d'exécution de cet algorithme est représenté par les Figures 3, 4, 5, 6, 7.
La charge en calculs engendrée par la construction du dictionnaire est très importante [5] mais n'est pas considérée comme un handicap prépondérant étant donné qu'un dictionnaire est construit une fois pour toutes.

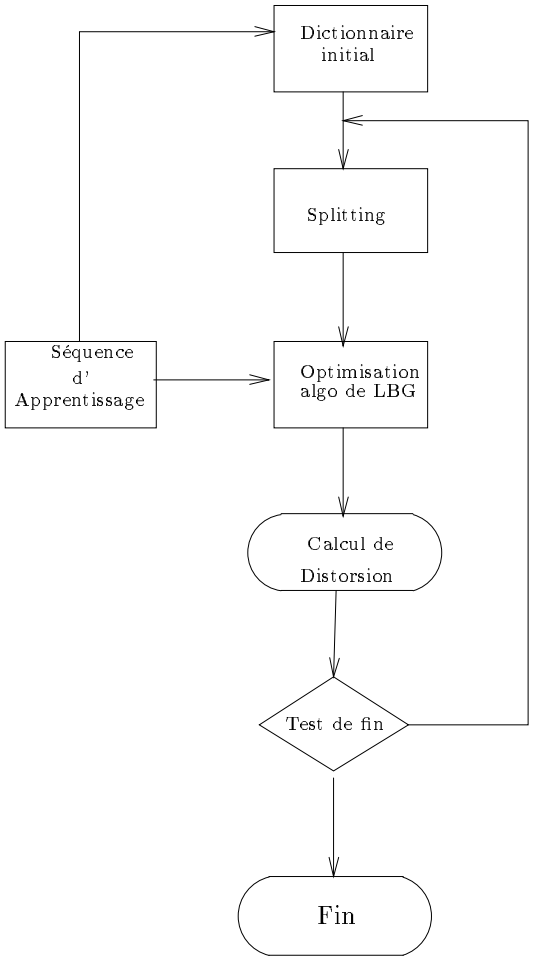

Figure 2 : organigramme de l'algorithme de LBG + méthode de *splitting*

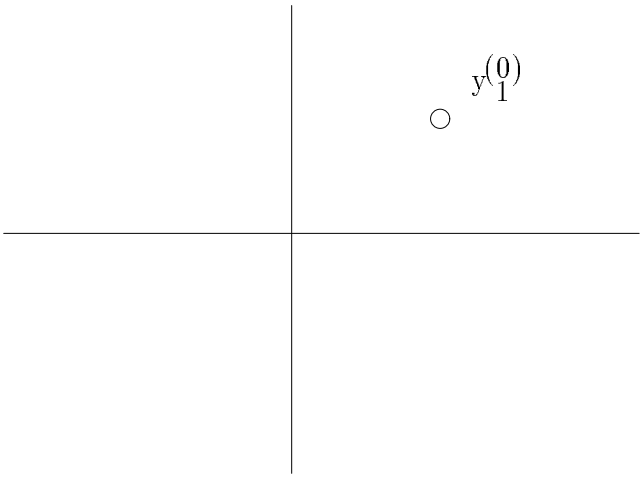

Figure 3 : centroïde de la S.A

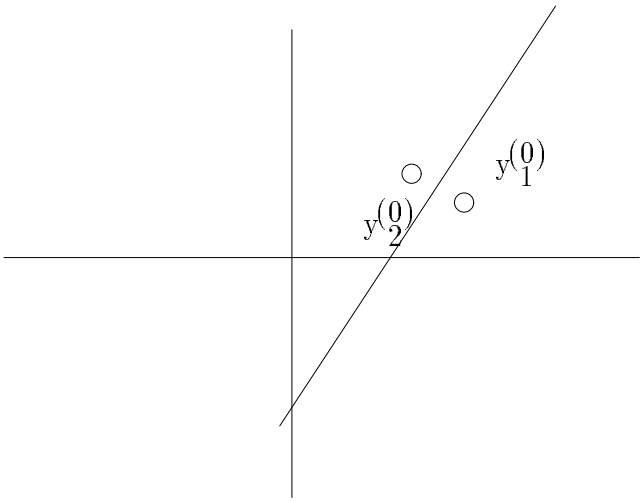

Figure 4 : perturbation 1

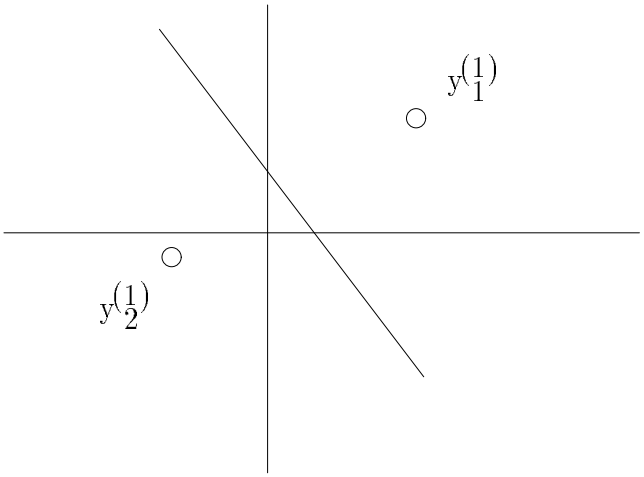

Figure 5 : optimisation 1

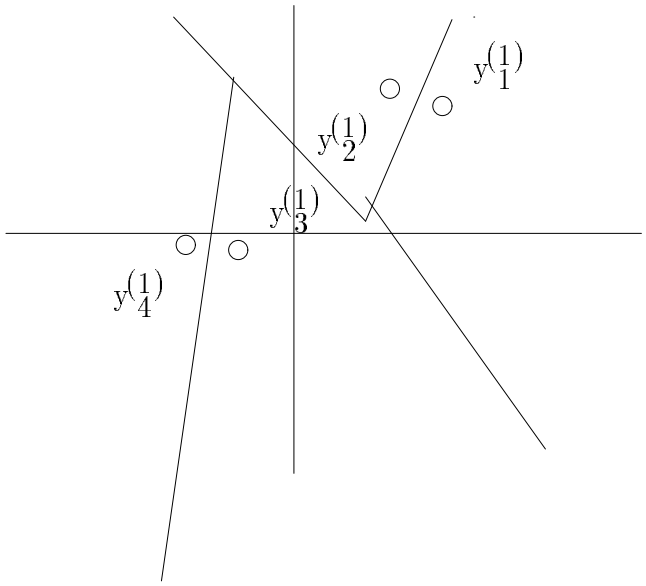

Figure 6 : perturbation 2

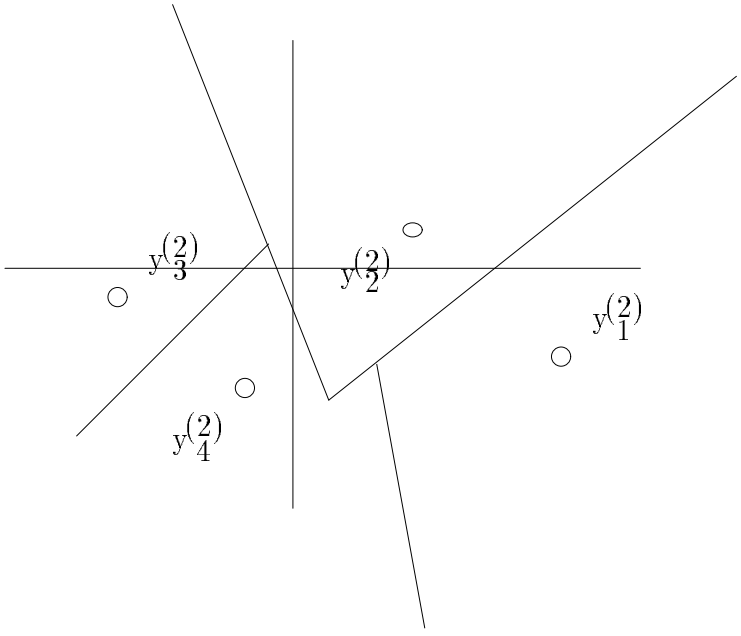

Figure 7 : optimisation 2

3.2.3 Recherche exhaustive

Comme nous l'avons vu précédemment, le codeur peut être vu comme un émetteur délivrant un code qui n'est autre que l'indice du code-vecteur le plus proche du vecteur source.
Le taux d'information du QV est donné par le débit r correspondant au nombre de bits par composante requis pour représenter le vecteur source.
Pour un dictionnaire de N représentants et avec des vecteurs à k composantes, le débit s'exprime par $r = \frac{log_2(N)}{k}$
Pour une taille de dictionnaire donnée N, et en utilisant pour calculer la distorsion la distance euclidienne, il est facile d'établir que la complexité est de l'ordre de 2^{rk}.
Il apparait donc que la complexité croît exponentiellement avec :

- **La dimension à débit fixé.**
- **Le débit à dimension fixée.**

Or des performances élevées ne sont atteignables qu'aux débits élevés mais l'opération de codage devient alors très coûteuse en calculs.
Il a donc été nécessaire d'envisager de nouvelles techniques permettant d'atteindre des résultats d'une précision acceptable tout en gardant une complexité en calculs au codage raisonnable. Des techniques permettant d'atteindre ce compromis sont les méthodes arborescentes.

4 QV à recherche arborescente

Une fois le dictionnaire construit, il faut donc mettre en place une technique de recherche rapide. La recherche exhaustive étant comme nous l'avons vu très coûteuse, des méthodes de recherche arborescente sont couramment utilisées.
Il convient de signaler que si ces techniques ne donnent pas nécessairement le plus proche représentant mais un représentant en moyenne proche, elles permettent de réduire considérablement le temps de codage et apparaissent comme un bon compromis entre la précision atteinte lors de la restitution de l'image et la vitesse de codage de celle-ci.
On distingue 2 grandes méthodes de QV arborescente :

- Une première solution consiste à intégrer la construction de l'arbre à celle du dictionnaire. Cette approche est une approche `descendante` et est la plus couramment utilisée.
- Une deuxième solution est de construire l'arborescence une fois le dictionnaire formé, c'est à dire à partir des codes-vecteurs obtenus par l'algorithme de LBG (feuilles de l'arbre) ne présentant a priori aucune structure et en effectuant une arborescence `ascendante`.

4.0.3.1 Principe de codage :

Le codeur, pour effectuer sa recherche dispose de l'arborescence et démarre sa recherche à partir de la racine de l'arbre.
A chaque étape, on calcule la distance par rapport à chacun des fils du noeud courant et on sélectionne le noeud apportant une distorsion minimale, la recherche se poursuit ensuite dans le sous-arbre ayant ce noeud comme racine et ce processus est itéré jusqu'à ce que l'on atteigne un noeud terminal de l'arbre. Le code-vecteur associé à ce noeud terminal est alors considéré comme le représentant du bloc source.

4.0.3.2 Principe de décodage :

Il convient de noter que le décodeur ne dispose généralement pas de l'ensemble des noeuds intermédiaires de l'arbre mais seulement des noeuds terminaux puisque le codeur lui transmet directement l'indice du code-vecteur qui représentera le vecteur source.
Cependant dans le cas particulier de la reconstruction progressive, le décodeur utilise les noeuds intermédiaires. Au lieu d'attendre que le vecteur source ait été complètement spécifié par le codeur, la

transmission se fait à chaque niveau de l'arbre. Au fur et à mesure que le codeur progresse dans sa recherche, le code-vecteur représentatif du vecteur source devient de plus en plus précis et est interprété par le décodeur.

4.1 Arborescences ascendantes

Ce type d'arborescence n'ayant pas été mis en oeuvre lors de cette étude, seul le principe général sera décrit.
La construction de l'arborescence se fait donc une fois le dictionnaire existant. Ce dictionnaire ayant été construit de manière tout à fait classique (algorithme de LBG par exemple) ne présente a priori aucune structure particulière. L'édification d'une classification hiérarchique ascendante consiste à former à partir de petites classes très homogènes des classes de moins en moins homogènes jusqu'à l'obtention d'une classe unique. Pour cela, il nous faut définir 2 fonctions qui sont :

- Une fonction de distance $d(.,.)$ calculant la distance entre deux vecteurs.
- Une fonction de dissimilarité $\delta(.,.)$ mesurant la dissimilarité entre deux classes de vecteurs.

La distance d utilisée est celle qui a été utilisée lors de la construction du dictionnaire.
Pour la stratégie de fusion, plusieurs fonctions de dissimilarité peuvent être utilisées [13], en voici deux exemples :

- indice du lien minimum

$$\delta_1(R_i, R_j) = \min_{x_{l_i} \in R_i, x_{l_j} \in R_j} d(x_{l_i}, x_{l_j})$$

- indice de la moyenne des distances

$$\delta_2(R_i, R_j) = \frac{1}{L_i L_j} \sum_{x_{l_i} \in R_i, x_{l_j} \in R_j} d(x_{l_i}, x_{l_j})$$

avec $L_i = card[R_i]$ et $L_j = card[R_j]$

L'algorithme utilisé pour construire l'arborescence est l'algorithme de Classification Ascendante Hiérarchique [6] que nous décrivons ci-dessous :

étape 0 : initialisation

- A partir de N classes disjointes R_i, calculer les paires $\delta(R_i, R_j)$

étape 1 :

- Déterminer dans la hiérarchie courante, le couple le plus similaire (R_i, R_j) au sens de la dissimilarité $\delta(.,.)$
- Si nécessaire, calculer le centroïde de $R_i \cup R_j$

étape 2 :

- Mettre à jour les distances

$$\delta(R_k, R_i \cup R_j), \forall k \neq i, k \neq j$$

étape 3 :

- S'il reste une seule classe, arrêter.
- Sinon aller à l'étape 1.

4.2 Arborescences descendantes

Ce type d'arborescence a été mis en oeuvre dans le développement de cette étude et va donc être développé plus complètement dans ce rapport.

4.2.1 Arborescence descendante équilibrée

Les noeud terminaux sont situés sur une même couche et leur hauteur H est donnée par la relation :

$$H = log_2(N)$$

4.2.1.1 Complexité du codage : ‘

A chaque niveau de recherche dans l'arbre, on calcule la distorsion associée à la représentation du vecteur source par chacun des fils du noeud courant (nous étant placé dans le cas d'un arbre binaire, le nombre de fils est égal à 2). La complexité de l'algorithme de recherche est donc $2H$ au lieu de 2^H pour une recherche exhaustive, soit un gain de $\frac{2H}{2^H} = 2^{H-1}H$.
Pour donner un ordre de grandeur, considérons un dictionnaire de profondeur 10 ou ce qui est équivalent de taille 1024 éléments :

- Le nombre de calculs de distorsion par bloc de l'image pour une recherche exhaustive est $2^{10} = 1024$
- Le nombre de calculs de distorsion par bloc de l'image pour une recherche arborescente est $2 * 10 = 20$
- La complexité a donc été divisée par $\frac{1024}{20} = 51.2$

4.2.1.2 Taille du dictionnaire : ‘

Il a fallu stocker non plus simplement les code-vecteurs mais toute l'arborescence du dictionnaire. Le nombre de noeuds non terminaux est égal à $2^H - 1$, il faut bien sûr rajouter le nombre de noeuds terminaux qui est 2^H. Le nombre de noeuds à stocker est donc $2^H - 1 + 2^H = 2^{H+1} - 1$, soit environ un facteur 2 par rapport au cas exhaustif. Toutefois, étant donné le coût très peu élevé de la mémoire dans les technologies actuelles, cet espace supplémentaire ne peut pas être considéré comme un handicap sérieux.

4.2.1.3 Construction du dictionnaire : ‘

La construction du dictionnaire repose sur le principe de la dichotomie vectorielle introduit précédemment. Lorsque l'on travaillera à une hauteur h, chaque noeud sera divisé en 2 noeuds qui constitueront l'arbre de hauteur $h + 1$.
En sortie, l'algorithme de construction du dictionnaire va fournir 2 dictionnaires :

- Un dictionnaire des noeuds terminaux.
- Un dictionnaire des noeuds intermédiaires.

Lors de la réalisation de ce QV, nous avons choisi comme structure pour ces dictionnaires des tableaux. Dans le tableau des noeuds intermédiaires, un noeud d'indice i aura pour fils les noeuds d'indices $2i+1$ et $2i+2$. Les noeuds de la dernière couche des noeuds intermédiaires ont leur indice compris entre $2^{H-1}-1$ et 2^H-2. Pour un de ces noeuds d'indice $i = 2^{H-1}-1+j$ les fils se situent dans le dictionnaire des noeuds terminaux aux indices $2j$ et $2j+1$.
L'algorithme de construction du dictionnaire est le suivant :

étape 0 : initialisation

- Noter Y le dictionnaire des noeuds terminaux.
- Fixer $Z[0] = z_0$ où z_0 est le centroïde de la séquence d'apprentissage prise dans son ensemble, Z étant le dictionnaire des noeuds intermédiaires.
- Noter u le vecteur de $\mathbb{R}^k$ ayant toutes ses composantes égales à 1.
- Fixer une perturbation scalaire ε.
- Fixer le débit R ou la profondeur H.
- Fixer un seuil de distorsion Δ.
- Initialiser $r = 0$ (r profondeur courante de l'arbre).
- Initialiser $indice = 0$.

étape 1 :

- Perturber le dictionnaire courant Y selon :

$$\forall i = 0, \ldots, 2^r - 1 \begin{cases} Y[2i] & = & Z[indice+i] - \varepsilon u \\ Y[2i+1] & = & Z[indice+i] + \varepsilon u \end{cases}$$

- Calculer $indice = indice + 2^{(r)}$
- Incrémenter r

étape 2 :

- Considérer le dictionnaire $Y = \{y_i\}_{\{i=1,\ldots,2^r\}}$.
- Exécuter l'algorithme de LBG sur le dictionnaire Y.
- Si ($\frac{r}{k} = R$ ou $D^{(r)} < \Delta$)
  - arrêter en retenant Y.
- Sinon
  - recopier les noeuds du dictionnaire Y dans le dictionnaire Z selon :

$$\forall i = 0, \ldots, 2^r - 1 : Z[indice+i] = Y[i]$$

  - retourner à l'étape 1

Cependant, la QV par méthode arborescente équilibrée est sous-optimale par rapport à la QV exhaustive et ceci pour deux raisons qu'il convient de distinguer :

- sous-optimalité du dictionnaire : Celui-ci contient moins de représentants pertinents que dans le cas exhaustif du fait de la contrainte à laquelle il est soumis lors de sa construction.
- sous-optimalité du codage : Du fait du procédé de recherche dans l'arbre, le code-vecteur trouvé n'est pas obligatoirement le code-vecteur le plus proche du vecteur source.

4.2.2 Arborescence descendante non-équilibrée

Le fait d'avoir un débit variable présente l'avantage de distribuer efficacement les bits selon l'activité des vecteurs. Le codeur affecte plus de bits aux régions actives et peut gagner un nombre de bits non-négligeable pour coder les régions homogènes.
Dans la construction d'un QV arborescent, il suffit pour avoir un débit variable de construire une arborescence non-équilibrée, c'est à dire que tous les noeuds terminaux ne se situent pas sur la même couche. La longueur du mot de code associée à chaque feuille est égale à la profondeur de la feuille dans l'arbre.
Les techniques de construction d'arborescences descendantes non-équilibrés peuvent être divisées en deux parties :

- Algorithmes par découpage → La construction de l'arbre non-équilibré se fait lors de la construction du dictionnaire grâce à une application non-systématique de la méthode de splitting.
- Algorithmes par élaguage → On construit tout d'abord un arbre équilibré par une méthode classique, cet arbre est ensuite élagué afin d'assurer le meilleur compromis entre la hausse de distorsion et la baisse de débit.

4.2.2.1 définitions : ‘

Les algorithmes de création d'arbre utilisent des séquences d'apprentissage. A l'issue du codage de la séquence, supposée de longueur L, on définit R_n comme la cellule de codage associée à un noeud n quelconque. Plus précisément, R_n désigne l'ensemble des vecteurs de la séquence d'apprentissage dont le chemin de codage passe par le noeud n.
On définit alors :

- La probabilité $p(n)$ du noeud n comme le taux de vecteurs de la séquence d'apprentissage dans R_n donné par :

$$p(n) = \frac{card[R_n]}{L}$$

- La distorsion $\Delta(n)$ associée au noeud n par :

$$\Delta(n) = \frac{1}{card[R_n]} \sum_{x \in R_n} \|x - y_n\|^2$$

où y_n est le centroïde de la cellule R_n

4.2.2.2 Algorithmes par découpage : ‘

A la différence de la construction d'un arbre équilibré, au lieu de diviser chaque noeud de la couche h courante en ses deux fils, on éclate seulement le noeud qui semble a priori le plus intéressant selon un critère qui va être défini par la suite. Ce noeud n'appartient pas à une couche particulière mais présente comme unique contrainte d'être temporairement un noeud terminal.
Plusieurs critères ont été définis pour déterminer quel noeud est le plus intéressant :

- Le critère le plus naturel est de prendre le noeud qui présente la plus grande baisse de distorsion.
- Un deuxième critère paraissant tout aussi concevable d'un point de vue logique serait de diviser le noeud contribuant le plus à la distorsion globale. De plus, ce critère paraît moins lourd en charge de calculs que le premier puisqu'il n'est plus nécessaire de calculer la distorsion que provoquerait l'éclatement de chacun des noeuds terminaux (ce qui revient à construire un arbre de profondeur moyenne $h' = h + 1$ où h est la profondeur moyenne de l'arbre courant).
- Un troisième critère consiste à choisir le noeud qui assure le meilleur compromis "baisse de la distorsion/hausse de débit". Pour chaque noeud terminal n qui est découpé en n_1 et n_2, on calcule :

– la variation de distorsion (diminution) égale à :

$$p(n)\Delta(n) - (p(n_1)\Delta(n_1) + p(n_2)\Delta(n_2)).$$

– la variation de débit (augmentation) égale à :

$$p(n)h(n) - (p(n_1)h(n_1) + p(n_2)h(n_2)).$$

Ces deux variations étant antagonistes, un compromis doit être trouvé. Pour cela, on mesure le rapport entre la diminution de la distorsion et la hausse de débit résultant du découpage de la feuille n, ce rapport est appelé retour marginal et est noté $\lambda(n)$. Par définition, $\lambda(n)$ est donné par la relation suivante :

$$\lambda(n) = -\frac{p(n)\Delta(n) - (p(n_1)\Delta(n_1) + p(n_2)\Delta(n_2))}{p(n)h(n) - (p(n_1)h(n_1) + p(n_2)h(n_2))}.$$

C'est la feuille n fournissant la plus grande valeur de retour marginal qui va être découpée [7] [8].

4.2.2.3 Algorithme d'élaguage optimal: ‘

Cette méthode a été mise en oeuvre lors de cette étude. L'algorithme utilisé est l'algorithme de BFOS du nom de ses auteurs Breiman, Friedman, Olshen et Stone [9].
La première étape, avant même l'application de cet algorithme consiste à construire un dictionnaire de taille importante, ce dictionnaire étant élagué par la suite. Le but de l'élaguage est de supprimer les parties de l'arbre n'ayant pas une contribution importante dans la baisse de la distorsion par rapport à leur contribution dans l'augmentation du débit. Plus concrètement, cela revient à supprimer les découpages de zones homogènes qui ont été effectués lors de la construction de l'arbre. Il est à noter que comme dans le cas de l'algorithme de découpage, l'algorithme de BFOS est basé sur le principe du retour marginal.

4.2.2.3.1 Principe de l'algorithme: ‘

Le principe de l'algorithme de BFOS est en fait relativement simple. Pour chaque noeud n non-terminal de l'arbre, on calcule le rapport marginal $\lambda(n)$. On élague alors l'arbre à partir du noeud n_i ayant le plus petit rapport marginal, c'est à dire que l'on remplace dans l'arbre la sous-branche S_{n_i} issue du noeud $n_i(S_{n_i})$ par le noeud n_i. On met alors à jour le rapport $\lambda(n)$ associé à tous les ascendants de n_i (pour les autres noeuds de l'arbre, ce rapport est inchangé) et on réitère ce procédé jusqu'à l'obtention d'un débit donné ou d'une distorsion maximum fixée.

4.2.2.3.2 Description de l'algorithme: ‘

Nous allons tout d'abord décrire la structure de données qui a été utilisée lors de la mise en oeuvre pour stocker les informations relatives à chaque noeud de l'arbre. En plus du tableau contenant le dictionnaire, nous avons construit un tableau contenant l'ensemble des noeuds de l'arborescence, chaque élément du tableau étant une structure de type :

- $\Delta l(S_t) \rightarrow$ baisse de débit occasionnée par le remplacement de la sous-branche d'origine t par le noeud t.
- $\Delta\delta(S_t) \rightarrow$ hausse de distorsion occasionnée par le remplacement de la sous-branche d'origine t par le noeud t.
- $\lambda(t) = -\frac{\Delta\delta(t)}{\Delta l(t)} \rightarrow$ valeur de retour marginal pour le noeud t.
- $\lambda_{min}(t) \rightarrow$ valeur minimale du retour marginal pour les descendants de t.

L'algorithme de BFOS est précédé d'une phase d'initialisation au cours de laquelle les champs de la structure sont initialisés par rapport à l'arbre initial [10].

Phase d'initialisation

- Pour chaque feuille t de l'arbre :
  - $\Delta l(S_t) = 0$
  - $\Delta \delta(S_t) = 0$
  - $\lambda_{min}(t) = \infty$
- Pour chaque noeud t non-terminal de l'arbre :
  - $\Delta l(S_t) = l(S_t) - l(t)$
  - $\Delta \delta(S_t) = \delta(S_t) - \delta(t)$
  - $\lambda(t) = -\frac{\Delta \delta(S_t)}{\Delta l(S_t)}$
  - $\lambda_{min}(t) = \min\{\lambda(t), \lambda_{min}(t_L), \lambda_{min}(t_R)\}$ où t_L et t_R sont respectivement les fils gauche et droit de t

Par définition, on prend :

- $l(t) = p(t)h(t)$ où $h(t)$ est la profondeur du noeud t
- $l(S_t) = \sum_{t \in \overline{S}} p(t)h(t)$ où $\overline{S}$ est l'ensemble des feuilles de l'arbre S
- $\delta(t) = p(t)\Delta(t)$
- $\delta(S_t) = \sum_{t \in \overline{S}} p(t)\Delta(t)$

Pour tout noeud non-terminal n_i de l'arbre de fils n_{2i+1} et n_{2i+2}, il est facile de montrer que :

$$\begin{cases} \Delta l(S_{n_i}) = \Delta l(S_{n_{2i+1}}) + l(n_{2i+1}) + \Delta l(S_{n_{2i+2}}) + l(n_{2i+2}) - l(n_i) \\ \Delta \delta(S_{n_i}) = \Delta \delta(S_{n_{2i+1}}) + \delta(n_{2i+1}) + \Delta \delta(S_{n_{2i+2}}) + \delta(n_{2i+2}) - \delta(n_i) \end{cases}$$

Pour l'initialisation, nous avons utilisé une procédure récursive qui, étant données la fonction de répartition p de la séquence d'apprentissage vis-à-vis du dictionnaire, la profondeur h de chacun des noeuds dans l'arbre ainsi que la distorsion Δ associée a chaque noeud initialise la structure de données dans l'ensemble de l'arbre. Cette procédure prend en paramètres "principaux" l'indice du noeud courant et le dictionnaire (toute l'arborescence). Lors de l'appel, le noeud courant est la racine de l'arbre. De plus les champs correspondant aux feuilles de l'arbre ont été préalablement initialisés.
Voici une description sommaire de cette procédure :

proc initialise (t_i : *t-noeud*, A : *t-arbre*);
`Debut`
  Si non (feuille (t_i, A))
    initialise (t_{2i+1}, A);
    initialise (t_{2i+2}, A);
    $\Delta l(S_{t_i}) = \Delta l(S_{t_{2i+1}}) + l(t_{2i+1}) + \Delta l(S_{t_{2i+2}}) + l(t_{2i+2}) - l(t_i)$
    $\Delta \delta(S_{t_i}) = \Delta \delta(S_{t_{2i+1}}) + \delta(t_{2i+1}) + \Delta \delta(S_{t_{2i+2}}) + \delta(t_{2i+2}) - \delta(t_i)$
    $\lambda(t_i) = -\frac{\Delta \delta(S_{t_i})}{\Delta l(S_{t_i})}$;
    $\lambda_{min}(t_i) = \min\{\lambda(t_i), \lambda_{min}(t_{2i+1}), \lambda_{min}(t_{2i+2})\}$
  Fsi
`Fin`

Algorithme de BFOS
L'algorithme mis en oeuvre dans le cadre de notre étude est très proche de l'algorithme proposé à l'origine [10]. Toutefois, la condition d'arrêt a été modifiée puisque dans la version originale, on construit l'ensemble des arbres élagués jusqu'à ne plus avoir que la racine alors que nous nous contenterons

de construire les sous-arbres élagués ayant un débit supérieur à un débit fixé fourni comme paramètre d'entrée au programme.
La version de l'algorithme de BFOS utilisée est donc la suivante :

étape 0 :

- $l\text{-}arbre = l(t_0) + \Delta l(S_{t_0})$
- $\delta\text{-}arbre = \delta(t_0) + \Delta\delta(S_{t_0})$

Les variables $l\text{-}arbre$ et $\delta\text{-}arbre$ contiennent respectivement la profondeur moyenne de l'arbre et la distorsion engendrée par le codage de la séquence d'apprentissage par les noeuds terminaux de cet arbre. Ces valeurs seront mises à jour après chaque opération d'élaguage.

étape 1 :

- Si (debit $\geq$ debit-fixé)
  - $i = 0$ /* On prend la racine de l'arbre comme noeud courant. */
- Sinon
  - arrêter

étape 2 :

- Tant que $\lambda(t_i) > \lambda_{min}(t_0)$
  - Si $\lambda_{min}(t_{2i+1}) = \lambda_{min}(t_0)$
    - $i = 2i + 1$
  - Sinon
    - $i = 2i + 2$

/* On cherche dans l'arborescence, le noeud ayant le λ minimum. */

étape 3 :
Le noeud courant correspond à la racine de la sous-branche qui doit être élaguée.

- $\Delta l\text{-}arbre = \Delta l(S_{t_i})$
- $\Delta\delta\text{-}arbre = \Delta\delta(S_{t_i})$
- $\lambda_{min}(t_i) = \infty$

étape 4 :

- Tant que $i \neq 0$
  - $i = \frac{i-1}{2}$ /* On se positionne sur le père de t_i */
  - $\Delta l(S_{t_i}) = \Delta l(S_{t_i}) - \Delta l\text{-}arbre$
  - $\Delta\delta(S_{t_i}) = \Delta\delta(S_{t_i}) - \Delta\delta\text{-}arbre$

- $\lambda(t_i) = -\frac{\Delta\delta(S_{t_i})}{\Delta l(S_{t_i})}$
- $\lambda_{min}(t_i) = \min\{\lambda(t_i), \lambda_{min}(t_{2i+1}), \lambda_{min}(t_{2i+2})\}$

étape 5 :

- $l\text{-}arbre = l\text{-}arbre - \Delta l\text{-}arbre$
- $\delta\text{-}arbre = \delta\text{-}arbre - \Delta\delta\text{-}arbre$

Le sous-arbre élagué a été construit, et la distorsion et profondeur moyenne de l'arbre ont été calculées.

- retourner à l'étape 1

Cet algorithme est dit optimal car il a été prouvé [14] qu'il ne générait que des sous-arbres optimaux au sens du rapport débit/distorsion.

5 Choix d'un espace de représentation

5.1 Intérêt de passer dans un domaine transformé

Une image est habituellement représentée sous la forme d'une matrice de pixels, l'espace de représentation étant généralement celui des luminances. A l'intérieur de cet espace, il existe souvent des redondances d'ordre statistique, la luminance d'un pixel étant fortement corrélée par avec celle de ses voisins.
Aussi, avant d'appliquer de la QV sur une image, il s'est avéré performant de changer d'espace de représentation ; aussi l'image n'est plus représentée dans le domaine des luminances mais dans un domaine proche de celui des fréquences spatiales. Chaque coefficient de l'image ne correspond alors plus à une luminance mais est associé à une activité fréquentielle déterminée dans le contexte du codage en sous-bandes ou par transformée .
Ce changement d'espace vectoriel a pour effets :

- Une concentration de l'énergie des blocs sur quelques coefficients (typiquement associés aux basses fréquences).
- L'élimination de certaines redondances statistiques présentes dans les blocs originaux.

5.2 Choix de la transformée

Au cours de notre étude, deux transformées ont été testées [11] [12]. Ces transformées sont des transformées linéaires par blocs bidimensionnels séparables :

- Une transformée par bloc : la DCT.
  Dans le cas monodimensionnel, l'opération pour passer du domaine des luminances au domaine transformé est la suivante : $X = A^T x$ si x est le bloc original et X le bloc transformé.
  La matrice de transformation A est la suivante :

$$A = [a_{n\,k}] = c(k).\sqrt{\frac{2}{N}}.cos\left[\frac{(n+\frac{1}{2}).k.\pi}{N}\right]$$

  $n, k = 0,\ldots,$N-1 , k indice de fréquence, n indice temporel/spatial.

$$c(k) = \begin{cases} \sqrt{\frac{1}{2}} & \text{si } k = 0 \\ 1 & \text{sinon} \end{cases}$$

  où N est la taille des blocs de la transformée.

- Entre transformée par bloc et décomposition en sous-bandes : la MLT).
  Dans le cas de la MLT, on projette le signal sur des bases de fonctions de taille $2N$ où N est la taille des blocs. Ces fonctions sont constituées d'une fonction "fenêtre" régulière $h(n)$ modulée par une fonction trigonométrique ici en cosinus.

$$f_k(n) = h(n).\sqrt{\frac{2}{N}}.cos\left[\left(n+\frac{N+1}{2}\right).\left(k+\frac{1}{2}\right).\frac{\pi}{N}\right]$$

  avec $h(n) = sin\left[\frac{n.\pi}{2.(N-1)}\right]$
  $k = 0,\ldots,N-1, n = 0,\ldots,2N-1$
  Du point de vue du filtrage/transformation, les fonctions de base sont les réponses impulsionnelles des filtres de synthèse. Les filtres d'analyse sont des versions "time-reversed" des filtres de synthèse.
  Pour cette transformée, le signal de taille N est projeté sur une fonction de base de largeur $2N$, ceci ayant comme effet de diminuer les effets de bloc par rapport à la DCT [19]. Du fait de l'étendue finie du signal image, pour ne pas modifier la dimension du signal, il est donc nécessaire d'étendre le signal aux bords (de $\frac{N}{2}$ coefficients de chaque côté).

Les différences quant aux performances de ces deux transformations sont à l'avantage de la MLT [19] [12], en effet par rapport à l'algorithme de DCT utilisé, on peut noter :

- Une diminution des effets de blocs.
- Une concentration plus importante de l'énergie sur les composantes continues (efficacité globale de codage supérieure).

On peut estimer a priori que la MLT sera plus adaptée que la DCT dans le cadre de la QV.
Dans les deux cas, la transformée se fait par blocs, elle est représentée sur la Figure 8 où F_i^a représente l'activité fréquentielle du bloc A dans la bande fréquentielle i (ici une décomposition 2×2 bandes).

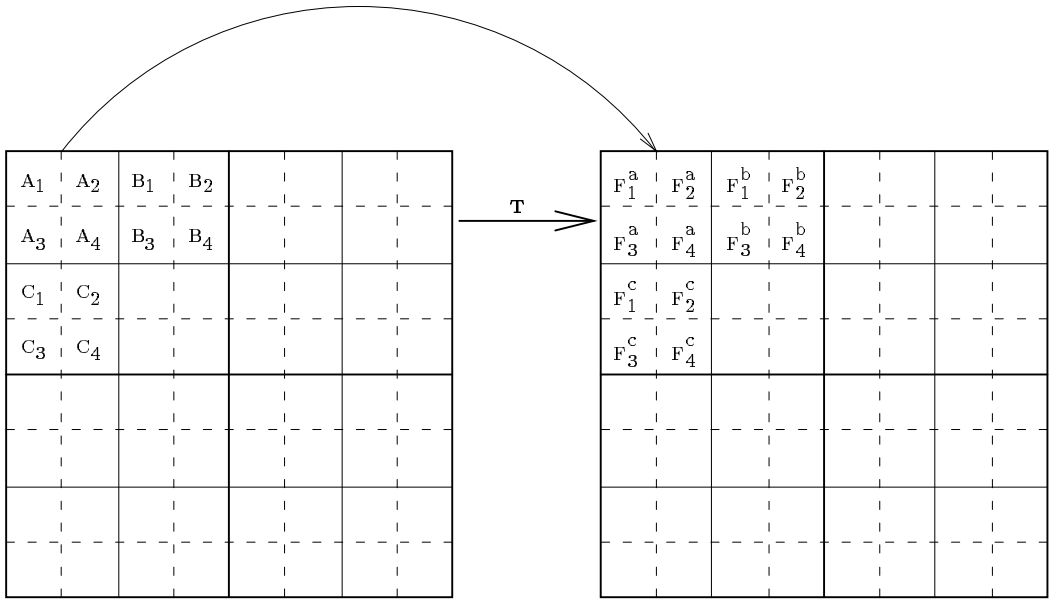

Figure 8 : passage dans le domaine fréquentiel

5.3 Méthode de quantification

Lorsque l'on travaille dans le domaine fréquentiel, il est possible d'organiser la matrice transformée de deux manières distinctes de façon à prendre des blocs de nature différente.

Ainsi, on peut décomposer l'image en sous-bandes, c'est à dire en regroupant les coefficients d'une même sous-bande fréquentielle et en effectuant la QV sur chacune de ces sous-bandes (intra-bande) ou en gardant une correspondance spatiale entre les blocs de l'image originale et ceux de l'espace transformé (inter-bande).

5.3.1 QV intra-bande

L'organisation des blocs de la matrice transformée a été représentée sur la Figure 9.

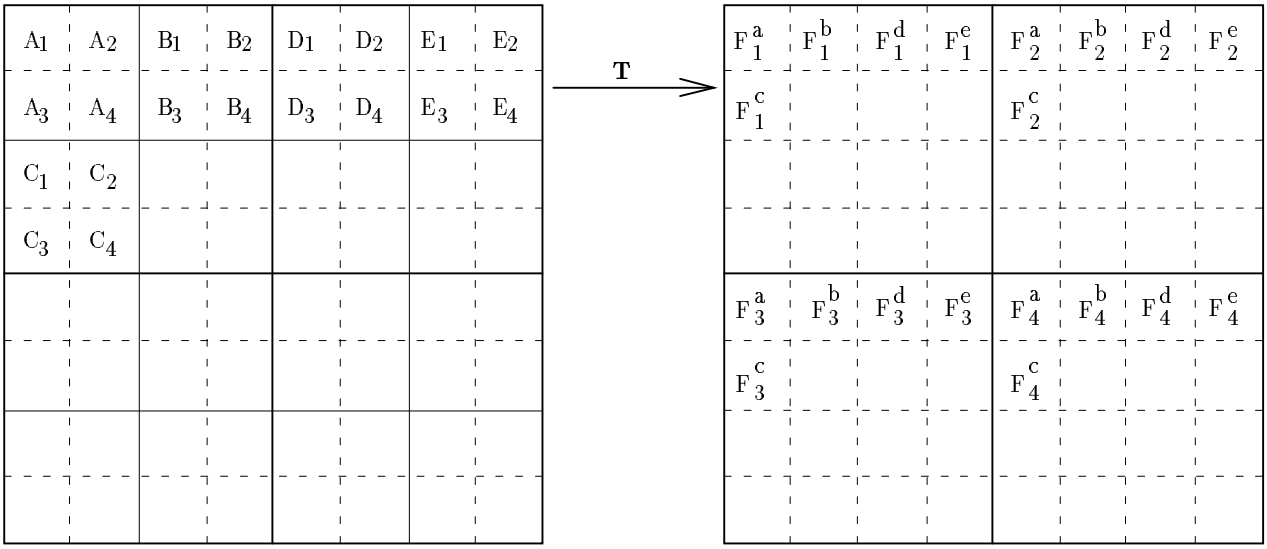

Figure 9 : représentation intra-bande

Les avantages de la QV intra-bande sont :

- L'énergie est concentrée dans une partie de l'image transformée (composantes basses et moyennes fréquences), il est donc possible d'accentuer la QV sur cette partie de l'image par rapport aux hautes fréquences.
- On va pouvoir construire un dictionnaire associé à chaque sous-bande, disposant ainsi d'un large éventail de stratégies possibles. La taille des blocs de quantification pourra être variable selon la fréquence. Ainsi, on effectuera souvent de la quantification scalaire sur les composantes continues, on utilisera des blocs de petite taille pour les basses fréquences et des blocs de taille plus importante pour les hautes fréquences. Dans certains cas, il sera même possible de ne pas quantifier les sous-bandes correspondant aux très hautes fréquences.

De plus, l'oeil humain ne perçoit pas de manière identique les différentes composantes fréquentielles obtenues. Aussi, étant donnée une distorsion maximum tolérée sur l'ensemble de l'image, il est possible de répartir cette distorsion dans chacune des sous-bandes afin de minimiser l'impact visuel de cette distorsion. Nous introduirons donc une matrice de coefficients `psychovisuels` qui sera justement chargée de cette répartition. Les caractéristiques de cette matrice sont données en Annexe A.

5.3.2 QV inter-bande

Les blocs considérés lors de la quantification correspondent aux blocs qui ont été utilisés lors de la transformation afin de tirer bénéfice des éventuelles dépendances entre bandes de même localisation. Toutefois, les composantes continues ont été enlevées et il sera fait de la quantification intra-bande sur ces coefficients. La disposition des blocs peut donc être représentée par la Figure 10.

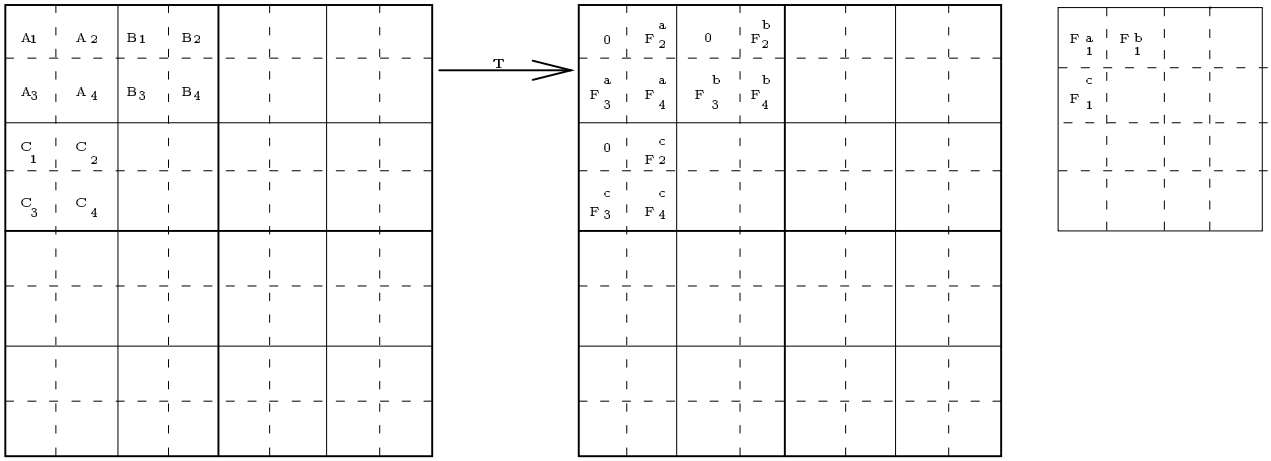

Figure 10 : représentation inter-bande

Les inconvénients de cette représentation sont bien sûr antagonistes aux avantages de la représentation intra-bande :

- Etant donné que tous les blocs, hormis les blocs contenant les composantes continues, ont une même structure et contiennent à priori une énergie comparable, on perd la possibilité d'allouer différents débits dans chacune des bandes.
- La taille des blocs de quantification doit être égale à celle des blocs de la transformée.

Un avantage que possède la QV inter-bande sur la QV intra-bande est la place occupée par les dictionnaires. En effet, dans le cas inter-bande, on a construit uniquement deux dictionnaires (un pour les composantes continues, l'autre pour les composantes fréquentielles) alors que dans le cas intra-bande, on a construit un dictionnaire par sous-bande (16 dans le cas de notre application puisque les blocs utilisés lors de la transformée sont de taille 16).

6 Expérimentations et résultats

La base de données utilisée pour tester nos algorithmes est formée de 4 images "classiques" fixes. Ces images ont été choisies car possédant des structures diverses et doivent donc permettre de tester notre QV sur des images de complexité différente (cf Annexe B).

- *Lena* (512 × 512) pixels.
- *Barbara* (512 × 512) pixels.
- *Interview* (536 × 674) pixels.
- *Cornouaille* (576 × 720) pixels.

Avant d'appliquer la QV sur ces images, nous donnons un bref rappel des outils méthodologiques employés :

- Algorithme de construction des dictionnaires : LBG associé à la méthode de Splitting.
- Algorithme d'élaguage : BFOS.

Les tests sont effectués sur des images dans un espace transformé. La taille des blocs pour passer dans l'espace transformé est de 16 pixels (4 × 4).
Tous les QV implantés ont été comparés lors de ces tests.
Les définitions des valeurs utilisées pour comparer les résultats de nos tests sont rappelées en Annexe A.

6.1 Différents QV et modes de quantification

La série de tests vise à comparer différents QV mis en oeuvre au cours de cette étude. Ces tests doivent permettre de retrouver des résultats que l'on suppose déjà d'un point de vue théorique :

- La non-optimalité au sens débit/distorsion d'un QV à recherche arborescente par rapport à un QV à recherche exhaustive.
- La supériorité de la transformation MLT par rapport à la DCT dans le cadre de la QV.
- Le gain de la quantification intra-bande par rapport à la quantification inter-bande.
- L'importance de l'introduction d'un facteur psychovisuel.

Les configurations utilisées pour la construction des dictionnaires dans le cas d'une quantification intra-bande sont représentées sur la Figure 11. Ce sont ces configurations qui seront utilisées tout au long de ces tests sauf spécification contraire. Il est facile de s'apercevoir que la troisième configuration n'est en fait pas réaliste, comprenant uniquement pour les composantes continues un dictionnaire de 16384 représentants, chaque représentant étant un vecteur de taille 16. On a ainsi largement dépassé la taille de la sous-bande. Cette configuration a été envisagée juste dans le but de voir le comportement des différents QV dans les bas-débits.

6.1.1 Comparaison intra-bande/inter-bande

Cette série de tests est effectuée sur les différentes images de notre base de données.
Pour ces comparaisons des QV à recherche exhaustive ont été construits, le dictionnaire respectif d'une image est conçu à partir de cette même image, la transformée utilisée est la DCT.

- Pour l'image *lena*

Débit	intra/inter	Distorsion	PSNR	Entropie
0.969	intra	23.13	34.49	0.853
0.969	inter	31.35	33.17	0.833
0.438	intra	24.53	34.23	0.348
0.438	inter	111.62	27.65	0.438

- Pour l'image *Barbara*

Débit	intra/inter	Distorsion	PSNR	Entropie
0.969	intra	51.13	31.04	0.816
0.969	inter	50.32	31.11	0.837
0.438	intra	62.11	30.20	0.335
0.438	inter	231.10	24.49	0.388

- Pour l'image *Interview*

Débit	intra/inter	Distorsion	PSNR	Entropie
0.969	intra	23.68	34.39	0.849
0.969	inter	28,70	33.55	0.837
0.438	intra	24.94	34.16	0.359
0.438	inter	123.58	27.21	0.347

- Pour l'image *Cornouaille*

Débit	intra/inter	Distorsion	PSNR	Entropie
0.969	intra	18.87	35.37	0.809
0.969	inter	23.91	34.34	0.791
0.438	intra	23.99	34.33	0.406
0.438	inter	109.95	27.72	0.354

Débit = 0.969

128 (scalaire)	256 (2*2)	256 (4*4)	256 (4*4)
256 (2*2)	256 (4*4)	256 (4*4)	256 (4*4)
256 (4*4)	256 (4*4)	0	0
256 (4*4)	256 (4*4)	0	0

Débit = 0.438

4096 (2*2)	4096 (4*4)	256 (4*4)	0
4096 (4*4)	256 (4*4)	256 (4*4)	0
256 (4*4)	256 (4*4)	0	0
0	0	0	0

Débit = 0.215

16384 (4*4)	1024 (4*4)	128 (4*4)	0
1024 (4*4)	128 (4*4)	0	0
128 (4*4)	0	0	0
0	0	0	0

Figure 11 : configuration des dictionnaires

La supériorité de l'intra-bande existe dès les hauts-débits mais devient flagrante essentiellement dans les bas-débits. En effet, le facteur prépondérant est la possibilité d'accentuer la quantification (accorder un débit plus important) dans les zones de l'image où est concentrée l'information. Aussi, dans le cas de la QV intra-bande à bas débit, il est possible de coder correctement les basses fréquences et de ne pas coder les hautes fréquences contenant un minimum d'informations. Par contre, dans le cas de la quantification inter-bande, le débit est uniformément réparti et est donc insuffisant pour coder certains coefficients de l'image

6.1.2 Comparaison DCT/MLT

Il s'agit de comparer ici deux types de décompositions de type bloc, l'une classiquement utilisée dans les applications de type codage (DCT), l'autre entre transformée et décomposition en sous-bandes présentant des performances de codage théoriquement supérieures (MLT), au niveau du codage, l'avantage de la MLT étant surtout la réduction des "effets de bloc" constatés pour la DCT dans les faibles débits. A priori, selon les propriétés de ces deux transformées, la MLT semble plus appropriée à la QV. On a testé les performances d'un QV après l'une ou l'autre de ces deux transformées.
La série de tests a été réalisée à nouveau à partir des images de notre base de données avec un QV à recherche exhaustive, le mode de quantification choisi est l'intra-bande.

- Pour l'image *Lena*

Débit	**Transformée**	**Distorsion**	**PSNR**	**Entropie**
0.969	DCT	23.13	34.49	0.853
	MLT	18.26	35.51	0.861
0.438	DCT	24.53	34.23	0.348
	MLT	19.61	35.21	0.355
0.215	DCT	37.50	32.39	0.151
	MLT	29.99	33.36	0.155

- Pour l'image *Barbara*

Débit	**Transformée**	**Distorsion**	**PSNR**	**Entropie**
0.969	DCT	51.13	31.04	0.816
	MLT	35.93	32.58	0.819
0.438	DCT	62.11	30.20	0.335
	MLT	41.89	31.91	0.330
0.215	DCT	105.24	27.91	0.145
	MLT	79.86	29.11	0.145

- Pour l'image *Interview*

Débit	**Transformée**	**Distorsion**	**PSNR**	**Entropie**
0.969	DCT	23.68	34.39	0.849
	MLT	18.55	35.45	0.860
0.438	DCT	24.94	34.16	0.359
	MLT	19.33	35.27	0.365
0.215	DCT	40.68	32.04	0.157
	MLT	33.05	32.94	0.161

- Pour l'image *Cornouaille*

Débit	Transformée	Distorsion	PSNR	Entropie
0.969	DCT	18.87	35.37	0.809
	MLT	14.28	36.58	0.828
0.438	DCT	23.99	34.33	0.406
	MLT	16.61	35.93	0.345
0.215	DCT	34.43	32.76	0.145
	MLT	24.23	34.29	0.148

Les tests tendent à confirmer les idées qui avaient été énoncées. La MLT serait supérieure à la DCT du point de vue de la QV :

- De manière objective, l'énergie est plus concentrée dans les composantes continues.
- De manière subjective, il y a moins d'effets de bloc.

6.1.3 Comparaison exhaustif/arborescent

Nous travaillons dans un espace MLT et en intra-bande.

- Pour l'image *Lena*

Débit	Recherche	Distorsion	PSNR	Entropie
0.969	exhaustive	18.26	35.51	0.861
	arborescente	24.14	34.30	0.812
0.438	exhaustive	19.61	35.21	0.355
	arborescente	25.75	34.00	0.332
0.215	exhaustive	29.99	33.36	0.155
	arborescente	35.44	32.64	0.143

- Pour l'image *Barbara*

Débit	Recherche	Distorsion	PSNR	Entropie
0.969	exhaustive	35.93	32.58	0.819
	arborescente	52.26	30.95	0.744
0.438	exhaustive	41.89	31.91	0.330
	arborescente	59.15	30.41	0.301
0.215	exhaustive	79.86	29.11	0.145
	arborescente	95.67	28.32	0.130

- Pour l'image *Interview*

Débit	Recherche	Distorsion	PSNR	Entropie
0.969	exhaustive	18.55	35.45	0.860
	arborescente	25.75	34.02	0.800
0.438	exhaustive	19.33	35.27	0.365
	arborescente	28.29	33.61	0.341
0.215	exhaustive	33.05	32.94	0.161
	arborescente	41.86	31.91	0.143

- Pour l'image *Cornouaille*

Débit	Recherche	Distorsion	PSNR	Entropie
0.969	exhaustive	14.28	36.58	0.828
	arborescente	19.61	35.21	0.755
0.438	exhaustive	16.61	35.93	0.345
	arborescente	24.76	34.19	0.305
0.215	exhaustive	24.23	34.29	0.148
	arborescente	33.15	32.93	0.129

Nous considérons, à présent, le temps de calcul CPU pour la réalisation des dictionnaires et des encodages, nous exprimons les résultats sous la forme de rapports entre temps de façon à être indépendant du calculateur utilisé, le débit est de 0.438 bpp.

Image	Construction du dictionnaire Temps exhaustif/arborescent	Encodage Temps exhaustif/arborescent
Lena	6.46	29.35
Barbara	6.48	30.40
Interview	7.28	34.97
Cornouaille	7.42	37.46

Comme nous nous y attendions la distorsion est plus élevée dans le cas d'un QV à recherche arborescente. Néanmoins cette hausse de distorsion apparait suffisamment peu élevée eu égard à la diminution de la charge en calculs qui a été réalisée lors de l'opération de codage. Par ailleurs l'entropie est inférieure dans le cas d'une recherche arborescente et la supériorité du QV exhaustif doit être modérée dans le cas d'un codage entropique des indexs transmis.

6.1.4 Importance du facteur psychovisuel

On se place évidemment dans le cas de la QV intra-bande puisque le but de la matrice de visualisation est de répartir la distorsion totale de l'image entre les sous-bandes fréquentielles afin de diminuer la "distorsion visuelle".
Nous avons effectué les tests qui suivent à partir de l'image *Interview*, en fixant une distorsion maximale apportée par la QV. Dans un premier cas, cette distorsion est équitablement répartie dans les sous-bandes, dans un deuxième cas, elle est répartie selon le poids des coefficients psychovisuels. Les valeurs de distorsion maximum ont été choisies afin d'obtenir des débits comparables dans les deux cas.
Deux valeurs de distorsion et de PSNR ont été calculées :

- La distorsion et le PSNR visuels, c'est à dire que l'on a multiplié la distorsion à l'intérieur de chaque sous-bande par le coefficient psychovisuel de cette sous-bande.
- La distorsion et le PSNR classiques.

Les résultats obtenus sont les suivants :

Débit	Psycho	Dist visuelle	PSNR visuel	Dist réelle	PSNR réel	Entropie
0.922	oui	60.46	28.79	45.29	31.57	0.726
0.922	non	65.04	27.32	46.36	31.47	0.716
0.766	oui	95.67	19.61	66.95	29.87	0.619
0.766	non	102.05	18.32	67.75	29.82	0.601

Du point de vue des résultas objectifs, il paraît difficile de tirer des conclusions au vu de ces résultats même si a priori, pour une même distorsion réelle, il semble que la distorsion visuelle soit moins importante dans le cas où l'on utilise la matrice de coefficients psychovisuels. D'un point de vue visuel, il paraît aussi difficile à ce niveau de distorsion de pouvoir statuer sur l'intérêt d'introduire le

facteur psychovisuel.
Ces résultats mitigés peuvent s'expliquer par le fait que la matrice de coefficients n'est pas adaptée à notre schéma de quantification.

6.2 Obtention d'un dictionnaire global

Comme nous l'avons précédemment exprimé, un dictionnaire idéal serait un dictionnaire pouvant servir à compresser et décompresser n'importe quelle image. Dans la première série de tests, nous nous sommes contentés de construire des dictionnaires à partir d'une certaine image puis de tester notre QV sur cette même image. Le but de cette deuxième série de tests est d'évaluer les performances d'un dictionnaire par rapport à un nombre plus important d'images. On distinguera deux types d'expériences :

- Le dictionnaire a été construit à partir de plusieurs images et ses performances sont évaluées pour chacune de ces images.
- On a essayé de construire un dictionnaire global et celui-ci est testé par rapport à des images n'ayant pas fait partie de la séquence d'apprentissage.

6.2.1 Dictionnaire construit à partir de plusieurs images

Le but de cette série de tests est d'évaluer les performances d'un dictionnaire construit à partir de plusieurs images. Ce test vise à déterminer si dans le cas d'une base de données statique, il est possible de construire un dictionnaire unique pour coder l'ensemble des images de la base ou du moins plusieurs images de cette base.
Pour ce faire, nous avons construit un dictionnaire à partir des 4 images énumérées auparavant et avons testé les performances de ce dictionnaire par rapport aux 4 images. En parallèle, nous avons construit un dictionnaire ayant exactement la même configuration mais uniquement à partir de l'image *Lena*. Nous avons pû ainsi comparer l'efficacité de ces deux dictionnaires par rapport à la base de données. Ces dictionnaires ont été construits dans l'espace DCT de manière arborescente. La configuration des dictionnaires est donnée par la Figure 12.
Les résultats obtenus sont les suivants :

Dictionnaire construit à partir des 4 images

Image	Distorsion	PSNR	Entropie
Lena	44.25	31.67	0.754
Barbara	75.43	29.35	0.799
Cornouaille	32.03	33.07	0.698
Interview	40.21	32.08	0.755

Dictionnaire construit à partir de *Lena*

Image	Distorsion	PSNR	Entropie
Lena	25.69	34.03	0.831
Barbara	157.90	26.15	0.833
Cornouaille	59.37	30.39	0.739
Interview	74.61	29.40	0.772

La distorsion commençant à devenir plus importante, il est possible de constater visuellement la distorsion engendrée par le codage de l'image *Barbara* à l'aide du dictionnaire construit à partir de *Lena* (Figure 13) alors que la distorsion engendrée par le codage de la même image à l'aide du dictionnaire construit sur les 4 images n'est pas encore perceptible visuellement (Figure 14).

Débit = 1

128 (scalaire)	512 (2*2)	256 (4*4)	256 (4*4)
512 (2*2)	256 (4*4)	256 (4*4)	256 (4*4)
256 (4*4)	256 (4*4)	0	0
256 (4*4)	256 (4*4)	0	0

Figure 12 : configuration du dictionnaire

Figure 13 : Dictionnaire construit sur *Lena*

Figure 14 : Dictionnaire construit sur les 4 images

On peut constater que les distorsions engendrées pour chaque image en les codant par le dictionnaire construit à partir de ces images sont tout à fait acceptables pour la plupart des applications. On a donc à peu près résolu le problème des bases de données statiques même s'il faut être conscient que plus le nombre d'images est important, moins bonne sera la reconstruction de ces images à partir d'un dictionnaire construit sur l'ensemble des images. Il faudra donc envisager de construire plusieurs dictionnaires si la base de données est importante. Dans ce cas, il sera possible de classer les images afin de construire chaque dictionnaire sur un même type d'images et améliorer ainsi les performances.

6.3 Elaguage

Les tests de notre algorithme d'élaguage ont été effectués à partir de l'image *Lena*. Dans un premier temps, nous avons construit des dictionnaires contenant deux fois plus de représentants que les dictionnaires que l'on désire obtenir après élaguage. Il faut bien prendre conscience que les résultats seraient améliorés si l'on partait d'un dictionnaire plus grand avant l'élaguage.
Ces tests d'élaguage ont été réalisés sur une matrice MLT, avec une configuration intra-bande.

Débit	**Transformée**	**Distorsion**	**PSNR**	**Entropie**
0.969	élaguage	18.65	35.42	0.820
	arborescent	23.35	34.39	0.812
0.438	élaguage	20.40	35.03	0.345
	arborescent	25.37	34.08	0.333

Les résultats obtenus montrent une assez nette baisse de distorsion pour un même débit par rapport à un dictionnaire ayant une structure arborescente équilibrée.
De plus, la méthode d'élaguage semble offrir des perspectives très intéressantes pour la construction d'un dictionnaire "global". Il semble en effet possible de construire un dictionnaire de très grande taille à partir d'un grand nombre d'images puis d'élaguer cet arbre pour chaque image que l'on veut coder. On pourrait ainsi, à partir d'un dictionnaire très vaste, extraire la partie de l'arbre spécifique à chaque image que l'on code. Cette technique pourrait être utilisée autant dans un but de transmission que dans un but d'archivage de données.

7 Conclusion

Dans le cadre de cette étude, nous avons testé différentes méthodes de QV appliquée au codage d'images fixes.
Un des premiers résultats a montré qu'il n'était pas envisageable d'effectuer une recherche exhaustive dans un dictionnaire ne comportant aucune structure géométrique ou topologique particulière. Nos recherches se sont donc vite orientées vers des QV arborescents entraînant une charge de calculs au codage nettement moins importante. Toutefois, dans le cas d'un débit constant (arborescence équilibrée), il s'est avéré que beaucoup d'éléments du dictionnaire étaient peu représentatifs et nous avons donc décidé de construire, par élaguage, un QV arborescent à structure variable.
Pour regrouper l'énergie de l'image sur quelques coefficients et afin d'allouer le débit dans les zones importantes de l'image, il semble intéressant de travailler dans un espace transformé. Dans ce cadre de transformations fréquentielles, nous avons comparé l'approche intra-bande et celle inter-bande. Il s'avère que pour vectoriser et quantifier au mieux les coefficients transformés, l'approche intra-bande est plus efficace. Nous étant placés dans ces conditions, nous avons implémenté un QV à structure variable. La méthode utilisée pour arriver à ce QV a été l'élaguage. Cette méthode nécessite la construction d'un arbre de taille importante dans un premier temps mais ceci n'est pas pénalisant, la construction du dictionnaire n'entrant pas dans le processus de quantification. Le processus d'élaguage a fourni des résultats très intéressants et est retenu comme étant la méthode assurant le meilleur rapport qualité/complexité parmi tous les QV testés.
Pour chacune des images testées, il a été possible d'obtenir une bonne qualité de restitution à de faibles débits. Néanmoins, un problème auquel nous nous sommes heurtés a été l'élaboration d'un dictionnaire universel pouvant servir à coder toute image. Ce problème est jusqu'à aujourd'hui resté sans réponse et devrait susciter encore des recherches. Toutefois, une démarche pouvant sembler intéressante à explorer serait la construction d'un dictionnaire arborescent de très grande taille construit à partir de plusieurs images de types différents. Ce dictionnaire étant ensuite élagué relativement à l'image à coder (c'est à dire que l'on sélectionnerait la partie de l'arbre pouvant coder le type de l'image en entrée).
Dans ce rapport, les résultats des tests ont été présentés pour la plupart sous forme de tableaux ; il eut été sans doute plus juste de montrer des résultats visuels (images résultats) mais la qualité de reconstruction était souvent trop bonne pour que la différence avec l'image originale soit évidente. Il ne faut pas pour autant oublier que le but de la QV appliquée au codage d'images est de pouvoir restituer un signal visuellement de bonne qualité.
Il convient de modérer notre choix sur un QV à structure arborescente dans le cas d'une architecture multi-processeurs. En effet, si l'on peut associer un processeur à chaque élément du dictionnaire, le codage pour chaque bloc de l'image se fait en calculant en parallèle la distorsion engendrée par chaque représentant du dictionnaire. Dans ce cas, il est avantageux d'utiliser une structure exhaustive car une structure arborescente ne permet pas ce parallèlisme.

Annexe A

Pondération psychovisuelle

Elle est basée sur une mesure de distorsion de type EQM pondérée.
Dans le cas où la transformation est unitaire, on rappelle que la distorsion des bandes est additive :

- $EQM = \sum_i D_i$
- $EQM_{\text{pondérée}} = \sum_i \omega_i D_i$ où ω_i est le coefficient psychovisuel associé à la sous-bande i.

Sous certaines conditions de modélisation des caractéristiques débit-distorsion du quantificateur utilisé et des densités de probabilité des sources considérées (ici les sous-bandes), l'allocation en distorsion optimale est du type :

$$D_i^* = \frac{D}{\omega_i} \text{ au facteur de normalisation près}$$

D étant la distorsion totale allouée.
Nous avons adopté ce type de pondération bien qu'elle n'ait pas été optimisée pour le schéma de quantification utilisé. Le calcul des poids ω_i est réalisé selon la procédure décrite par B. Macq [16] avec une perception visuelle de type passe-bande associée à la fonction de contraste déterminée expérimentalement par Mannos et Sakrison [17].
On donne la matrice qui a été employée dans notre application:

$$\omega = \begin{bmatrix} 32.81 & 31.07 & 23.61 & 14.18 \\ 32.06 & 26.57 & 19.85 & 12.07 \\ 25.84 & 10.96 & 15.76 & 9.81 \\ 16.87 & 13.83 & 10.65 & 6.88 \end{bmatrix}$$

Rappel des notions utilisées

distorsion (D) :
Concernant le traitement d'images, la distance euclidienne est la plus répandue des mesures de distorsion.

$$d(x, y) = \sum_{j=1}^{k} (x_j - y_j)^2$$

La distorsion correspond à l'erreur quadratique moyenne qui a été introduite lors des opérations de codage-décodage.

En traitement d'images, il est d'usage d'associer à cette distorsion le rapport signal sur distorsion PSNR défini par :

$$PSNR = 10 log_{10} \left[\frac{255^2}{D} \right] \text{ (dB)}$$

L'entropie qui est calculée est fonction de la distribution des représentants du dictionnaire à l'intérieur de l'image codée. L'entropie correspond au débit qui pourrait être atteint si postérieurement au QV, on mettait en place un codage de Huffman ou codage entropique [18].
Soit p la fonction de densité de probabilité de l'image codée, $p(a) = p(X = a)$ où a est un élément du dictionnaire A et X un vecteur de l'image codée. Soit $l(a)$ la longueur du code du représentant a, alors :

$$entropie = \sum_{a \in A} p(a) l(a)$$

Annexe B

Images utilisées pour les tests

Figure 15 : *Lena*

Figure 16 : *Barbara*

Figure 17 : *Interview*

Figure 18 : *Cornouaille*

Bibliographie

[1] C.E. Shannon, "A mathematical theory of communication", *Bell Systems Technical Journal 27*, pp. 379-423, pp. 623-656, 1948.

[2] C.E. Shannon, "Coding theorems for a discrete source with a fidelity criterion", *IRE National Convention Record*, Part 4, pp. 142-163, 1959.

[3] Y. Linde, A. Buzo, R.M. Gray, "An algorithm for vector quantizer design", *IEEE Trans. on Communic.*, Vol. COM-28, No. 1, pp. 84-95, Jan. 1980.

[4] S.P. Lloyd, "Least squares quantization in PCM", *IEEE Trans. Inform. Theory*, Vol. IT-28, No. 2, pp. 129-137, March 1982.

[5] A. Benazza, *Quantification vectorielle en codage d'images*, Thèse de l'Université de Paris-Sud, centre d'Orsay, p. 36.

[6] A. Benazza, *Quantification vectorielle en codage d'images*, Thèse de l'Université de Paris-Sud, centre d'Orsay, pp. 52-63.

[7] E.A. Riskin, *Variable rate vector quantization of images*, PhD thesis, Stanford University, Stanford, Ca., June 1988.

[8] E.A. Riskin, "A greedy tree growing algorithm for the design of variable rate vector quantizers", *IEEE Trans. Signal Process.*, Vol. 39, No. 11, pp. 2500-2507, Nov. 1991

[9] L. Breiman, J.H. Friedman, R.A. Olshen, C.J.Stone, *Classification and regression trees*, The Wadsworth Statistics/Probability Series, Belmont, California : Wadsworth, 1984.

[10] A. Gersho, R.M. Gray, *Vector quantization and signal compression*, Kluwer academic publishers, pp. 645-652, 1992.

[11] K.R. Rao, P. Yip, *Discrete Cosinus Transform : Algorithms, Advantages, Applications*, Academic Press, Chap. 2, p. 7, 1990.

[12] E. Nguyen Phuc, *Bases Trigonométriques Locales. Application au Codage d'Images*, DEA, Institut National Polytechnique de Toulouse, Sept. 92.

[13] A. Benazza, *Quantification vectorielle en codage d'images*, Thèse de l'Université de Paris-Sud, centre d'Orsay, pp. 57-60.

[14] P.A. Chou, T. Lookabaugh, R.M. Gray, "Optimal pruning with applications to tree-structured source coding and modeling", *IEEE Trans. Inform. Theory*, pp. 299-315, March 1989.

[15] H.S. Malvar, *Signal Processing with Lapped Transforms*, Universidade de Brasilia, Brazil, Artech House, Boston, London, 1992.

[16] B. Macq, *Perceptual Transforms and Universal Entropy Coding for an Integrated Approach to Picture Coding*, Thèse de l'Université Catholique de Louvain, Spt. 1989.

[17] J. Mannos, D. Sakrison, "The effect of a visual fidelity criterion on the encoding of images", *IEEE Trans. Inform. Theory*, July 1974.

[18] A. Gersho, R.M. Gray, *Vector quantization and signal compression*, Kluwer academic publishers, pp. 271-277, 1992.

[19] H.S. Malvar, "Lapped transform for efficient transform/subband coding", *IEEE Transactions on acoustics, Speech and Signal Processing*, Vol. 38, No. 6, Juin 1990.

[20] J.P. Marescq, *Etude de schémas de quantification vectorielle adaptative multiclasses. Application au codage de séquences d'images télévisuelles*, Thèse de l'Université de Rennes1, 1986.

[21] M. Antonini, *Transformées en ondelettes et Compression numériques des images*, Thèse de l'Université de Nice Sophia-Antipolis, 1991.

[22] P. Monet, C.labit, " Codebook Replenishment in Classified Pruned Tree-Structured Vector Quantization of Image Sequences", *ICASSP*, pp. 2285-2288, 1990.

Unité de recherche INRIA Lorraine, Technôpole de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399