

JPEG2000 Codeur/Décodeur Arithmétique

Note d'Application ver 0.1

Mouhcine Chami / Telecom Paris

Crée le : 11/04/2003

Modifié le : 16/04/2003

1- Introduction

Dans cette note d'application, on explique le codeur arithmétique et son utilisation pratique dans le JPEG2000. Les principales caractéristiques du JPEG2000 sont :

- Meilleure qualité pour un débit de compression plus élevé
- décodage progressive
- Possibilité de compression avec peu ou sans perte de qualité.

Pour les images fixes ou les films, on suit l'ordre des procédures suivantes :

- 1- Transformation en ondelettes
- 2- Quantification
- 3- Réorganisation des coefficients
- 4- Codeur arithmétique
- 5- Organisation du flux de code

Pour le décodage, on suit vraiment le sens inverse de cet ordre.

2- Vision Globale sur la théorie

2.1 Codeur arithmétique

Le codage arithmétique transmet seulement l'information requise pour permettre à un décodeur de déterminer l'intervalle partiel particulier entre 0 et 1. Cette information inclut un mot de code partiel **C** qui pointe sur la valeur minimale de l'intervalle et la largeur d'intervalle **A**. La longueur du nouveau intervalle (après mise à jour) est la probabilité associée à la séquence des symboles déjà effectués. La position de l'intervalle partiel est aussi importante que la longueur.

2.2 Codage arithmétique binaire

Dans le codage arithmétique binaire (**BAC**), les symboles dans un jet de code sont classifiés en MPS "most probability symbol" et LPS "least probability symbol". L'intervalle **A** a deux subdivisions, LPS et MPS. la largeur de chaque division est déterminée par la probabilité de chaque symbole. l'intervalle associé LPS doit être petit que celui associé à MPS.

Notation : Q_e = Probabilité relatif à LPS

2.2.1 Le codage BAC

Le codage en utilisant le BAC peut être résumé comme suit :

1. Initialiser $A_0 = 1$ et $C_0 = 0$
2. Déterminer la valeur de l'évènement (MPS ou LPS), Q_e
3. Mise à jour du mot de code et/ou l'intervalle dépendant de la valeur de l'évènement

si c'est un MPS :

a) $C = C + (Q_e \times A)$

b) $A = A - (Q_e \times A)$

sinon "LPS"

a) $A = Q_e \times A$

2.2.2 Le décodage BAC

On recevant le mot de code C, il suffit simplement d'inverser l'opération de codage en calculant les mêmes intervalles et utilisant les mêmes distribution de probabilité. Quand la borne inférieure de l'intervalle est égal à C, ça veut dire qu'on a décodé le dernier symbol.

2.3 Codeur Arithmétique JPEG2000

Le standard JPEG2000 utilise le codage arithmétique binaire (BAC) qui a été adapté au prétendu Q-codeur. Il traite la mise en pratique en détournant les problèmes qui surgissent à la mise en application.

- Enlever la multiplication requises pour chaque symbole codage/décodage
- Traitement l'échange conditionnel du sens de MPS, qui surgit quand l'intervalle résultant pour les LPS excède celui du MPS
- Utilisation d'un codeur arithmétique adaptatif
- Traiter avec une précision finie
- Les problèmes associé à la longueur croissante du mot de code et la propagation du reste.

La convention du codage : les opérations du codage sont faites en utilisant l'arithmétique des nombres entier de précision finie et employer une représentation des nombres entier des valeurs partielles dans lesquelles $0x8000$ est équivalent à la décimale 0.75 . La longueur de l'intervalle A est maintenu dans la gamme $0.75 < A < 1.5$ en le doublant à chaque fois que sa valeur devient inférieure à $0x8000$.

Enlever la multiplication : Le JPEG2000 BAC maintiens la largeur A dans l'intervalle $[0.75 \ 1.5)$ par un process de normalisation. Sa moyenne est approximativement proche de 1, donc $(Q_e \times A) = Q_e$. cette approximation peut être appliqué dans l'algorithme utilisé ce qui va simplifier l'implémentation du codeur et du décodeur.

L'algorithme devient :

Si c'est un MPS

a- MAJ du mot de code $C = C + Q_e$

b- MAJ de la longueur de l'intervalle $A = A - Q_e$

sinon "LPS"

a- Le mot de code ne change pas $C = C$

b- MAJ de la longueur de l'intervalle $A = Q_e$

Traitement d'échange conditionnel du sens de MPS : Si on continue de recevoir MPS symboles, la largeur A de l'intervalle va se rétrécire ($A = A - Q_e$) ce qui va la rendre plus petite que la valeur de LPS. A ce niveau là, le sens de MPS doit être inverser. Même chose pour LPS.

Le BAC adaptative : Encore un avantage du JPEG2000, la probabilité associé à LPS et MPS peut être adaptée. Pour cela, il est nécessaire d'avoir un modèle statistique des symboles de données d'entrée qui va permettre de mettre à jour les probabilités associées au MPS et LPS. Le modèle doit également déterminer si le prochain

évènement s'est un MPS ou un LPS.

JPEG2000 a 19 contextes possible. A partir de l'entrée CX du codeur et du décodeur, il permet de définir les différentes valeur de Qe et selon si le prochain est un MPS ou un LPS de définir l'indexation du prochain Qe à prendre en compte. voir le tableau...

Index	Qe Value		NMPS	NLPS	
SWITCH					
	Hexadecimal	Binary	Decimal		
29	\$1101	0001 0001 0000 0001	0.099 634	30	27 0
30	\$0AC1	0000 1010 1100 0001	0.063 012	31	28 0
31	\$09C1	0000 1001 1100 0001	0.057 153	32	29 0
32	\$08A1	0000 1000 1010 0001	0.050 561	33	30 0
33	\$0521	0000 0101 0010 0001	0.030 053	34	31 0

Tableau 1 : Une portion de la Table C.2 de la norme JPEG 2000

Supposons dans un exemple que notre modèle est à l'état 31 (index 31), $Q_e = 0.057153$. Si un LPS arrive, le modèle passe à l'état 29 avec un $Q_e = 0.099634$, lequel est plus grand que 0.057153 . Une normalisation suit. Si un MPS arrive, le modèle passe ou ne passe pas. Ca dépend de la valeur de A (longueur de l'intervalle). Si $A < 0.75$, le modèle passe à l'état 32 avec un $Q_e = 0.050561$. Une normalisation suit. Dans d'autre cas ($A > 0.75$) il n'y a pas de changement d'index et donc pas de normalisation.

Utilisation de la précision finie : Pour maintenir un emplacement de la virgule fixe unifié des valeurs décimal A et C, C est décalé à gauche à chaque normalisation. Ce décalage a besoin d'un registre à longueur infinie. Pour résoudre ce problème, un buffer de data prend les bits décalés de C, avec un compteur qui permet de dire que ce buffer est plein et qu'on peut l'envoyer à la sortie.

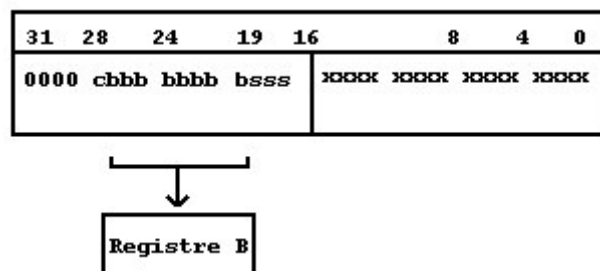


Figure 1 : Les partitions du registre C et buffer de sortie

3. Codage JPEG2000



Figure 2 : Bloc du codeur

On initialise le tableau des contextes propre au codeur, il contient l'emplacement de 19 contextes différents pointant sur la table C-2 de la norme. Ce tableau contient une liste d'indice I et de MPS correspondant (0 ou 1). La phase d'initialisation permet de remplir l'indice du contexte de démarrage et un MPS par défaut.

La procedure de codage devient :

- Initialisation des compteurs géant le flux de sortie (CT = 8)
- Lire D, CX
- Pointer sur le tableau contexte(CX), CX représente l'indice dans la table propre du codeur
- A partir de l'indice, on lit I et MPS
- Si (D = MPS)
 - $A = A - Qe(I)$ // on cherche la valeur de Qe dans la table C-2 de la norme
 - $C = C + Qe(I)$
 - $I = NMPS(I)$ // on cherche la valeur de NMPS dans la table C-2 de la norme
- Sinon (D =LPS)
 - $A = Qe(I)$ // on cherche la valeur de Qe dans la table C-2 de la norme
 - $I = NLPS(I)$ // on cherche la valeur de NLPS dans la table C-2 de la norme
- Si Switch(I) = 1 $MPS = 1 - MPS$ // on cherche la valeur de Switch dans la table C-2 de la norme
- Normalisation (Pour maintenir $A \leq 0x8000$)
 - { $A \ll 1$
 - $C \ll 1$
 - $CT = CT - 1$ // Compteur des bits en sortie si (CT = 0) alors un octet est prêt
 - }

Mise à jour du tableau des contextes

4. Décodage JPEG2000

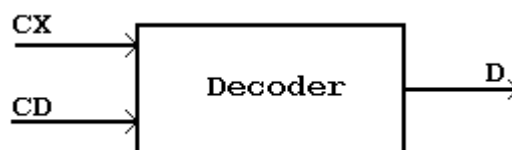


Figure 2 : Bloc du décodeur

La structure des registres A et C comme elle a été définie dans la norme JPEG 2000.

	MSB	LSB
Chigh register	xxxx xxxx	xxxx xxxx
Clow register	bbbb bbbb	0000 0000
A register	aaaa aaaa	aaaa aaaa

Figure 1 : Les partitions du registre C, A et buffer d'entrée

Le registre 32 bits de C est divisé en deux parties Chigh utilisé pour le mot de code et le Clow représente le buffer d'entrée. Au début du décodage, on remplit le Chigh et le Clow et c'est la procédure de normalisation qui gère le décalage des nouveaux bits du mot de code.

On utilise dans la comparaison montré dans les différentes procédures de la norme une précision de 16 bits.

On initialise le tableau des contextes propre au décodeur, il contient l'emplacement de 19 contextes différents pointant sur la table C-2 de la norme. Ce tableau contient une liste d'indices I et de MPS correspondant (0 ou 1). La phase d'initialisation permet de remplir l'indice du contexte de démarrage et un MPS par défaut (on utilise la table D-7).

La procédure de décodage devient :

- Initialisation des compteurs gérant le flux d'entrée et de sortie.
- Lire CX
- Pointer sur le tableau contexte(CX), CX représente l'indice dans la table propre du codeur
- A partir de l'indice, on lit I et MPS
- Une comparaison du mot de code Chigh avec le Qe du contexte permet de décodifier la décision.
- Normalisation (Pour maintenir $A \leq 0x8000$)


```

      {
        A << 1
        C << 1
        CT = CT - 1 // Compteur des bits en sortie si (CT = 0) alors un octet
      }
      
```
- Mise à jour du tableau des contextes

5- Bibliographie :

- [1] : JPEG 2000 Image coding system, ISO/IEC JTC 29/WG 1 N2678, project 1.29.15444, revised 19 July 2002)
- [2] : JPEG 2000 Arithmetic Encoding on the StarCore SC140, Application Note by Sue Twelves and Mike Wu, Rev 1.0, 10/2001, MOTOROLA
- [3] : page web : <http://www.datacompression.info/JPEG2000.shtml>
- [4] : page web : <http://dogma.net/markn/articles/arith/part1.htm>