Université Badji-Mokhtar. Annaba
Faculté de Technologie
Département d'Informatique

Chapitre 4

Compression image JPEG

S3- 2025/2026

Introduction

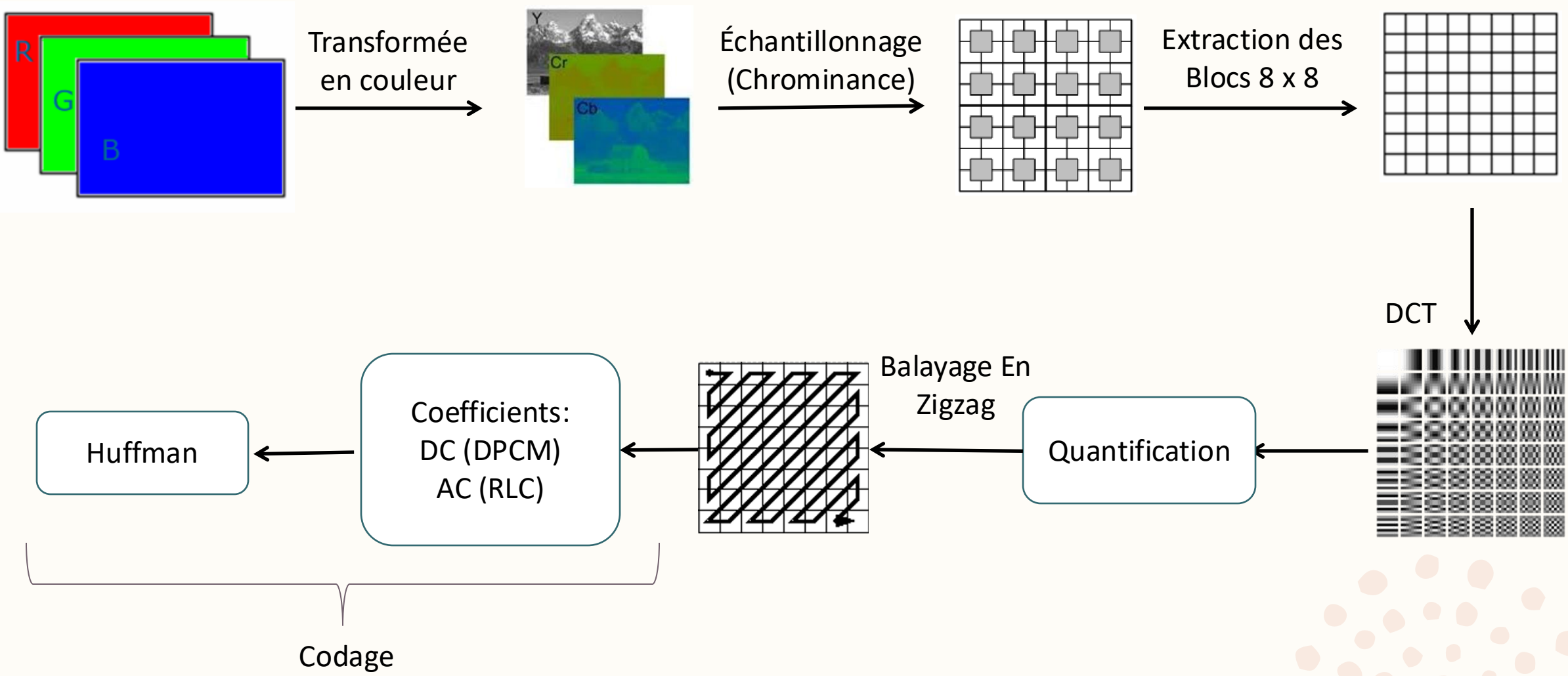
- **La méthode de compression avec pertes:** élimine certaines données pour réduire la taille total des données. Cela peut entrainer une dégradation de la qualité.
- Les objectifs de la compression avec pertes sont d'éliminer les données non pertinentes.
- Utilisé dans le domaine des fichiers multimédias: images, audio, vidéo.
- **Exemples**: JPEG, MPEG, MP3.

Compression JPEG

- **JPEG** est une norme de compression d'image qui a été développée par le « **Joint Photographic Experts Group** » en 1992.
- JPEG est une méthode de compression d'image avec perte.
- Cela permet d'obtenir des fichiers de taille nettement plus petite avec peu ou pas d'impact perceptible sur la qualité et la résolution de l'image.
- JPEG fonctionne en supprimant les informations qui ne sont pas facilement visibles à l'œil humain tout en conservant les informations que l'œil humain est capable de percevoir.

Compression JPEG

- Les étapes de compression JPEG



Compression JPEG

1. **La transformée en couleur :**

- ✓ La vision humaine: manque de sensibilité couleurs et forte sensibilité aux variations des intensités lumineuses.
- ✓ JPEG modifie l'espace colorimétrique de l'image de RGB à YCbCr.
- ✓ Le mode YCrCb sépare la luminosité du pixel
  - **Y** représente la luminance qui décrit le degré de clarté ou d'obscurité d'un pixel, capturant essentiellement la version en niveaux de gris d'une image.
  - **Cb** représente la composante de chrominance liée à la différence de couleur bleue.
  - **Cr** représente la composante de chrominance liée à la différence de couleur rouge.

Compression JPEG

1. **La transformée en couleur :**

✓ Changement d'espace colorimétrique

- **RGB -> YCrCb**

Luminance : Y = 0.299 R + 0.587 G + 0.114 B

Chrominance : Cb = - 0.1687 R - 0.3313 G + 0.5 B

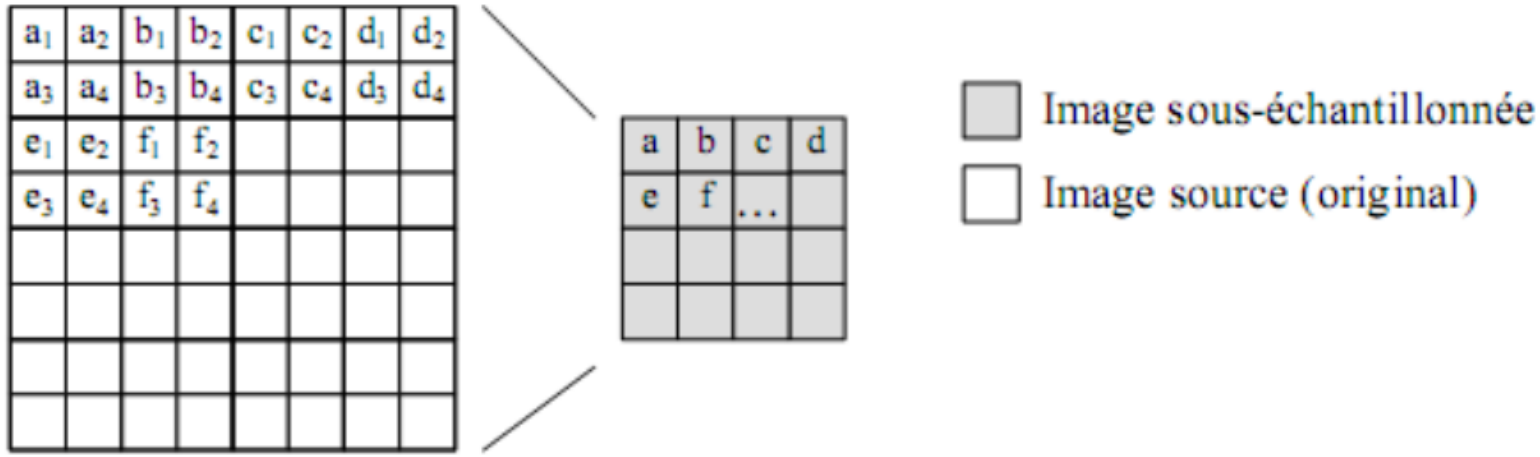
Cr = 0.5 R - 0.4187 G - 0.0813 B

- Images Chrominance : **moins d'information -> fort taux de compression**

Compression JPEG

2. Sous échantillonnage:

- ✓ L'œil ne distingue pas bien la différence de couleurs entre pixels voisins, donc :
- ✓ Coder la luminance de chaque pixel
- ✓ Sous échantillonnage de la chrominance (consiste à réduire la taille d'une image ou bien suppression de pixels)

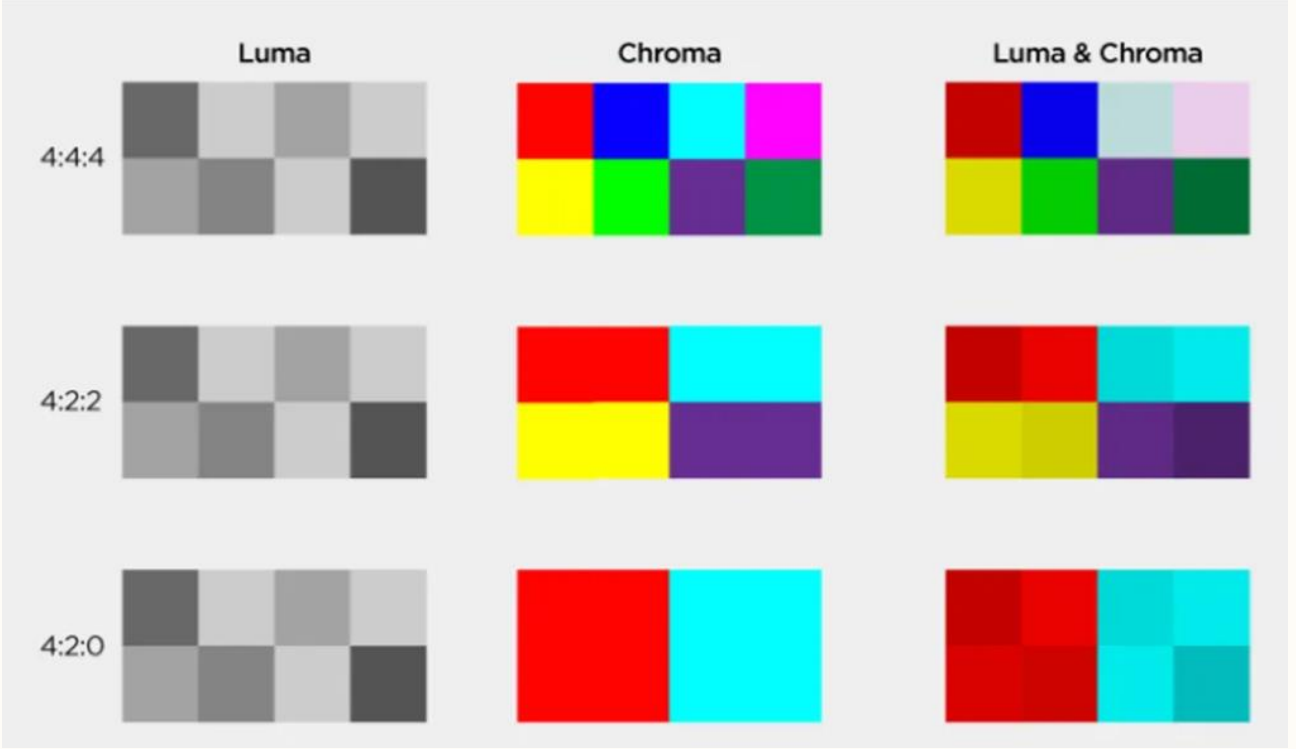


Compression JPEG

2. Sous échantillonnage:

Les schémas de sous-échantillonnage sont :

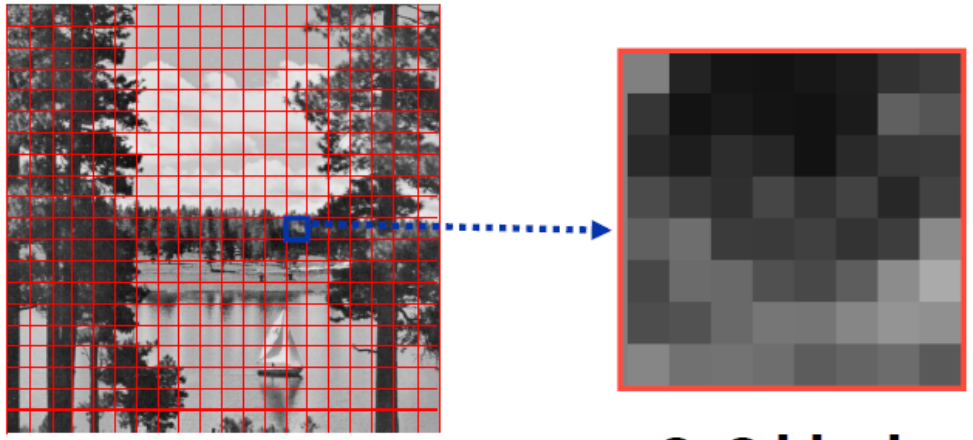
- ✓ 4:4:4: pas de sous-échantillonnage
- ✓ 4:2:2: sous-échantillonnage chroma horizontal
- ✓ 4:2:0: sous-échantillonnage chroma horizontal et vertical



Compression JPEG

3. Extraction des Blocs 8 x 8:

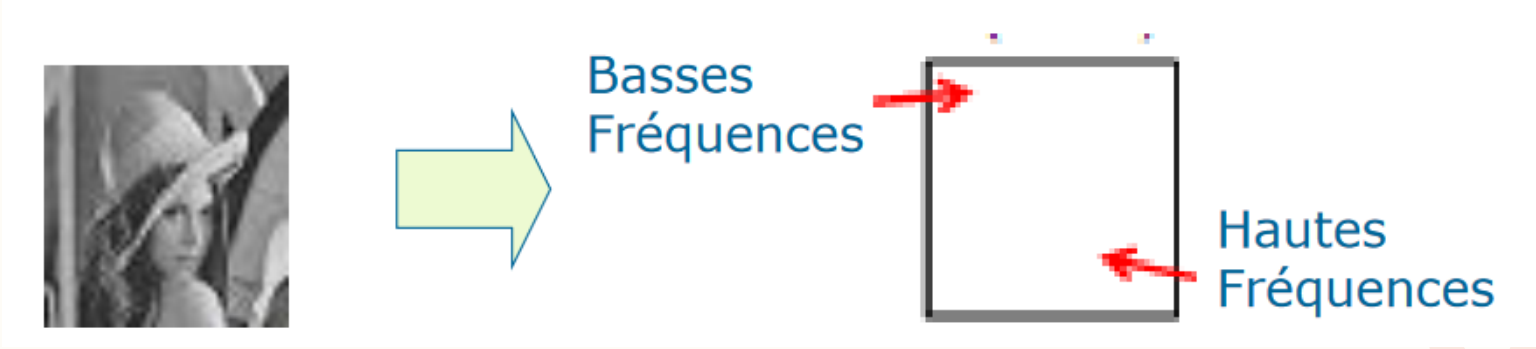
- ✓ JPEG divise l'image en blocs de 8x8 pixels.
- ✓ Pour chaque bloc de 8x8, les données de l'image (pour les composants Y, Cb et Cr) sont traitées séparément.



Compression JPEG

4. La transformation DCT:

- ✓ Pour chaque bloc de 8x8 pixels, JPEG applique une opération mathématique appelée transformée en cosinus discrète (DCT).
- ✓ La DCT convertit les informations du domaine spatial (valeurs des pixels) en informations du domaine fréquentiel.
  - ✓ **Les coefficients basse fréquence** représentent les informations générales de l'image (changements progressifs et fluides de couleur et de luminosité).
  - ✓ **Les coefficients haute fréquence** représentent les détails fins, les contours et le bruit.



Compression JPEG

4. La transformation DCT:

✓ La transformation calcule les coefficients F(u,v) par :

$$F(u,v) = \frac{2}{N} c(u)c(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} Img(x,y) \cos\left[\frac{\pi}{N} u\left(x+\frac{1}{2}\right)\right] \cos\left[\frac{\pi}{N} v\left(y+\frac{1}{2}\right)\right]$$

- Img(x,y): valeur de pixel à la position (x,y)
- N : taille du bloc (généralement N=8 pour JPEG).
- u , v: indices de fréquence (0 à N−1).
- c(u), c(v) facteurs d'échelle:

$$ou \begin{cases} c(0) = (2)^{-1/2} \\ c(w) = 1 \; pour \; w = 1,2, \ldots N-1 \end{cases}$$

Compression JPEG

4. La transformation DCT:

- ✓ **Exemple:**

- ✓ On considère la Matrice x des valeurs img(x,y) des pixels d'un bloc

Img (x,y) =

212	220	216	216	220	216	220	220
208	216	216	220	220	220	216	220
172	204	212	216	220	216	220	220
144	156	180	208	216	216	216	216
136	140	144	164	196	212	216	216
132	136	140	144	160	184	208	216
132	132	136	136	140	148	168	192
136	136	132	136	140	144	148	160

Compression JPEG

4. La transformation DCT:

- ✓ **Exemple:**

- ✓ On obtient après transformation la matrice DCT:

F(u,v) =

1471	-90	-1	-7	0	-5	-3	0
160	21	-17	-1	-3	-1	-2	0
-16	38	13	-4	1	-1	0	0
-3	-15	15	6	-1	1	1	1
-6	-3	-9	8	3	0	0	0
-3	-1	-2	-8	2	2	0	1
4	-1	1	-1	-5	-1	-2	1
-2	1	2	1	-1	-3	-1	1

Compression JPEG

5. La quantification:

- ✓ Représente la phase non conservatrice de compression JPEG
- ✓ Elle réduit le nombre de bits nécessaires (JPEG supprime certaines de ces informations haute fréquence qui sont moins perceptibles à l'œil humain sans affecter la qualité perçue de l'image).
- ✓ Les matrices de quantification doivent être connues du décodeur, donc transmises avec l’image.
- ✓ Des matrices ont été proposées par les membres du comité JPEG
- ✓ Chaque valeur de la matrice DCT est divisée par un nombre, puis arrondie à l’entier le plus proche.

Compression JPEG

5. La quantification:

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

la table de quantification de luminance

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

La table de quantification de chrominance

Compression JPEG

5. La quantification:

✓ **Exemple:**

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

$Q(u,v)$

$$\widehat{F}(u,v) = round\left(\frac{F(u,v)}{Q(u,v)}\right)$$

1471	-90	-1	-7	0	-5	-3	0
160	21	-17	-1	-3	-1	-2	0
-16	38	13	-4	1	-1	0	0
-3	-15	15	6	-1	1	1	1
-6	-3	-9	8	3	0	0	0
-3	-1	-2	-8	2	2	0	1
4	-1	1	-1	-5	-1	-2	1
-2	1	2	1	-1	-3	-1	1

$\widehat{F}(u,v))$

92	-8	0	0	0	0	0	0
13	2	-1	0	0	0	0	0
-1	3	1	0	0	0	0	0
0	-1	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

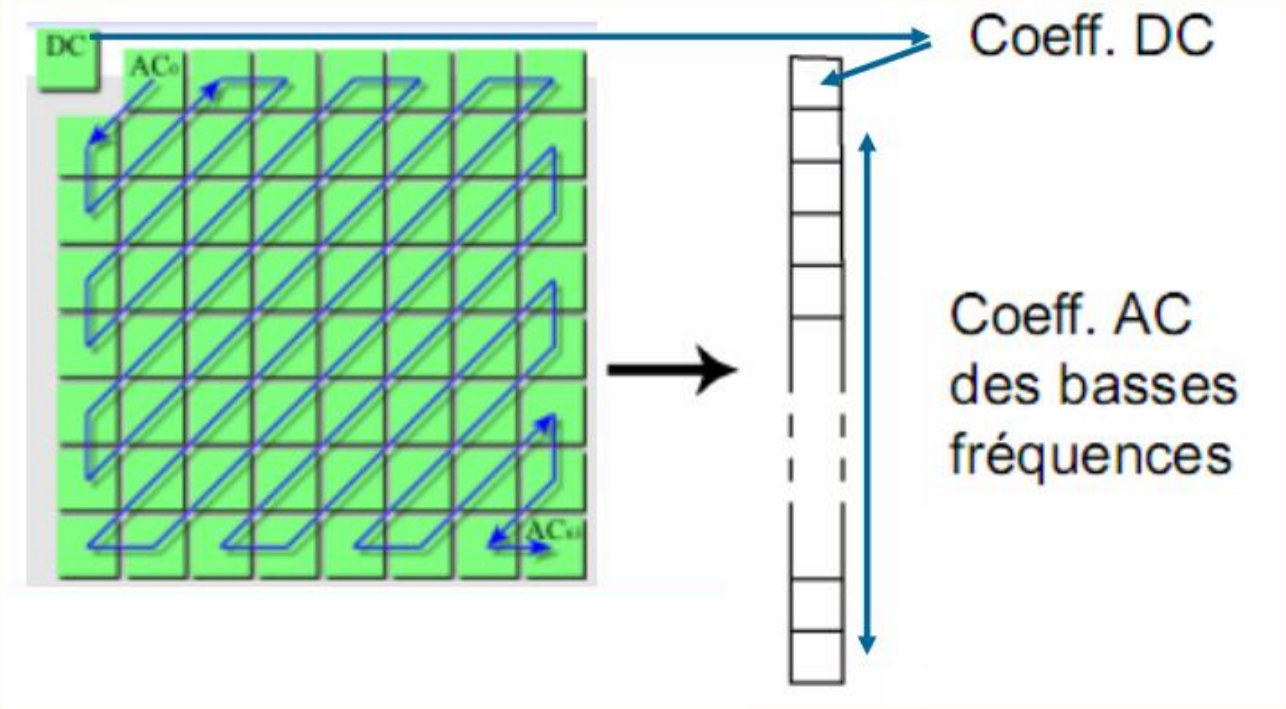
$F(u,v)$

Compression JPEG

6. Codage :

✓ Balayage en «zig-zag» (d'un bloc):

Former un vecteur où les coefficients relatifs aux basses fréquences sont regroupés



Compression JPEG

6. Codage :

✓ **Differential Pulse Code Modulation (DPCM):**

- DPCM est utilisé pour encoder **le coefficient DC (Direct Current)**, qui est la première valeur dans l'ordre en zigzag du bloc DCT quantifié.
- Le coefficient DC représente la luminosité moyenne du bloc.
- Les blocs voisins d'une image ont souvent des niveaux de luminosité similaires, de sorte que la différence entre les coefficients DC consécutifs (delta) est généralement faible.
- Le codage de cette différence (au lieu de la valeur DC réelle) réduit le nombre de bits requis.

$$\Delta DC = DC_{\text{actuel}} - DC_{\text{précédent}}$$

Compression JPEG

6. Codage :

- ✓ **Differential Pulse Code Modulation (DPCM):**

- **Exemple:**

- Supposons que les coefficients DC pour trois blocs soient : 120, 123, 124.

- Différences (ΔDC) : ΔDC=[120−0, 123−120, 124−123]=[120, 3, 1]

Compression JPEG

6. Codage :

✓ **Differential Pulse Code Modulation (DPCM):**

- **Table Huffman:**

- ΔDC est catégorisé en fonction de la plage de ses valeurs.
- Chaque catégorie est codée à l'aide d'un code Huffman.
- **Exemple**:
- Pour ΔDC = 3 :
- Il appartient à la catégorie 2 (plage de 2 à 3).
- Code Huffman : 011.
- Bits supplémentaires : 11 (représentation binaire de 3).
- Valeur codée finale : 01111.

Category	Range	Huffman Code
0	0	00
1	-1 to 1	010
2	-3 to -2, 2 to 3	011
3	-7 to -4, 4 to 7	100
4	-15 to -8, 8 to 15	101
5	-31 to -16, 16 to 31	110
6	-63 to -32, 32 to 63	1110
7	-127 to -64, 64 to 127	11110
8	-255 to -128, 128 to 255	111110
9	-511 to -256, 256 to 511	1111110

Compression JPEG

6. Codage :

✓ **Run-Length Coding (RLC):**

- RLC est utilisé pour coder les **coefficients AC (Alternating Current)**, qui représentent les variations du bloc après la suppression de la composante DC.
- Après quantification, de nombreux coefficients AC sont nuls, en particulier pour les composants haute fréquence.
- Au lieu de stocker tous les zéros, RLC encode efficacement les séries de zéros consécutifs.
- Chaque coefficient AC différent de zéro est représenté par une paire **(Run, Value)**
  - **Run** : le nombre de zéros consécutifs avant ce coefficient.
  - **Value** : le coefficient différent de zéro réel.
- Un symbole spécial « **EOB** » est utilisé pour indiquer que toutes les valeurs restantes dans le bloc sont nulles (**1010**).

Compression JPEG

6. Codage :

- ✓ **Run-Length Coding (RLC):**

- **Exemple:**

- Exemple de données : 4,0,0,0,-2,0,0,3,0,0,0,0,0,-1,0,5, 0,…0

- Compression RLC (0,4) ; (3,-2) ; (2,3) ; (5,-1) ; (1,5) ; EOB.

Compression JPEG

6. Codage :

- ✓ **Run-Length Coding (RLC):**
- **Table Huffman :**
- Coefficients AC: **(2,-3)**
- **Run** = 2 (2 zéros avant -3).
- **Taille** = 2 (-3 nécessite 2 bits)
- Code de Huffman pour (2, 2) : **111010**.
- Bits supplémentaires pour -3 : **00** (représentation binaire de -3).
- Valeur codée finale : **11101000**.

(Run, Size)	Huffman Code
(0, 1)	00
(0, 2)	01
(0, 3)	100
(0, 4)	1010
(0, 5)	10110
(0, 6)	101110
(1, 1)	1100
(1, 2)	11010
(1,3)	111010
(2, 1)	11100
(2, 2)	111010
(3, 1)	1111000
(3,2)	1111100
(4, 1)	11111000
(5,1)	11111100
EOB (End of Block)	1010

Compression JPEG

6. Codage :

- ✓ **Run-Length Coding (RLC):**
- **Table Huffman : Exemple 2**
- **Compression RLC** (0,4) ; (3,-2) ; (2,3) ; (5,-1) ; (1,5) ; EOB.
- **(Run, Taille):** (0,3) ; (3,2) ; (2,2) ; (5,1) ; (1,3) ; EOB.
- **Code Hufman**: 100, 1111100,1111000,11111100,111010, 1010
- **Codage de magnitude** 4(100), -2 (01), 3(11), -1(0), 5(101)
- **Ajouter les bits de magnitude:**

100100 111110001 111100011 111111000 111010101 1010

(Run, Size)	Huffman Code
(0, 1)	00
(0, 2)	01
(0, 3)	100
(0, 4)	1010
(0, 5)	10110
(0, 6)	101110
(1, 1)	1100
(1, 2)	11010
(1,3)	111010
(2, 1)	11100
(2, 2)	111010
(3, 1)	1111000
(3,2)	1111100
(4, 1)	11111000
(5,1)	11111100
EOB (End of Block)	1010

Décompression JPEG

La décompression est l'inverse du processus de compression :

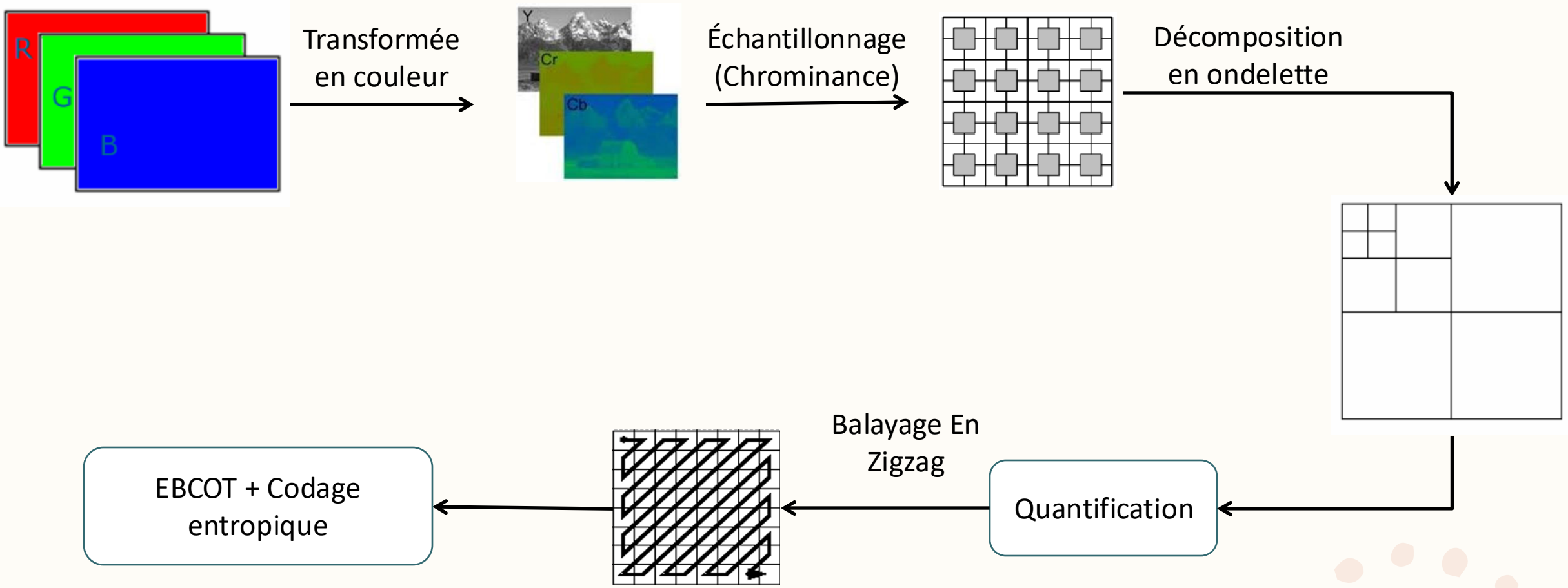
- ✓ Lit les données compressées.
- ✓ Applique le décodage Huffman pour restaurer les longueurs d'exécution et les coefficients DCT.
- ✓ Inverse l'étape de quantification (bien que ce ne soit pas exact, car la quantification élimine certaines informations).
- ✓ Effectue la DCT inverse (IDCT) pour revenir au domaine spatial.

$$Img(x,y) = \frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} c(u)c(v)F(u,v)\cos\left[\frac{\pi}{N}u\left(x+\frac{1}{2}\right)\right]\cos\left[\frac{\pi}{N}v\left(y+\frac{1}{2}\right)\right]$$

- ✓ Reconstruit l'image RGB en inversant la conversion de l'espace colorimétrique YCbCr en RGB.

JPEG 2000

- Les étapes de compression JPEG 2000:



JPEG 2000

- **Discrete Wavelet Transform (DWT): La transformation en ondelette :**

1. La transformation est réalisée sur l’image entière

2. Les algorithmes d'ondelettes ont deux composant:

    a- une fonction d'ondelette

    b- une fonction d'échelle (dénommé parfois filtre passe haut et filtre passe bas).

- Plusieurs types de transformations, on prend la transformation de **Haar**

JPEG 2000

- **La transformation en ondelette :**

✓ Pour calculer la transformée en ondelettes de **Haar** d'un tableau de **n = 2^m** données, il faut:

1. Calculer la moyenne de chaque paire de données (n/2 moyennes);
2. Calculer la différence entre chaque données et sa moyenne respective (n/2 différences);
3. Placer les moyennes dans la première moitié du tableau de données;
4. Placer les différences dans la seconde moitié du tableau de données;
5. Répéter le processus sur la première moitié des données

JPEG 2000

- **La transformation en ondelette :**

✓ Pour obtenir une décomposition sur une image , il faut appliquer l'algorithme sur les lignes, et ensuite sur les colonnes (LL = 1x1 ou 2x2).

✓ Exemple : décomposition pour obtenir le 1 niveau

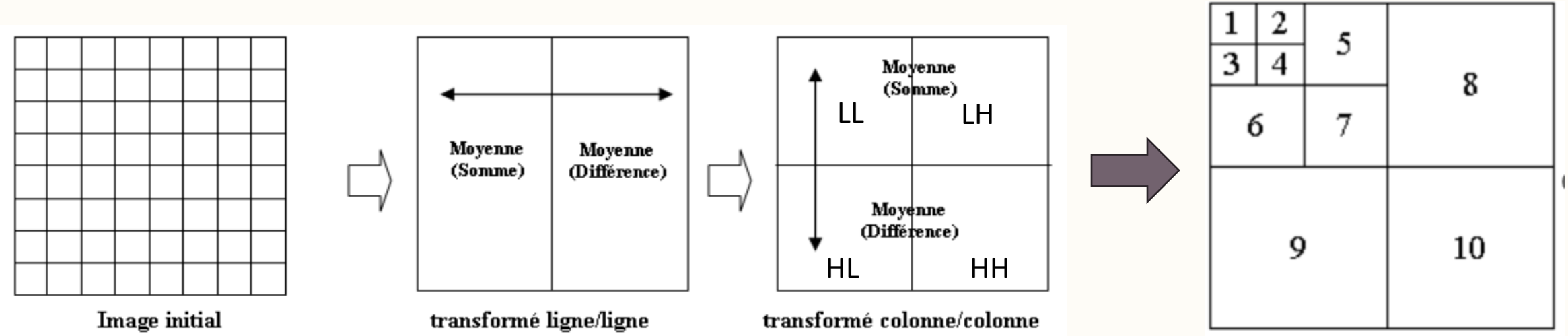


Image initial | transformé ligne/ligne | transformé colonne/colonne

JPEG 2000

- **La transformation en ondelette :**
- **Ligne/ligne:**

- [52, 55, 61 , 66]
  - Somme: 52+55/2= 53.5 , 61+66/2= 63.5
  - Différence: 52-55/2= -1.5 , 61-66/2= -2.5
- [55, 59, 60 , 65]
  - Somme: 55+59/2= 57 , 60+65/2= 62.5
  - Différence: 55-59/2= -2 , 60-65/2= -2.5
- [62, 59, 68 , 70]
  - Somme: 62+59/2= 60.5 , 68+70/2= 69
  - Différence: 62-59/2= 1.5 , 68-70/2= -1
- [63, 64, 70 , 74]
  - Somme: 63+64/2= 63.5 , 70+74/2= 72
  - Différence: 63-64/2= 0.5 , 70-74/2= -2

52	55	61	66
55	59	60	65
62	59	68	70
63	64	70	74

53.5	63.5	-1.5	-2.5
57	62.5	-2	-2.5
60.5	69	1.5	-1
63.5	72	0.5	-2

JPEG 2000

- **La transformation en ondelette :**
- **Colonne/ Colonne :**

- [53.5, 57, 60.5 , 63.5]
  - Somme: 53.5+ 57 /2= 55.25 , 60.5 + 63.5 /2= 62
  - Différence: 53.5-57 /2= -1.75 , 60.5 - 63.5 /2= -1.5
- [63.5, 62.5, 69 , 72]
  - Somme: 63.5+62.5 /2= 63 , 69 + 72 /2= 70.5
  - Différence: 63.5-62.5/2= 0.5 , 69 - 72 /2= -1.5
- [-1.5, -2, 1.5 , 0.5]
  - Somme -1.5 -2 /2= -1.75 , 1.5 + 0.5 /2= 1
  - Différence: - 1.5- (-2) /2= 0.25 , 1.5 - 0.5/2= 0.5
- [-2.5, -2.5, -1 , -2]
  - Somme: -2.5 -2.5 /2= -2.5, -1 -2 /2= -1.5
  - Différence: -2.5 – (-2.5) /2= 0 , -1 – (-2) /2= 0.5

53.5	63.5	-1.5	-2.5
57	62.5	-2	-2.5
60.5	69	1.5	-1
63.5	72	0.5	-2

55.25	63	-1.75	-2.5
62	70.5	1	-1.5
-1.75	0.5	0.25	0
-1.5	-1.5	0.5	0.5

Niveau 1

JPEG 2000

- **La quantification:**

✓ La quantification est faite pour rendre nuls les coefficients les plus faibles et faciliter le codage des autres

✓ On peut Chacune des sous-bandes(bloc) peut être quantifiée avec des étapes différentes

✓ On peut prendre un seuil unique pour toute l'image

- Exemle: **Q= 4**

55.25	63	-1.75	-2.5
62	70.5	1	-1.5
-1.75	0.5	0.25	0
-1.5	-1.5	0.5	0.5

$$\boldsymbol{round}\left(\frac{\boldsymbol{I(i,y)}}{\boldsymbol{Q}}\right) \longrightarrow$$

14	16	0	-1
16	18	0	0
0	0	0	0
0	0	0	0

JPEG 2000

- **Embedded Block Coding with Optimal Truncation (EBCOT)**

✓ EBCOT fonctionne sur un bloc de code (généralement 64 × 64 ou 32 x32 pixels avec balayage en zig-zag) et utilise une série de passages pour encoder les données de l'image:

1. Identifie les coefficients non nuls (significatifs) et encode leur signification.

✓ 14, 16, -1 et 18 sont **significatifs**.

✓ Tous les coefficients restants sont égaux à 0, ils ne sont donc pas significatifs.

✓ Le flux binaire(bitstream) pour le passage de signification est :

bitstream : **1110101000000000**

1	1	0	1
1	1	0	0
0	0	0	0
0	0	0	0

JPEG 2000

- **Embedded Block Coding with Optimal Truncation (EBCOT)**

2. Ajoute les bits les plus significatifs suivants pour chaque coefficient.

✓ Chaque coefficient sera traité un bit à la fois, en commençant par le bit le plus significatif (**MSB**) jusqu'à ce que tous les bits soient codés.

✓ **Représentation binaire**:

14 → 01110

16 → 10000

16 → 10000

18 → 10010

-1 → 11111

MSB	2ème MSB	3ème MSB	LSB
14 → 0 16 → 1 16 → 1 18 → 1 -1 → 1	14 → 1 16 → 0 16 → 0 18 → 0 -1 → 1	14 → 1 16 → 0 16 → 0 18 → 0 -1 → 1	14 → 1 16 → 0 16 → 0 18 → 1 -1 → 1

✓ **bitstream** : **1110101000000000 01111100011000110011**

JPEG 2000

- **Codage Arithmétique:**

✓ **Le codage arithmétique** encode un message entier ou une séquence de symboles sous la forme **d'un nombre unique**.

✓ Le processus commence par une séquence de **symboles** (par exemple, des caractères ou des pixels), **leurs probabilités d'occurrence**, et un **intervalle initial** : **[bas, haut] = [0, 1].**

✓ L'intervalle de 0 à 1 est **divisé en sous-intervalles**, chacun correspondant à un symbole du message. La taille de chaque sous-intervalle est proportionnelle à la probabilité du symbole correspondant.

✓ Après avoir traité toute la séquence, l'intervalle final est très petit et représente de manière unique l'ensemble du message. **Un numéro de cet intervalle final est choisi comme code pour le message.**

JPEG 2000

- **Codage Arithmétique:**

✓ **Exemple: "10110"**

✓ **Symboles** : {0, 1}

✓ **Probabilités** : P(0) = 0.4 | P(1) = 0.6

✓ **l'intervalle initial** de [0, 1]: **bas** = 0 et **haut** = 1

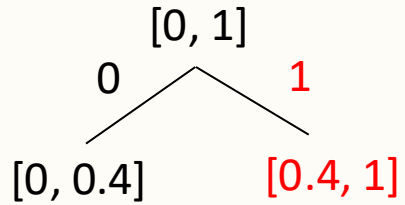
✓ Pour chaque symbole, nous allons subdiviser l'intervalle initial **[bas, haut]** en fonction des probabilités de « 0 » et « 1 ».

JPEG 2000

- **Codage Arithmétique:**

✓ **Exemple: "10110"**

✓ **10110**



✓ Pour '0' : P(0) = 0.4 → Cela donne une intervalle de [bas, bas + P(0)] = **[0, 0.4].**

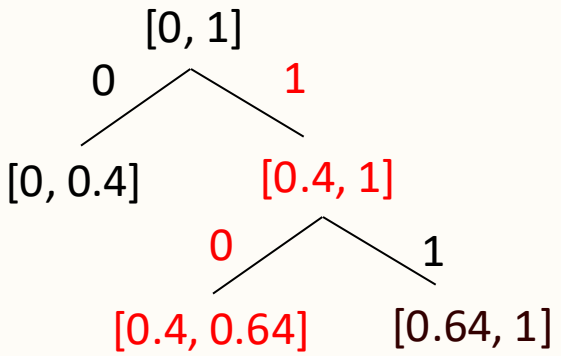
✓ Pour '1' : P(1) = 0.6 → Cela donne une intervalle de [bas + P(0), haut] = **[0.4, 1].**

JPEG 2000

- **Codage Arithmétique:**

✓ **Exemple: "10110"**

✓ **10110**



✓ Le deuxième symbole est « **0** ».

✓ L'intervalle actuel est **[0.4, 1].**

✓ Pour « **0** » : l'intervalle pour « **0** » est **[bas, bas + P(0)] = [bas, bas + bas*(haut-bas)]** =

[0.4, 0.4 + 0.4 * (1 – 0.4)] = **[0.4, 0.64].**

✓ Pour « **1** » : l'intervalle pour « **1** » est **[bas + P(0), haut]** = **[bas + bas*(haut-bas), haut]** = **[0.64, 1].**

JPEG 2000

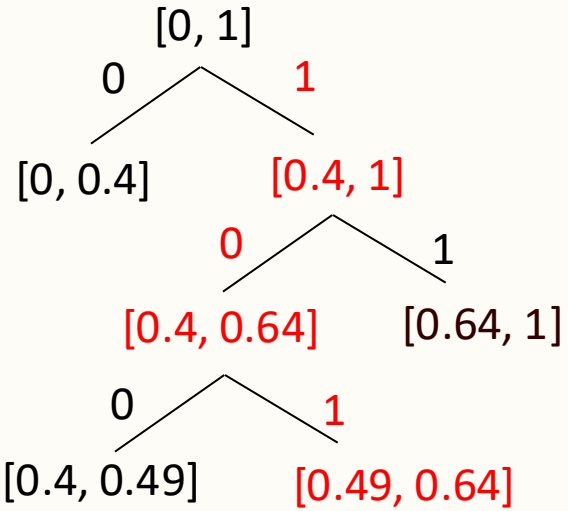
- **Codage Arithmétique:**

✓ **Exemple: "10110"**

✓ **10110**

✓ Le troisième symbole est « 1 ».

✓ L'intervalle actuelle est [0.4, 0.64].



✓ Pour « 0 » : l'intervalle pour « 0 » est [bas, bas + P(0)] = [0.4, 0.4 + 0.4 * (0.64 – 0.4)] = [0.4, 0.49].

✓ Pour « 1 » : l'intervalle pour « 1 » est [bas + P(0), haut] = [0.49, 0.64].

JPEG 2000

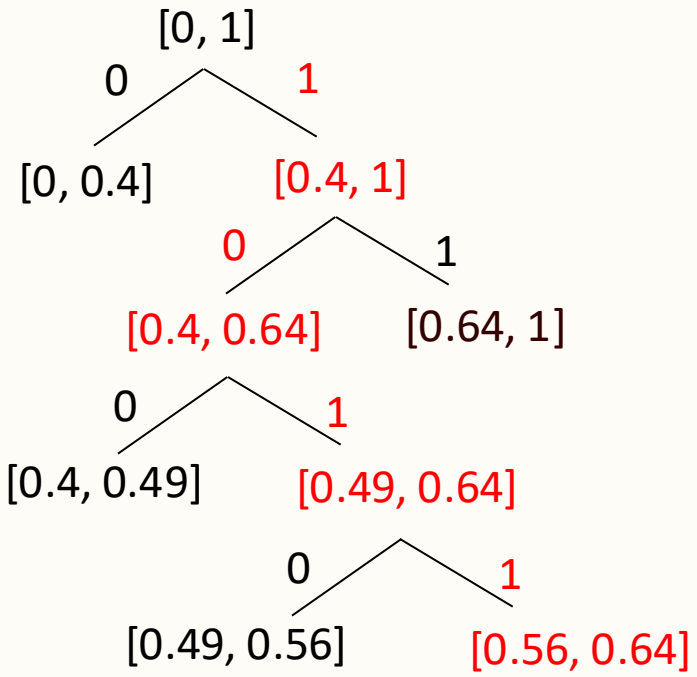
- **Codage Arithmétique:**

✓ **Exemple: "10110"**

✓ **10110**

✓ Le quatrième symbole est « 1 ».

✓ L'intervalle actuelle est [0.49, 0.64].



✓ Pour « 0 » : l'intervalle pour « 0 » est [bas, bas + P(0)] = [0.49, 0.49 + 0.49 * (0.64 – 0.49)] = [0.49, 0.56].

✓ Pour « 1 » : l'intervalle pour « 1 » est [bas + P(0), haut] = [0.56, 0.64].

JPEG 2000

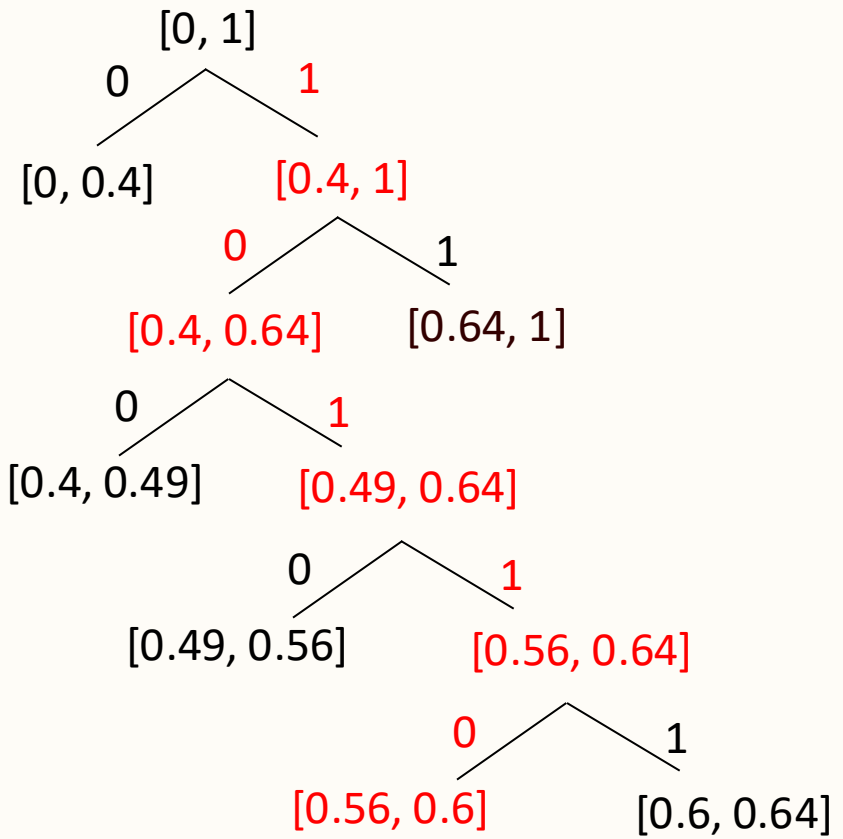
- **Codage Arithmétique:**

✓ **Exemple: "10110"**

✓ **10110**

✓ Le cinquième symbole « 0 ».

✓ L'intervalle actuelle est [0.56, 0.64].



✓ Pour « 0 » : l'intervalle pour « 0 » est [bas, bas + P(0)] = [0.56, 0.56 + 0.56 * (0.64 – 0.56)] = [0.56, 0.6].

✓ Pour « 1 » : l'intervalle pour « 1 » est [bas + P(0), haut] = [0.6, 0.64].

JPEG 2000

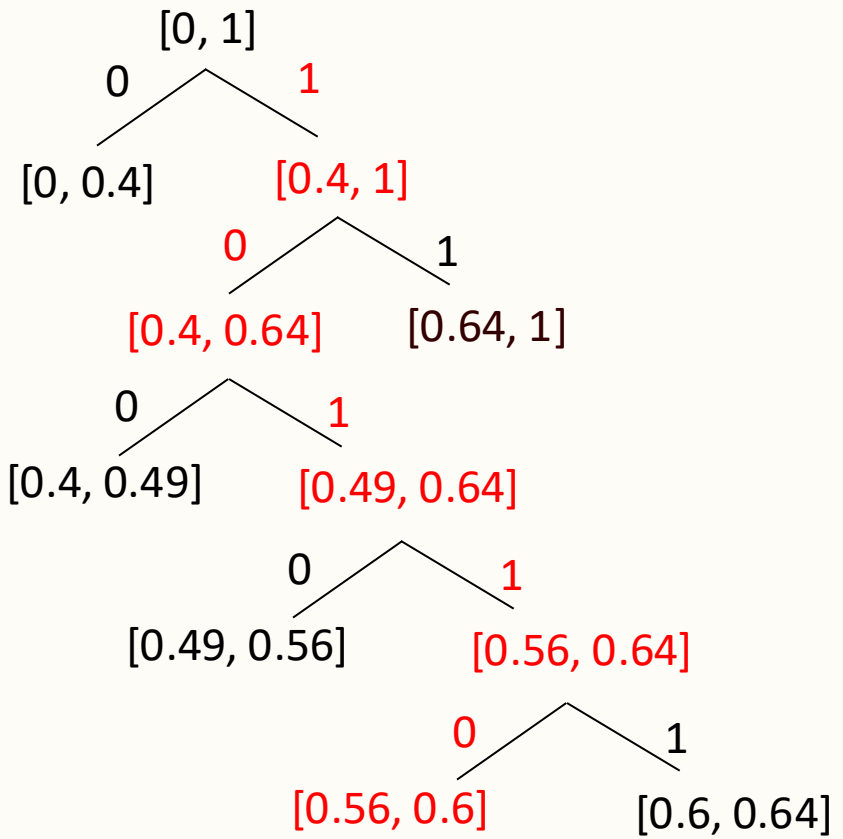
- **Codage Arithmétique:**

✓ **Exemple: "10110"**

✓ **10110**

✓ l'intervalle final est : [0.56, 0.6].

✓ Nous sélectionnons un nombre entre [0.56, 0.6]



✓ pour représenter la séquence entière « 10110 »: 0.56+0.6/2=0.58

JPEG LS

- **JPEG LS (Lossless and Near-Lossless Compression) :** est une norme de compression sans perte ou presque sans perte.
- **Les étapes de compression JPEG LS**

✓ Il utilise un schéma prédictif basé sur les trois voisins les plus proches (**left , top et top-left**).

✓ La valeur prédite est calculée en comparant les valeurs des pixels voisins (c'est-à-dire le prédicteur de bord médian ((Median Edge))).

✓ La valeur d'erreur de prédiction est la différence entre la valeur réelle du pixel et la valeur prédite. Les valeurs plus petits indiquent un potentiel de compression plus élevé.

✓ JPEG LS utilise le codage Golomb-Rice pour coder les valeurs d'erreur de prédiction.

JPEG LS

- **Les étapes de compression JPEG LS**

✓ **Exemple:**

✓ **Le prédicteur du bord médian:**

	TL	T	
	100	102	104
L	103	106	106
	108	111	115

$$\text{Prédiction(E)}= \begin{cases} min\,(TL,T)\ if\ L \geq max\,(TL,T) \\ max\,(TL,T)\ if\ L \leq min\,(TL,T) \\ TL+T-L \end{cases}$$

✓ **P(2,2) : TL=100, T= 102, L=103**

✓ $min\,(100,102)\ if\ 103 \geq max\,(100,102) = 100$

JPEG LS

- **Les étapes de compression JPEG LS**

✓ **Exemple:**

✓ **Le prédicteur du bord médian:**

Prédiction(p)= 100

✓ **Calculer l'erreur de prédiction:**

✓ **E=** Valeur actuelle- Prédiction(p) = 106-100 =6

	TL	T	
	100	102	104
L	103	106	106
	108	111	115

100	2	2
3	6	4
5	8	9

JPEG LS

- **Codage Golomb-Rice**

✓ Le codage Golomb-Rice est une forme de codage entropique très efficace pour coder de petits entiers.

✓ Le codage Golomb-Rice ne fonctionne qu'avec des entiers positifs, il traite donc les entiers négatifs en utilisant la méthode suivante :

$$e_m = \begin{cases} 2 \cdot e, & e \geq 0 \\ -2 \cdot e - 1, & e < 0 \end{cases}$$

✓ e_m = Mapped error

JPEG LS

- **Codage Golomb-Rice**

✓ Il fonctionne en deux parties :

✓ **Quotient** qui code le nombre de multiples d'une base M dans l'erreur de prédiction.

✓ **Reste** qui code le reste après la division.

✓ **La base M** utilisée pour le codage Golomb-Rice est sélectionnée en fonction de la taille moyenne des valeurs d'erreurs prédites dans l'image.

✓ $M = 2^k$

✓ $k = log_2(mean)$

JPEG LS

- **Codage Golomb-Rice**
- **Exemple:**
- **M=4**
- **P(0, 1)=2 -> 2/4 = 0 (quotient) | Reste = 2**
- Le quotient est **0** on a codé en codage unaire (deux zéros suivis d'un un) : **1**
- Le reste est 2: en binaire : **10**
- **Code Golomb-Rice pour 2 : 110**
- **P(2, 2)=6 -> 6/4 = 1 (quotient) | Reste = 2**
- Le quotient est **1:** codage unaire **= 01**
- Le reste est 2: en binaire : **10**
- **Code Golomb-Rice pour 6 : 0110**

100	2	2
3	6	4
5	8	9

-	110	110
111	0110	010
011	0010	0011

JPEG vs JPEG LS vs JPEG 2000

Aspect	JPEG	JPEG LS	JPEG 2000
Méthode de compression	Discrete Cosine Transform (DCT)	Median Edge	Discrete Wavelet Transform, DWT
Divisions d'images	Blocs 8x8	pixel par pixel	régions
Cas d'utilisation	Web, appareils photo numériques, stockage d'images	Imagerie médicale, archivage, fax	Imagerie professionnelle, cinéma numérique, archives de haute qualité, vidéo de haute qualité
Extensions de fichiers	.jpg, .jpeg	.jls	.jp2, .j2k
Efficacité de compression	Efficace pour la photographie, mais peut introduire des artefacts	Excellent pour la compression sans perte, bon pour les images simples	Excellent par rapport au JPEG
Complexité	Faible	Faible à modéré	Plus élevée

End