

Introduction au traitement d'images

Enseignement intégré

TS223 | Systèmes Électroniques Embarqués 2A | 2024-2025

Chapitre 6 : Compression

Rémi Giraud

remi.giraud@enseirb-matmeca.fr

<https://remi-giraud.enseirb-matmeca.fr/>

- **Introduction**
- **Formation / Acquisition**
- **Image couleur**
 - Format/Affichage/Synthèse
 - Espaces couleur caractéristiques (YCbCr)
 - Applications : compression, esquisse
- **Traitements**
 - Filtrage linéaire / non linéaire
 - Applications : débruitage, anonymisation
 - Détection de contours
 - Applications : réhaussement de contraste
- **Transformée de Fourier**
 - Application : recouvrement fréquentiel
- **Compression d'images**
 - Application : algorithme JPEG
- **Transformation spatiales**

COMPRESSION D'IMAGE

Compression d'image fixe

Propriétés

- Algorithmes « **lossless** » ou « **lossy** »
- « **Lossy** » : dégradation du contenu peu « perceptible »
- Fort facteur de compression ~ 10

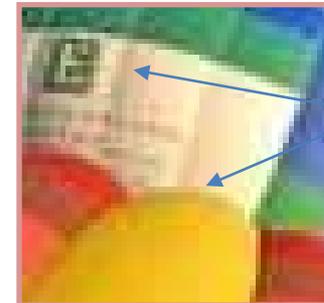
$$F = \frac{Q_{\text{initiale}}}{Q_{\text{finale}}}$$

Format	Dimensions spatiales	Taille mémoire
3 Mpixels	2048×1536	9 Mo
7 Mpixels	3072×2304	20,2 Mo
10 Mpixels	3648×2736	28,6 Mo
12 Mpixels	4000×3000	34,3 Mo

500×375, 549 ko



35,9 ko, $F \sim 15$ 17,9 ko, $F \sim 30$



dégradation
des contours
et des aplats
de couleurs

Compression de vidéo

Propriétés

- Algorithmes « **lossy** » nécessaires

Durée\Format	352×288 VCD	720×576 DVD Vidéo	1280×720 HD 720p	1920×1080 HD 1080p
1 s	7,60 Mo	31,1 Mo	69,1 Mo	156 Mo
1 min	456 Mo	1,87 Go	4,15 Go	9,33 Go
1 h	27,4 Go	112 Go	249 Go	560 Go
1 j	657 Go	2,69 To	5,97 To	13,4 To

25 image/s, format couleur = 3 canaux



**Gain F ~ 100 (MPEG-4 AVC)
à 400 (MPEG-4 SVC)**

NOTION D'INFORMATION UTILE

Définition

- **Entropie** : Variabilité/Quantité d'information d'un signal
- Information \approx Innovation
 - Degré de l'information inversement proportionnel à la probabilité d'apparition

Il neige à Tahiti

~~**Le soleil se lève à l'est**~~

- Notations
 - Chaîne de symboles (source) à compresser

Exemple : $S = \text{AABCABBCBBCDCCCD}$

- Notations

Dictionnaire et alphabet de symboles

Source

avec

$$D = \{C_i\}_{i=1:N}$$

$$S = \{C_j\}$$

$$C_j \in D$$

Définition

- Quantité (unitaire) d'information

$$Q(C_i) = -k \times \log(p_i)$$

$$\left\{ \begin{array}{l} k = \frac{1}{\log(K)} \\ p_i = \frac{n_i}{M} \end{array} \right.$$

Nombre d'états différents du codage

Nombre d'occurrence du symbole

Nombre total de symbole de la source

- Codage binaire

$$k = \frac{1}{\log(2)} \longrightarrow Q(C_i) = -\log_2\left(\frac{n_i}{M}\right)$$

Calcul de l'Entropie H :

- Espérance des quantités d'informations des différents symboles

$$H(S) = E\{Q(C_i)\} = - \sum_{i=1}^N \frac{n_i}{M} \log_2\left(\frac{n_i}{M}\right) \quad \text{bits/symbole}$$

Théorème du codage :

- L'entropie donne la limite basse théorique du nombre de bits en moyenne sur lequel coder un symbole :

$$L_{moy}(S) \geq H(S)$$

$$\text{avec } L_{moy}(S) = \sum_{i=1}^N p_i l_i$$

Exemples :

- 1)



$$H_1 = 1 \text{ bit/symbole}$$



$$H_2 = 7.56 \text{ bits/symbole}$$

- 2) $S = \text{AABCABBBCBBCDCCCD}$ $N = 4$ et $M = 16$
- 3) $S = 0011110011$?

Implémentation du calcul d'entropie

- Écrire une fonction **my_entropie** qui calcule l'entropie d'une image définie par des entiers non signés codés sur 8 bits et l'appliquer aux images : *implant.bmp* et *lezard.bmp*



- Rappels :
 - La fonction **np.histogram** permet d'obtenir le tableau d'occurrences $\{n_i\}$
- Comparer les résultats obtenus avec la fonction **shannon_entropy** de `skimage.measure`

Compression utilisant la cohérence spatiale

- Construire une matrice D qui stocke la différence d'intensité « colonne à colonne » de l'image A de taille $h \times w$:

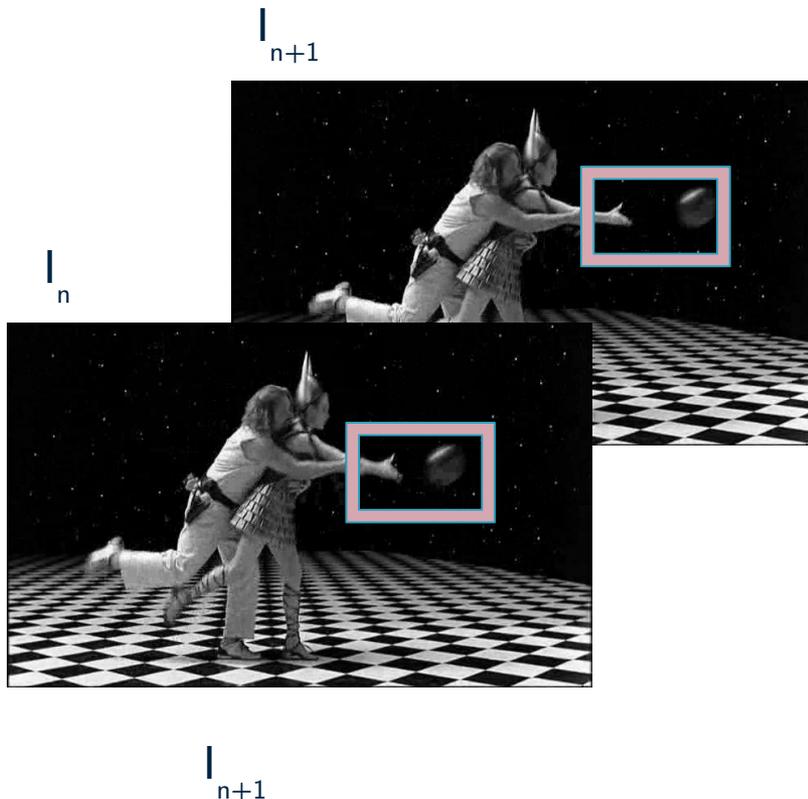
$$D[i, 0] = A[i, 0]$$

$$D[i, j] = A[i, j] - A[i, j - 1], \quad \text{pour } 1 \leq j \leq w - 1$$

- Calculer l'entropie de cette matrice et la comparer à l'entropie de l'image initiale.
- Observer l'histogramme de D et expliquer les résultats obtenus.
- Reconstruire l'image A , depuis D pour montrer qu'aucune information n'a été perdue.

Principe

- **Hypothèse** : Consistance du contenu au cours du temps
- **Méthodologie** : Ne coder que les différences d'un frame à l'autre
Block-Matching : Recherche de blocs similaires



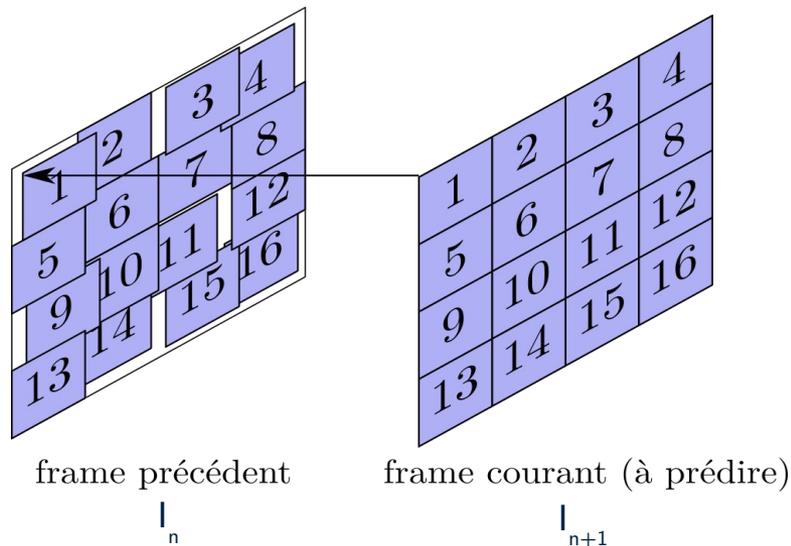
différence simple entre I_n et I_{n+1}



différence avec compensation de mouvement (I_{n+1} et I_{n+1} reconstruit depuis I_n)

Algorithme de Block-Matching

- Recherche de blocs/patches similaires dans une image de référence
 - On utilise les blocs de I_n pour estimer l'image I_{n+1} (compensation de mouvement)
 - Pour coder parfaitement I_{n+1} en connaissant I_n , on n'a alors besoin que de coder que les vecteurs de déplacement des blocs et la carte de différence entre l'image I_{n+1} estimée depuis les blocs de I_n (après compensation) et l'image I_{n+1} originale (faible entropie).



Algorithme de Block-Matching

- Recherche de blocs/patches similaires dans une image de référence



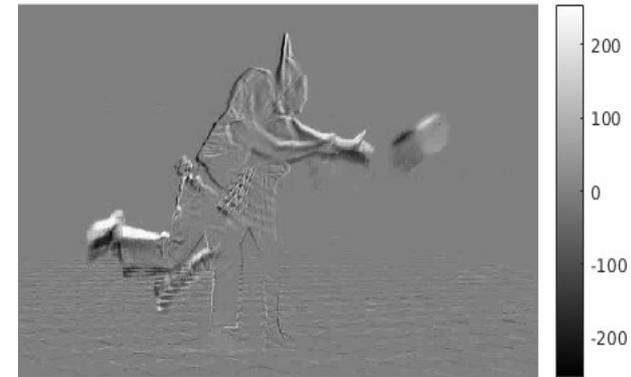
I_n



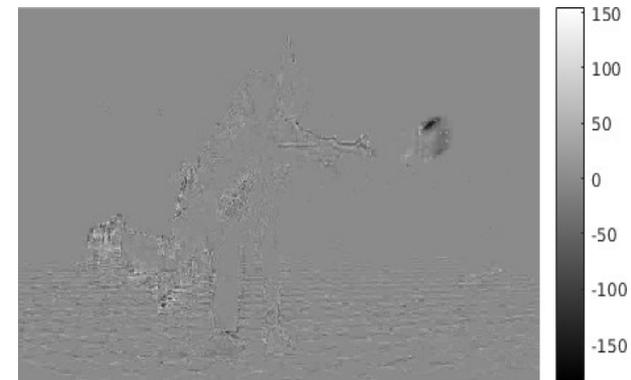
I_{n+1} reconstruit depuis
les blocs de I_n



I_{n+1}



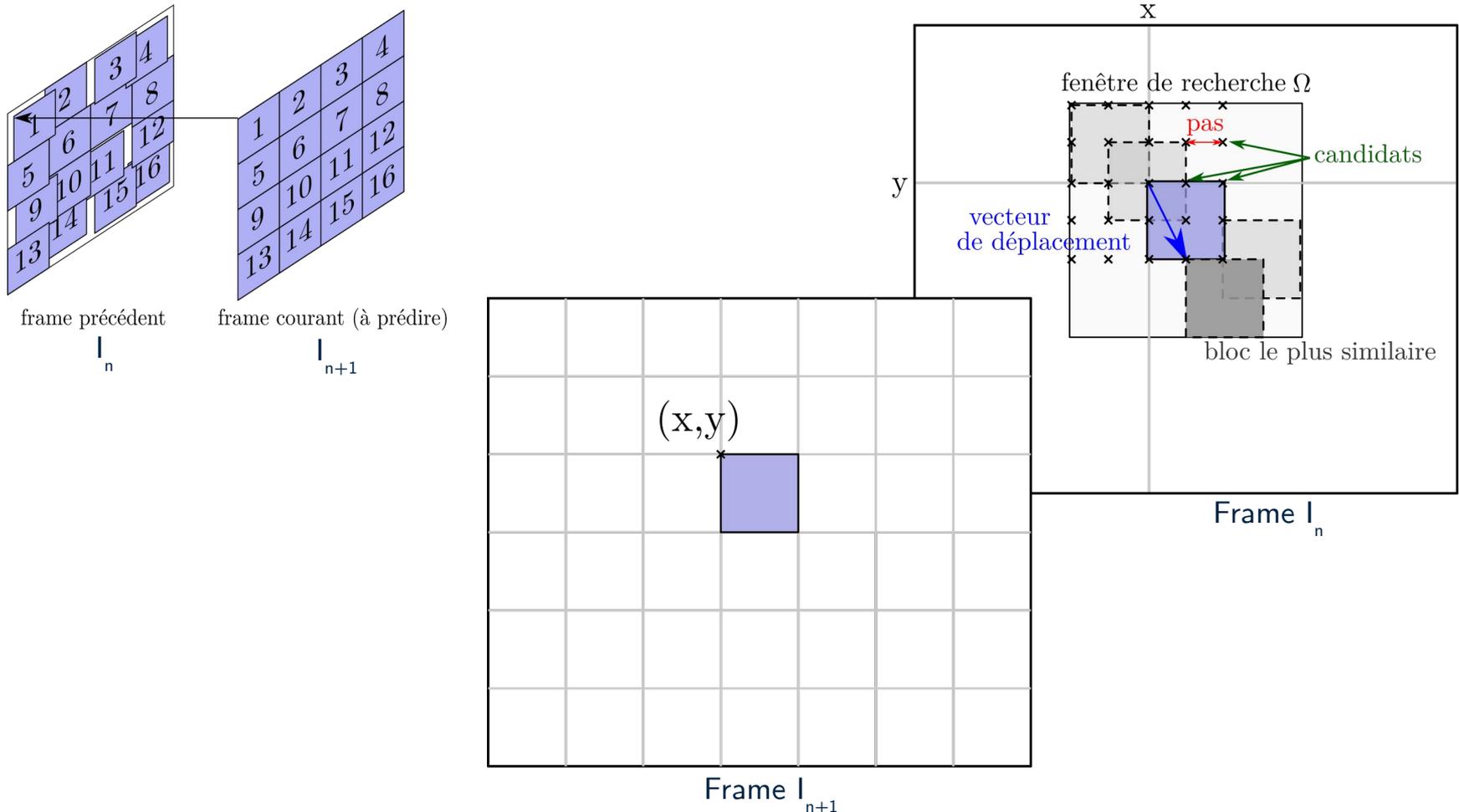
$I_n - I_{n+1}$



$I_n - I_{n+1}$ reconstruit

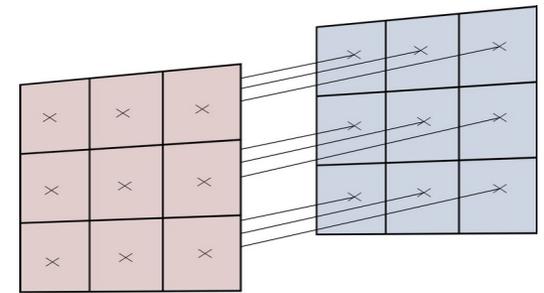
Algorithme de Block-Matching

- Recherche de blocs/patches similaires dans une image de référence



Paramètres du Block-Matching

- **Taille des blocs** ($ps \times ps$) : 4×4 , 8×8 , 16×16 , ...
- **Taille et pas de la fenêtre de recherche** Ω : $((2win+1) \times (2win+1))$
Pour un bloc dans I_{n+1} à la position (i,j) , on va tester tous les blocs dans I_n aux positions $(i+u, j+v)$ avec $(u, v) \in \Omega$
- **Algorithme** de recherche de candidats
 - Exhaustif ($pas = 1$ dans la figure précédente)
 - Diamant (sans garantie de trouver le meilleur bloc)
- **Distance** entre deux blocs en (i,j) dans I_{n+1} et $(i+u, j+v)$ dans I_n
 - SAD
 - SSD
 - ...



$$\sum_{d_i=0}^{ps-1} \sum_{d_j=0}^{ps-1} (I_{n+1}(i + d_i, j + d_j) - I_n(i + u + d_i, j + v + d_j))^2$$

COMPRESSION INFORMATION COULEUR

Espace RGB

- Sensibilité aux contrastes différente sur les trois canaux



Niveaux de gris



[0, 123, 0]



Palettes de primaires pures
(intensités identiques et autres composantes éteintes)

De nombreux espaces

luminance \rightarrow

$$Y = \alpha_R R + \alpha_V V + \alpha_B B + \beta_Y$$

chrominances $\begin{cases} \rightarrow \\ \rightarrow \end{cases}$

$$\begin{cases} U = \alpha_U (B - Y) + \beta_U \\ V = \alpha_V (R - Y) + \beta_V \end{cases}$$

ITU-R BT656 (CCIR 656)

ITU-R BT601 (CCIR 601)

ITU-R BT709 (CCIR 709)

norme

YUV

YC_bC_r

YCC

YIQ

noms

$Y \in [0,1]$

$Y \in [0,255]$

$Y \in [16,235]$

intervalles

PAL

SECAM

NTSC

JPEG 2000

JPEG

standard TV/vidéo/image

analogique

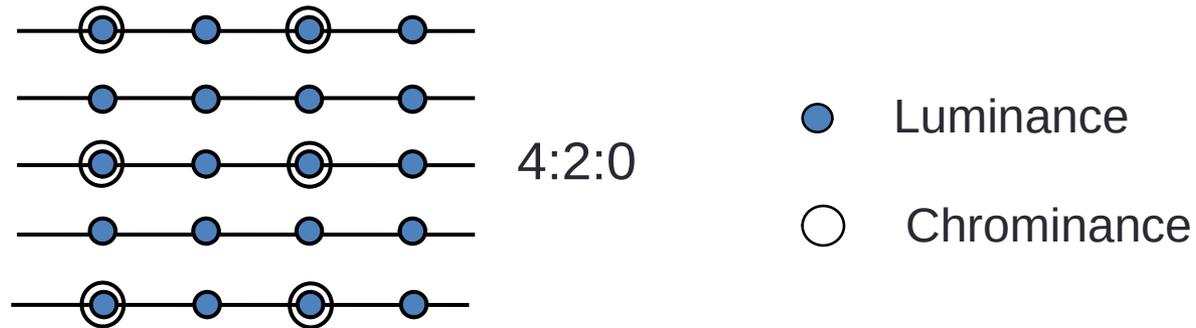
numérique

domaine

$$\begin{cases} Y = 0.299 R + 0.587 V + 0.114 B \\ U = 0.564(B - Y) \\ V = 0.713(R - Y) \end{cases}$$

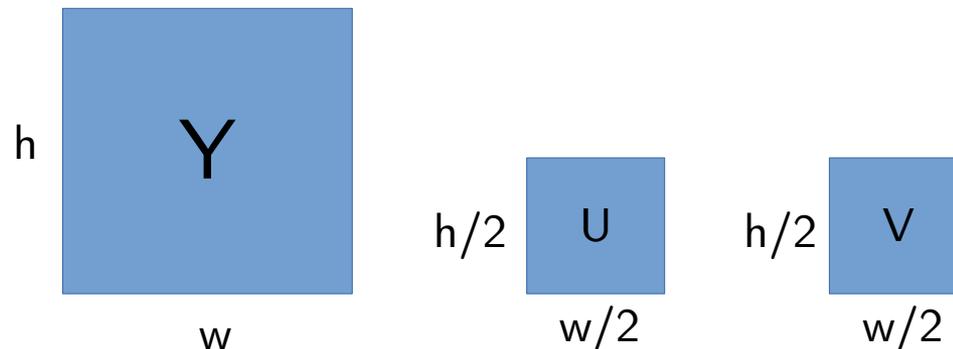
Sous-échantillonnage chromatique : Format N1:N2:N3

- N1 lignes, N2 : taux d'éch. ligne paire, N3 : taux d'éch. ligne impaire



- Format YUV 4:2:0 utilisé par les caméscopes DV en Europe :

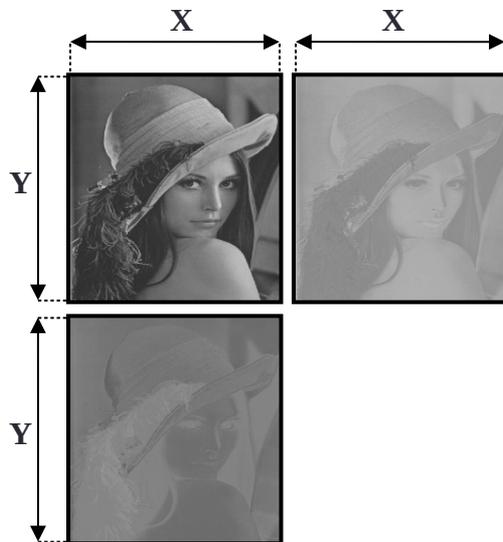
Images codées sous la forme de blocs de luminance de taille $W \times H$ suivis par deux sous-blocs de chrominance de taille $(W/2) \times (H/2)$



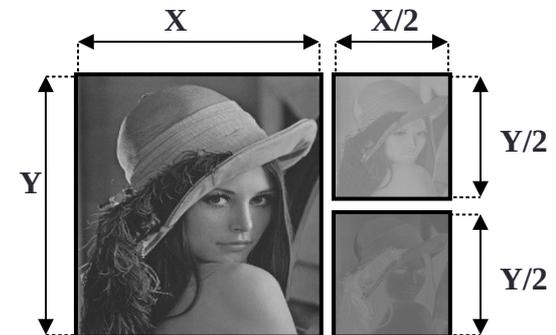
Sous-échantillonnage chromatique : Format N1:N2:N3

- Calcul de la taille mémoire en octets

$$N1:N2:N3 \quad (XY) \left[1 + 2 \left(\frac{N_2}{N_1} \frac{1}{2} + \frac{N_3}{N_1} \frac{1}{2} \right) \right]$$



Sous échantillonnage



Taille de la représentation (4:2:0)

$$YCrCb = 3/2(X*Y) \text{ octets}$$

NOTION DE CODAGE

Objectifs

- Minimiser la bande passante ou la taille de stockage
 - Cette minimisation peut être faite sans (*lossless*) ou avec (*lossy*) perte d'information
 - Secteur d'activité (biomédical, grand public, militaire ...)
 - Qualité de service (MPEG 4 avec le streaming)
- Quels algorithmes de compression d'images sont *lossy* ou *lossless* ?
 - JPEG, PNG, TIFF, ...

Principe

- Analyser la structure de la source pour coder efficacement chaque symbole

Codage par répétition (RLC)

- **Principe :**
 - Choix d'un caractère de contrôle
 - Coder les plages de k octets identiques
- **Exemple :**
 - I = 01abbbbcbZeed
 - I = 01a#6bcZeed
- **Problème :** trouver un caractère de contrôle non élément du dictionnaire.

Codes à longueur variable

- **Principe** : affecter une longueur de code proportionnelle à la rareté d'apparition des symboles

Image à coder

R	R	O	Y
O	O	O	Y
O	Y	Y	G
Y	Y	Y	G

Couleur	Code
R	001
O	01
Y	1
G	000

RR0YR00Y00YGYYYG



00100101101010110111000111000

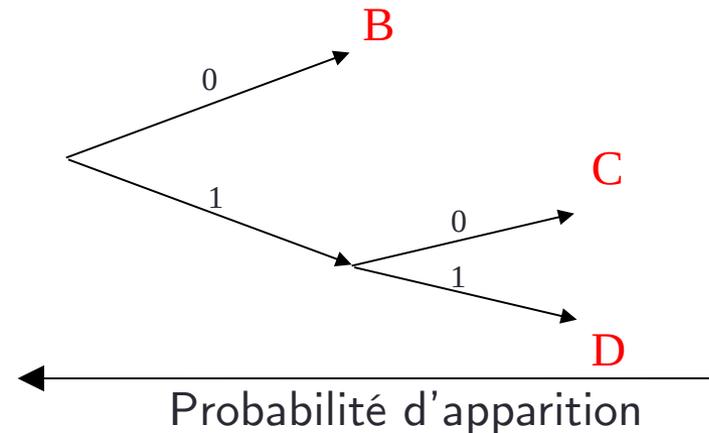
Règle du préfixe : un code est à décodage unique SSI aucun mot code n'est le **préfixe** d'un autre code.

Codage de Huffman

- **Principe** : Codage à arbre binaire selon la probabilité p_i d'apparition des symboles

$p_i \rightarrow 1$ $L(C_i)$ faible

$p_i \rightarrow 0$ $L(C_i)$ grande

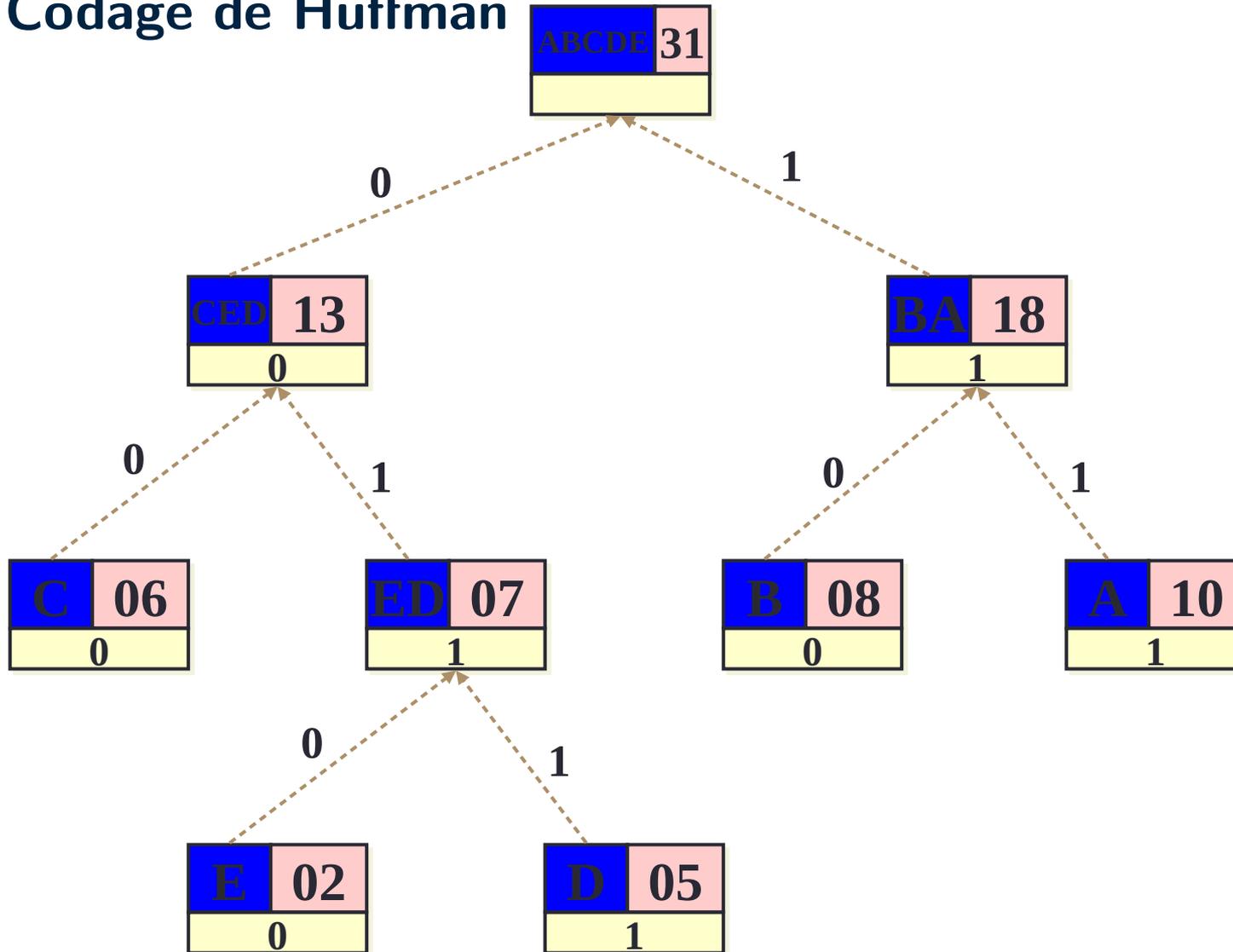


- **Méthodologie** :

- Ordonner les symboles dans l'ordre des probabilités décroissantes.
- Créer un symbole composite de probabilité égale à la somme des probabilités des deux symboles de plus faibles probabilités.
- Supprimer ces deux derniers.
- Répéter jusqu'à obtenir un symbole de probabilité égale à 1.

Notion de codage

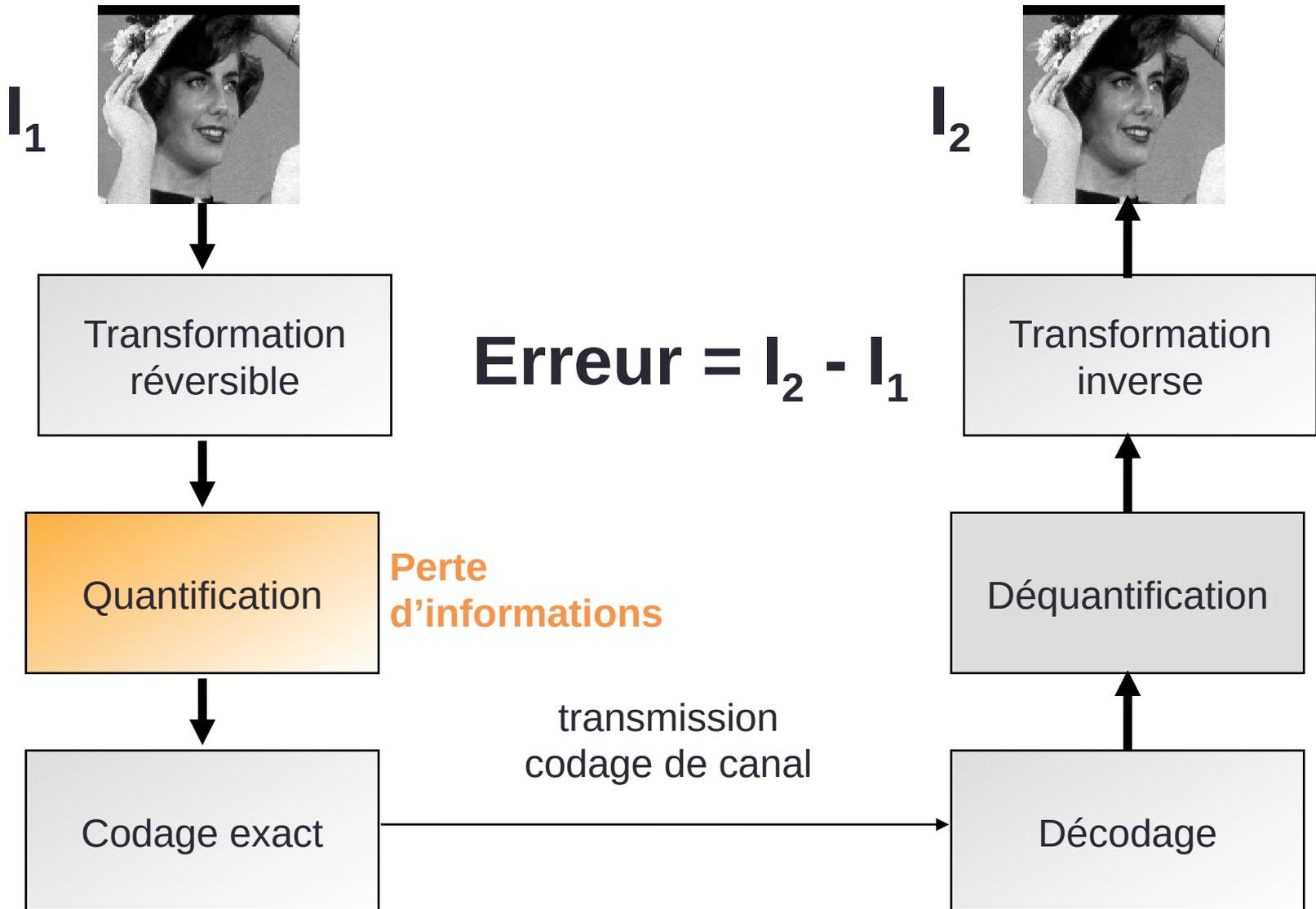
Codage de Huffman



A = 11
B = 10
C = 00
D = 011
E = 010

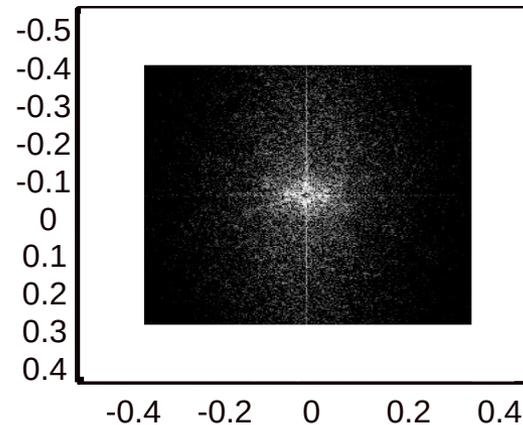
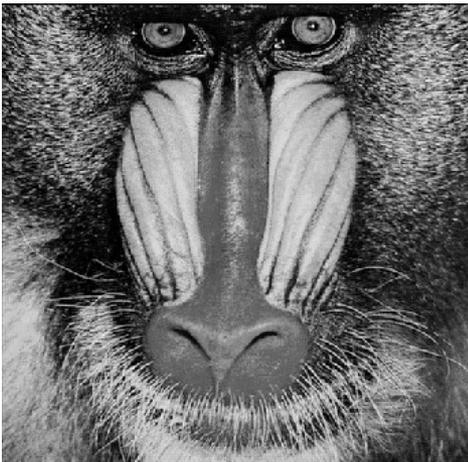
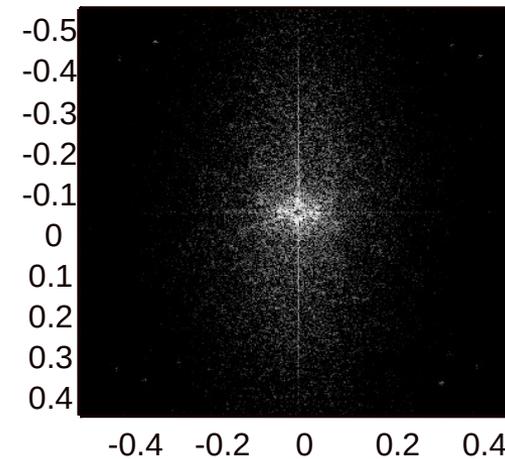
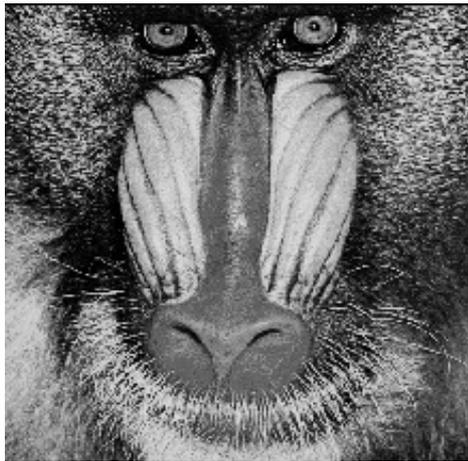
A = 10
B = 8
C = 6
D = 5
E = 2

Compression/Décompression



Principe psychovisuel

- La perception est peu sensible aux composantes hautes fréquences



Mise à zéro
des
coefficients
haute
fréquence

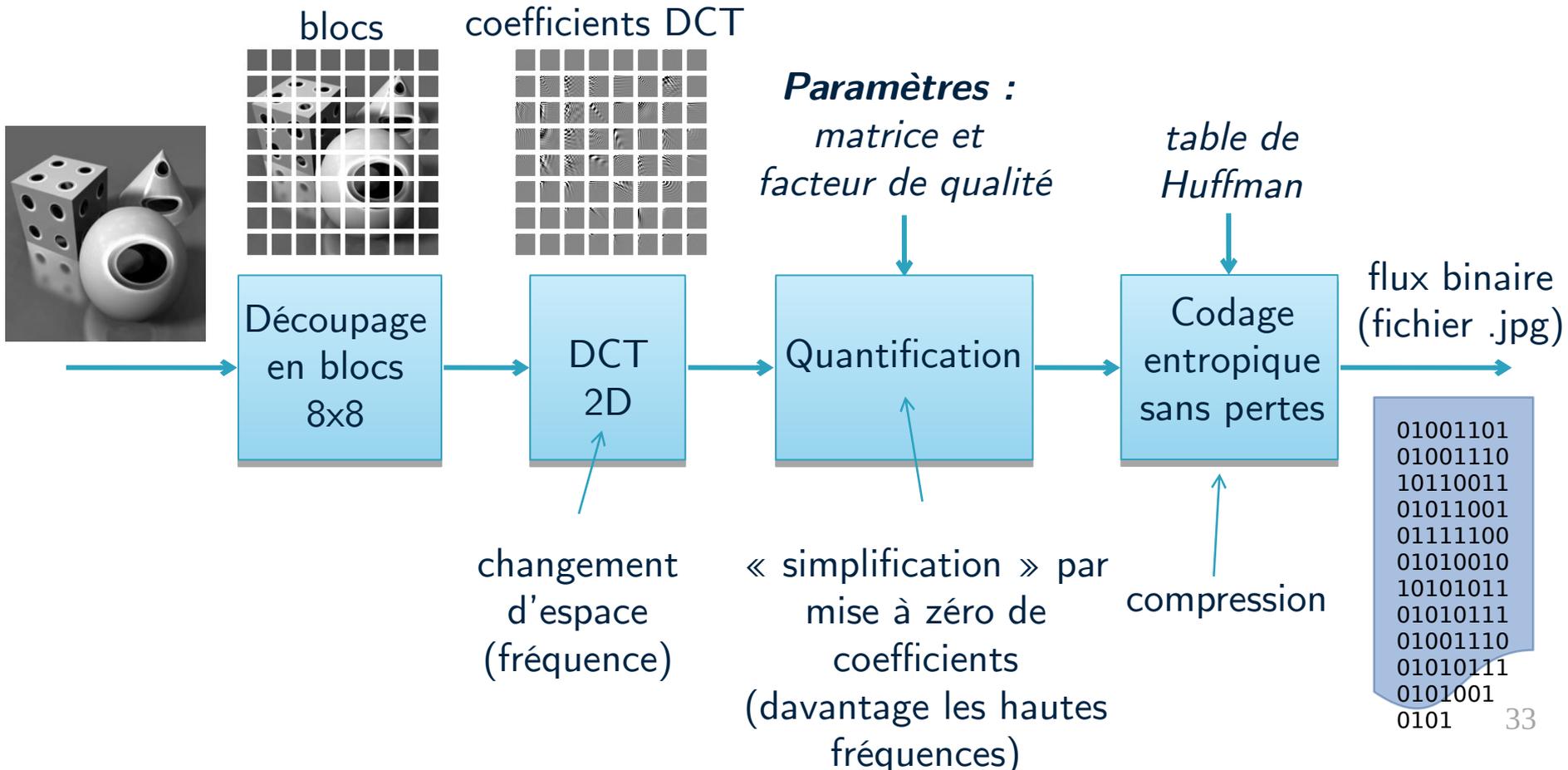
JPEG

Joint P hoto graphic E xpert G roup

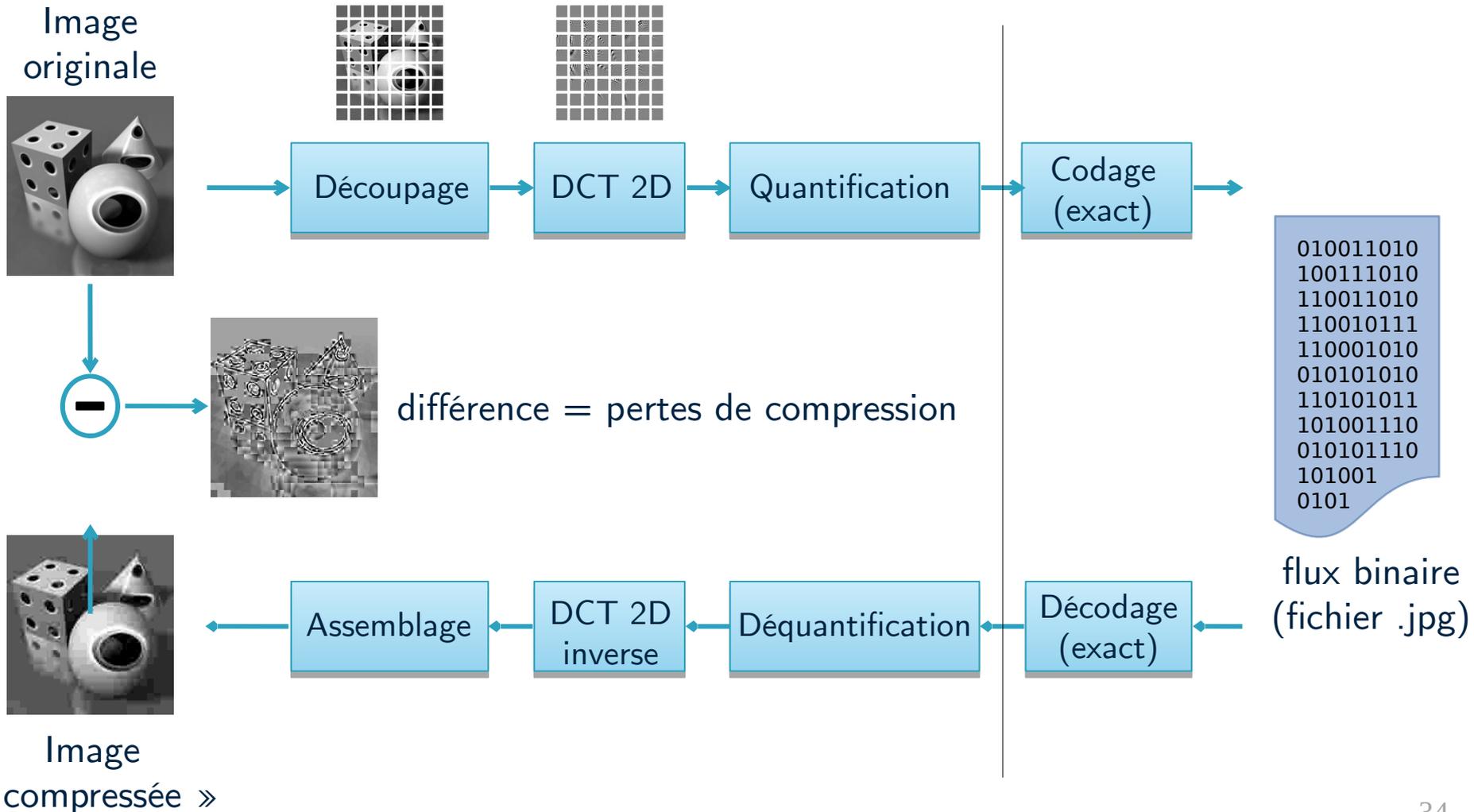
Norme ISO/IEC CCITT T81

Principe

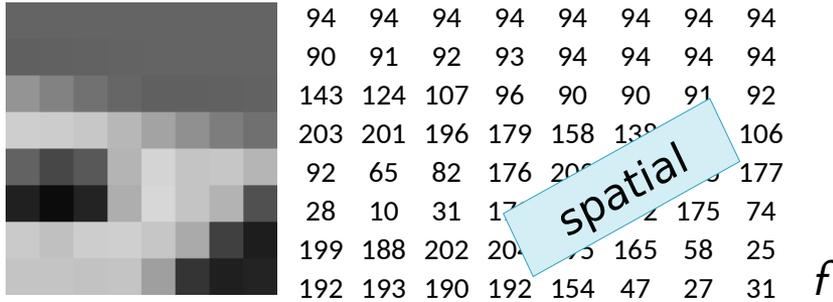
- Codage du contenu fréquentiel « perceptible »
- Traitement indépendant par blocs



Chaîne complète



Changement d'espace et « simplification »



Bloc 8x8

Transformée en
cosinus
(DCT)

	996,13	75,11	-123,06	50,57	39,88	13,67	-15,19	-1,40
1 coefficient DC	-122,05	-84,56	121,19	-27,83	-31,88	-3,89	10,27	-0,08
	-98,84	148,55	6,40	-42,46	-5,07	-19,66	10,44	4,80
	-24,01	-218,45	-76,09	67,08	8,07	22,27	10,17	0,17
63 coefficients AC	70,38	39,24	29,75	-44,52	16,13	10,14	10,24	7,89
	73,17	135,83	-0,77	2,11	-30,3	10,14	26,02	-5,54
	-89,35	-69,77	-0,04	13,08	12,25	10,44	-26,65	3,62
	10,98	-11,74	25,11	1,49	9,46	-8,00	14,95	-2,36

Quantification

entiers faibles et
plages de 0
codage efficace

24	2	-4	1	0	0	0	0
-4	-2	3	0	0	0	0	0
-2	4	0	0	0	0	0	0
0	-5	-1	0	0	0	0	0
1	0	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

Partie entière sur \mathbb{R}^+
Et entière +1 sur \mathbb{R}^-
→ simplification

$$F_Q = E_0 \left[\frac{F}{\alpha \times Q} \right]$$

fonction non-linéaire du
facteur de qualité (1 à 99)

Matrice 8x8
(normalisée ou au
choix) atténuant une
gamme désirée de
fréquences

fréquentiel

F_Q

Transformée DCT 1D

- Discrete Cosine Transform

$$F(k) = W(k) \sum_{n=0}^{N-1} f(n) \cos\left(\pi k \frac{2n+1}{2N}\right)$$

- Transformée directe
 - coefficients réels pour un signal réel
 - transformation réversible

Transformée de Fourier

$$F(k) = \sum_{n=0}^{N-1} f(n) e^{-j2\pi k \frac{n}{N}}$$

← base de cosinus ← base d'exponentielles

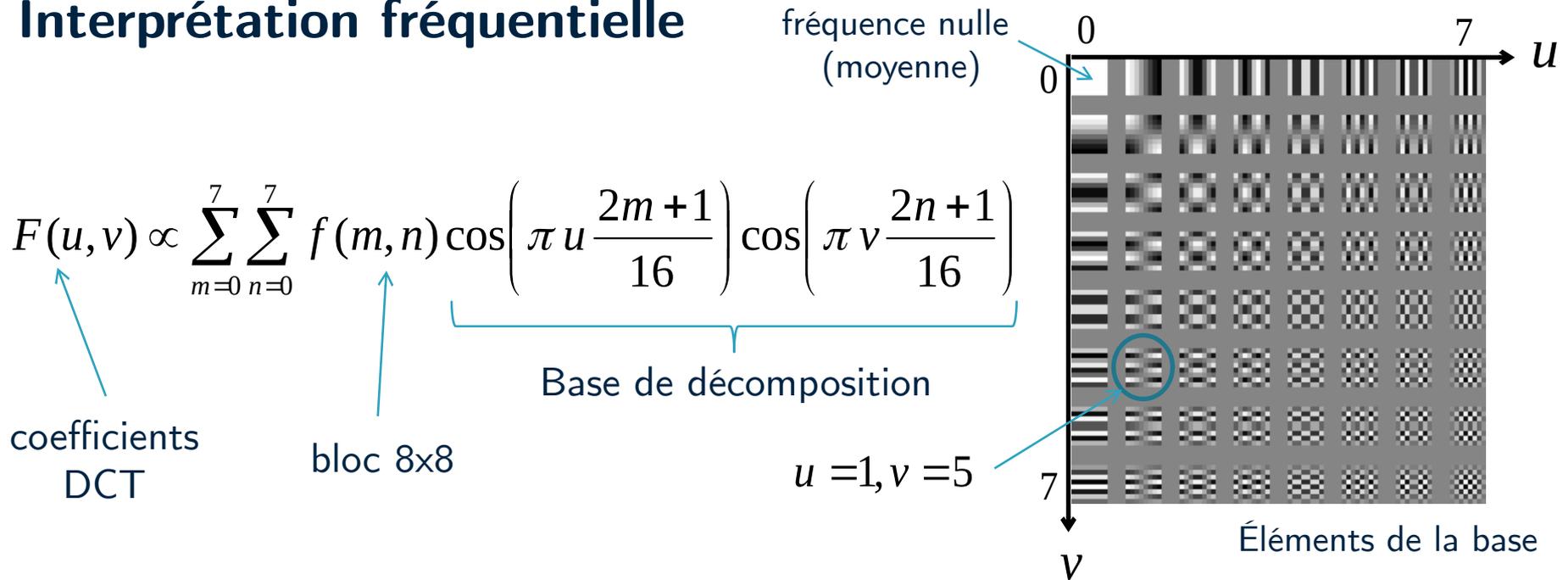
- Transformée inverse

$$f(n) = \sum_{k=0}^{N-1} W(k) F(k) \cos\left(\pi k \frac{2n+1}{2N}\right)$$

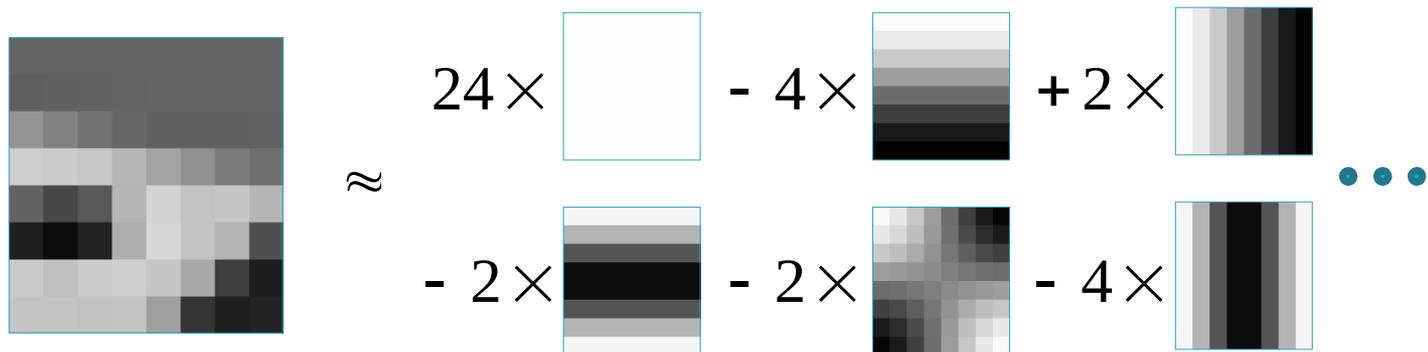
Transformée DCT 2D

$$F(u, v) = W(u, v) \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) \cos\left(\pi u \frac{2m+1}{2M}\right) \cos\left(\pi v \frac{2n+1}{2N}\right)$$

Interprétation fréquentielle



→ Décomposition approchée en une somme de motifs fréquentiels



Phase de quantification

$$F_Q = E \left[\frac{F}{\alpha \times Q} \right]$$

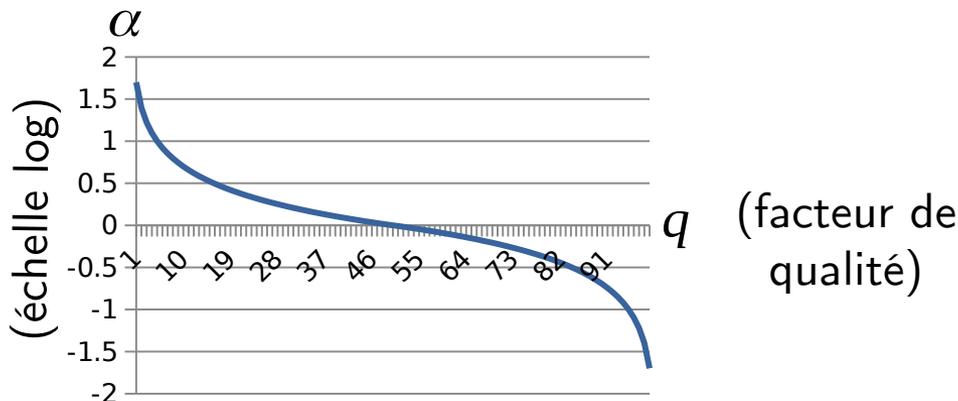
$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121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

Luminance
(Image niveaux de gris ou canal Y)

$$Q = \begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}$$

Chrominance
(Canaux Cb Cr)

$$\alpha = \begin{cases} \frac{50}{q} & \text{si } 1 \leq q \leq 50 \\ \frac{100 - q}{50} & \text{si } 50 \leq q \leq 99 \end{cases}$$



Évaluation des performances (quantitatif)

- Facteur (de compression) :

$$F = \frac{Q_{\text{initiale}}}{Q_{\text{finale}}}$$



Q_{finale}

Q_{initiale}

- Taux (de compression, exprimé en %) :

$$T = 100 \times \frac{Q_{\text{initiale}} - Q_{\text{finale}}}{Q_{\text{initiale}}} = 100 \left(1 - \frac{1}{F} \right)$$

Évaluation des performances (qualitatif)

- **Rapport Signal à Bruit (PSNR)**
 - Compare l'image initiale et l'image compressée
 - Métrique **globale** sur l'image, sans information de structure

$$PSNR = 10 \log_{10} \left(\frac{255^2}{MSE} \right) \quad \text{avec} \quad MSE = \frac{1}{hw} \sum_{i=1}^h \sum_{j=1}^w (I(i, j) - I_c(i, j))^2$$

PSNR = 24.85



Image initiale



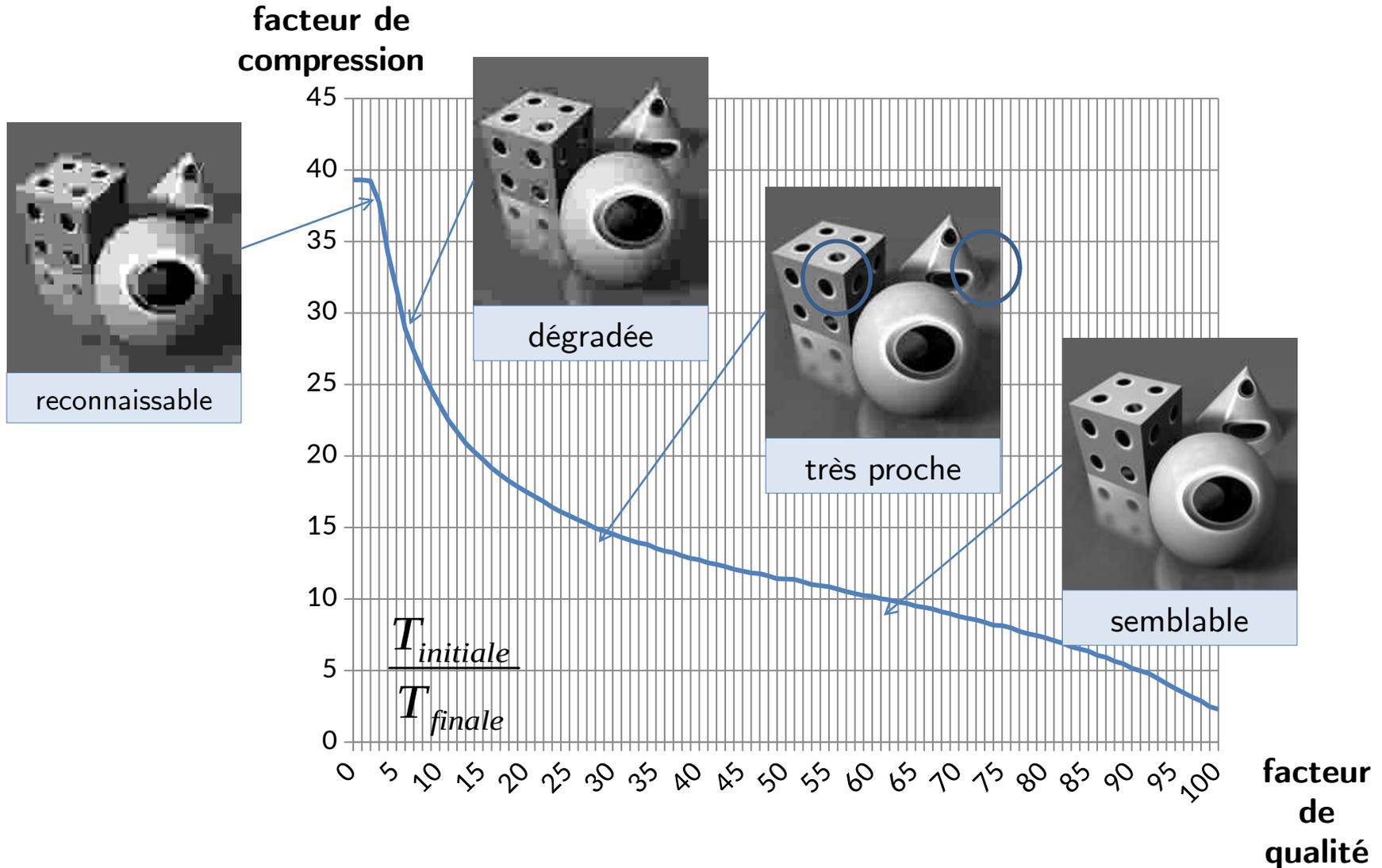
Image initiale - 15

PSNR = 25.72

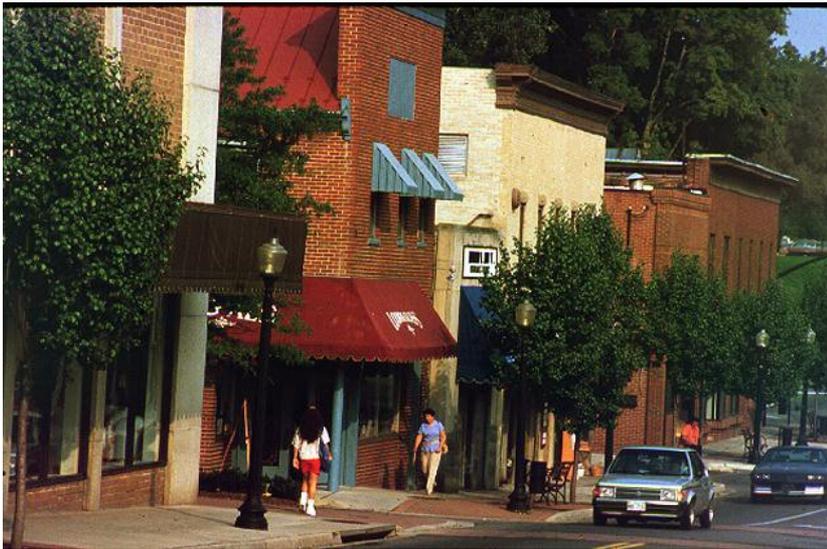


Image initiale - bloc

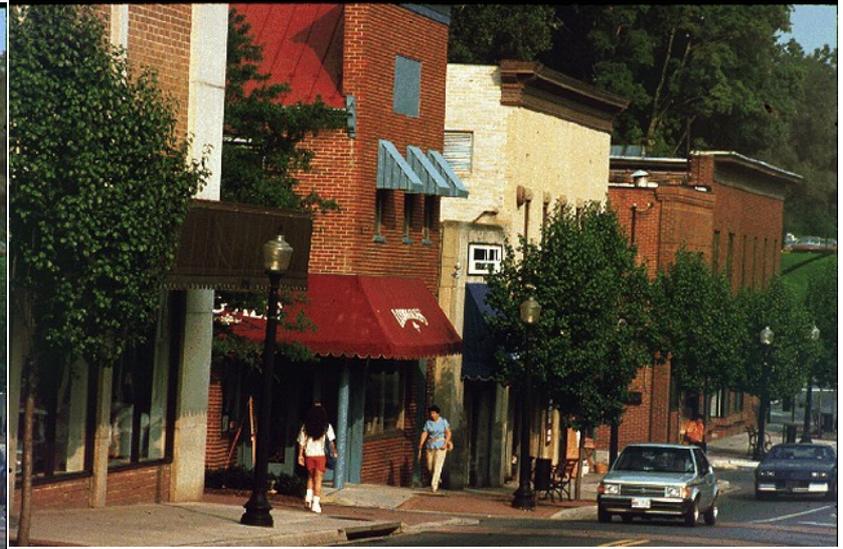
Évaluation des performances (quantitatif vs qualitatif)



Compression Jpeg



Original



TC=3.36



TC=16.89



TC=27.431

Implémentation de l'algorithme

Compléter le script : *jpeg.py*

- 1) Convertir le facteur de qualité q en paramètre de quantification *alpha*
- 2) Appliquer la DCT (*dct2*)
- 3) Appliquer la quantification (*fix* pour la troncature)
- 4) Réaliser la reconstruction : déquantification, DCT inverse (*idct2*)
- 5) Appliquer l'algorithme à tous les blocs de l'image (*blockproc*)
- 6) Constater le gain en taille mémoire de stockage et l'entropie réduite

```
cv2.imwrite('cameraman.jpg', img, [cv2.IMWRITE_JPEG_QUALITY, 100])
file_stat = os.stat('cameraman.jpg')
size_init = file_stat.st_size #%en octets
```
- 7) Comparer la qualité de manière qualitative en calculant le PSNR entre l'image initiale et l'image compressée