Introduction au traitement d'images

Enseignement intégré
IT220 | Informatique 2A | 2023-2024

Chapitre 6 : Compression

Rémi Giraud
remi.giraud@enseirb-matmeca.fr
https://remi-giraud.enseirb-matmeca.fr/

Plan

- **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

Propriétés

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

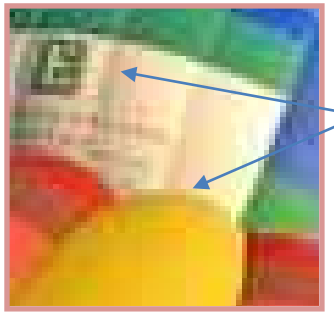
$$F = \frac{Q_{\text{initiale}}}{Q_{\text{finale}}}$$

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 ~15

17,9 ko, F ~30



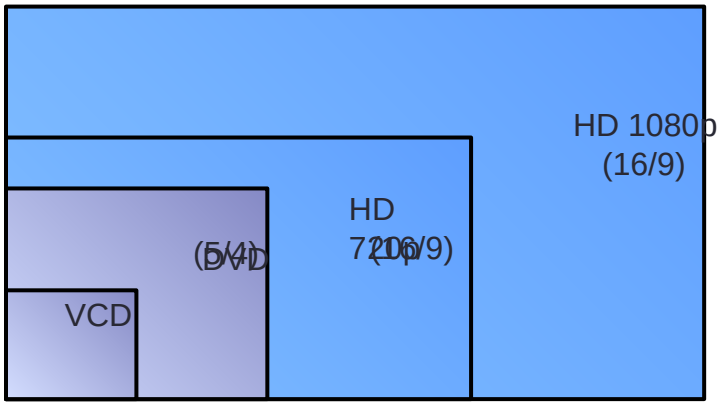
dégradation des contours et des aplats de couleurs

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 ≈ 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 : $S = \mathrm{AABCABBCBBCDCCCD}$
  - Notations

    Dictionnaire et alphabet de symboles $D = \{C_i\}_{i=1:N}$

    Source $S = \{C_j\}$ avec $C_j \in D$

Définition

- Quantité (unitaire) d'information

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

$$k = \frac{1}{log(K)}$$

Nombre d'états différents du codage

$$p_i = \frac{n_i}{M}$$

Nombre d'occurrence du symbole

Nombre total de symbole de la source

- Codage binaire

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

Calcul de l'Entropie H :

- Espérance des quantités d'informations des différents symboles

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

Théorème du codage :

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

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

Exemples :

- 1)

$H_1 = 1$ bit/symbole

$H_2 = 7.56$ bits/symbole

- 2) $S = \text{AABCABBCBBCDCCCD}$ $\quad \text{N} = 4 \quad$ et $\quad \text{M} = 16$

- 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

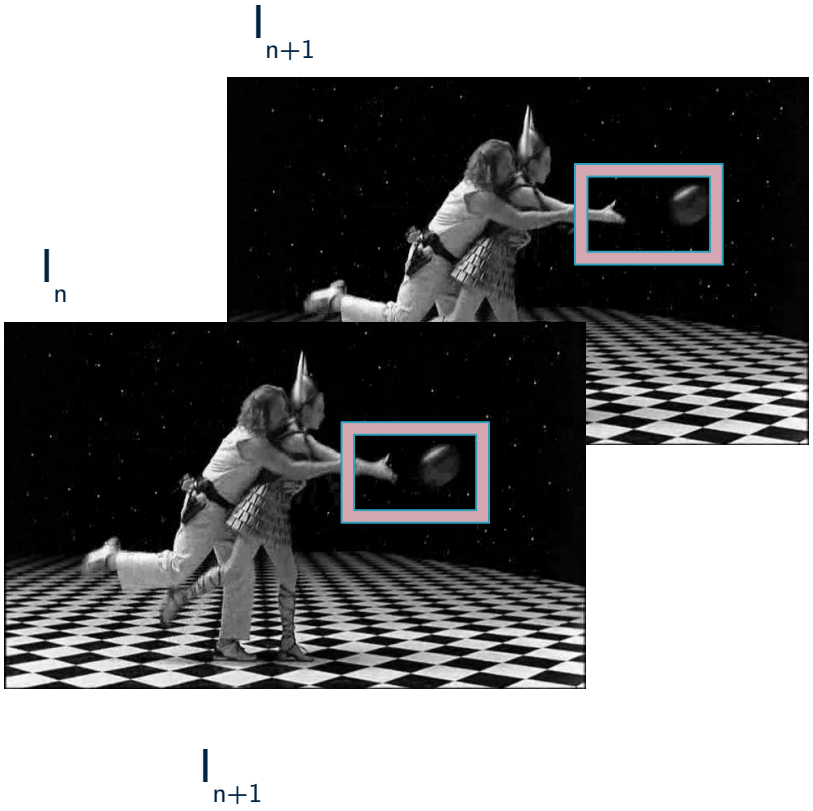
- Construire une matrice D qui stocke la différence d'intensité « colonne à colonne » de l'image A de taille hxw :

$$D[i,0] = A[i,0]$$
$$D[i,j] = A[i,j] - A[i,j-1], \quad \text{pour } 1 \leq j \leq w-1$$

- 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

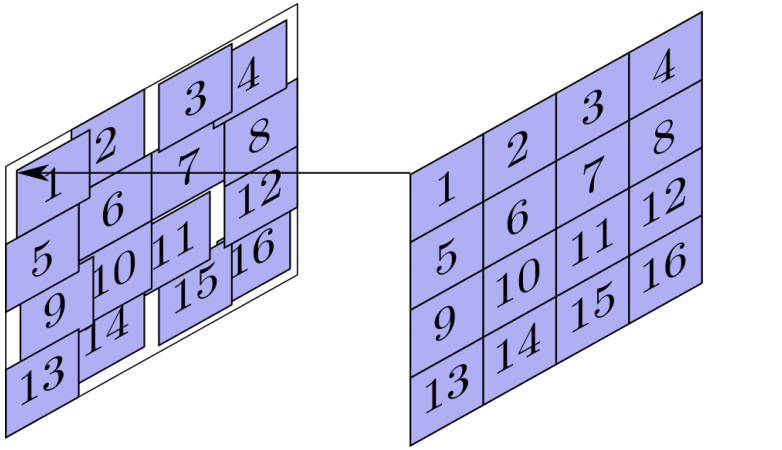


différence simple entre I_n et I_{n+1}

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

Algorithme de Block-Matching

- Recherche de blocs/patchs 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).



frame précédent I_n frame courant (à prédire) I_{n+1}

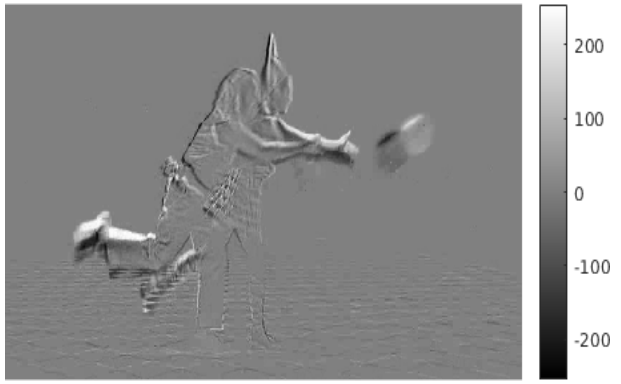
Algorithme de Block-Matching

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

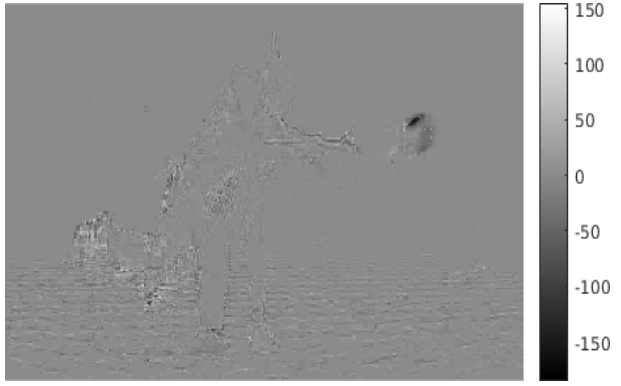
I_n

I_{n+1}

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



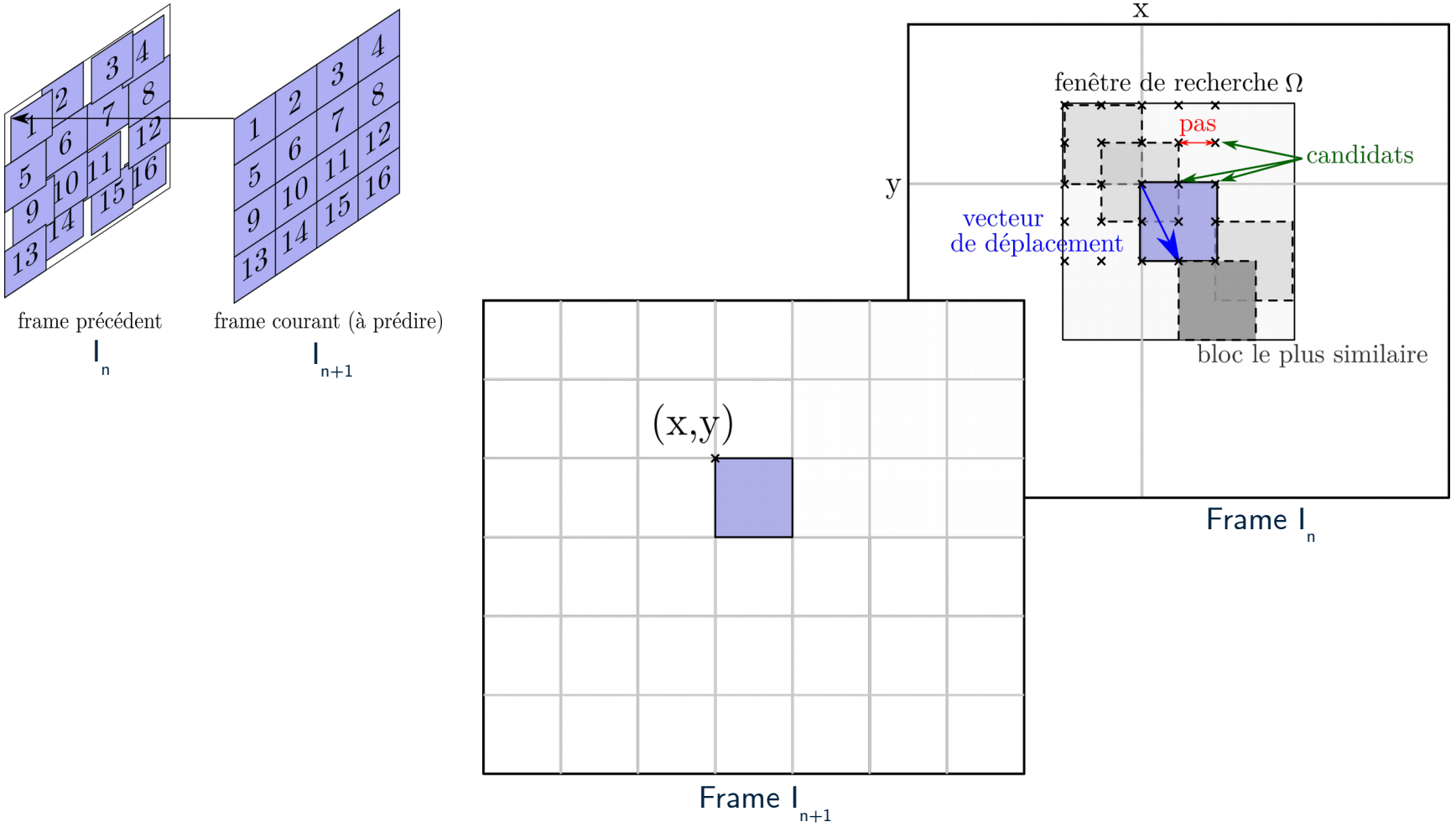
$I_n - I_{n+1}$



$I_n - I_{n+1}$ reconstruit

Algorithme de Block-Matching

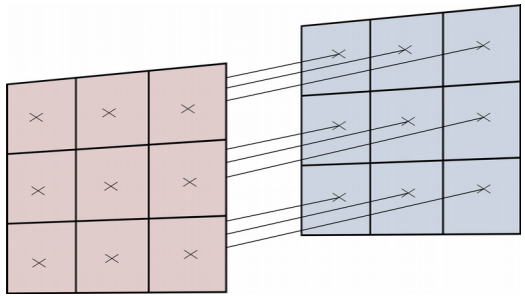
- Recherche de blocs/patchs similaires dans une image de référence



Paramètres du Block-Matching

- **Taille des blocs** (*ps x ps*) : 4×4, 8×8, 16×16, ...
- **Taille et *pas* de la fenêtre** de recherche Ω : *((2win+1)x(2win+1))*

  Pour un bloc dans In+1 à la position (i,j), on va tester tous les blocs dans In 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 In+1 et (i+u,j+v) dans In
  - SAD
  - SSD
  - ...



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

COMPRESSION INFORMATION COULEUR

Espace RGB

- Sensibilité aux contrastes différente sur les trois canaux

Niveaux de gris

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

De nombreux espaces

luminance

chrominances

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

ITU-R BT656 (CCIR 656)
ITU-R BT601 (CCIR 601)
ITU-R BT709 (CCIR 709)

norme

analogique
numérique

domaine

YUV YC_bC_r YCC YIQ

noms

$Y \in [0,1]$
$Y \in [0,255]$
$Y \in [16,235]$

intervalles

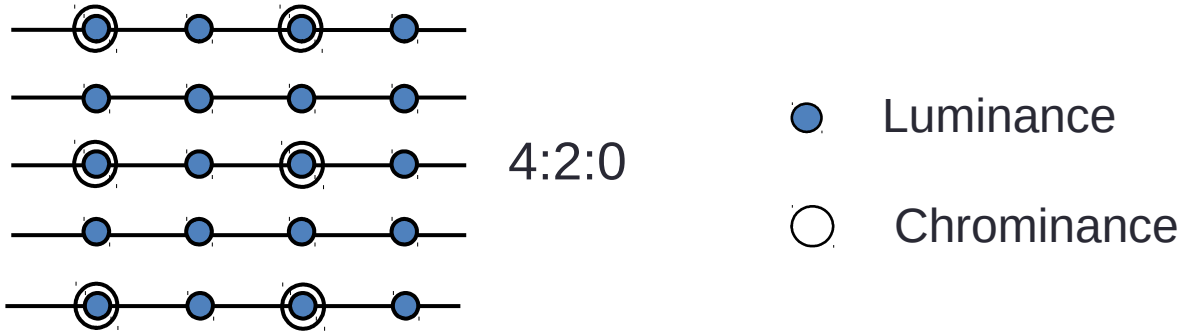
PAL NTSC JPEG 2000 SECAM JPEG

standard TV/vidéo/image

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

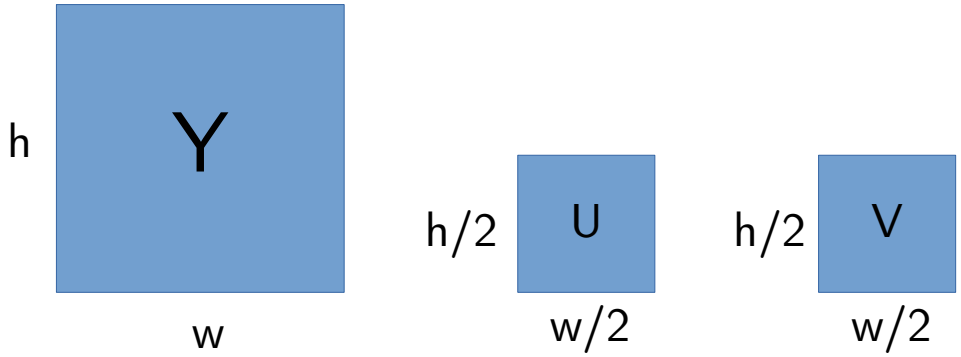
Sous-échantillonnage chromatique : Format N1:N2:N3

- 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 :

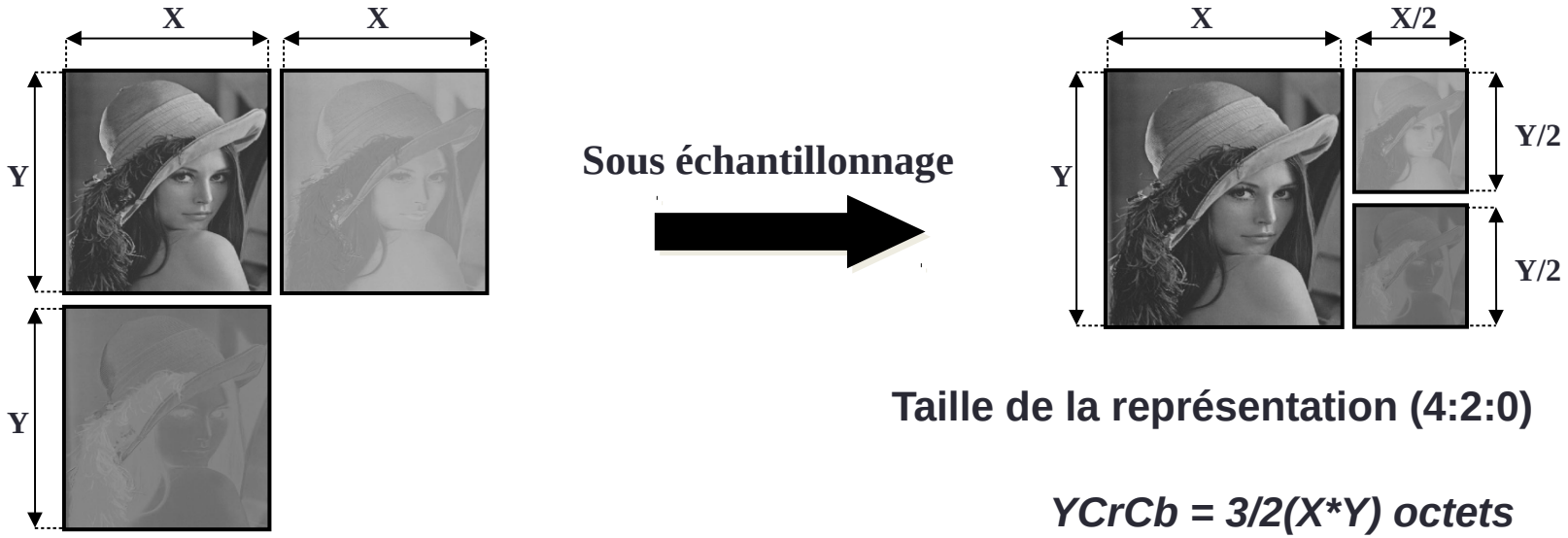
  Images codées sous la forme de blocs de luminance de taille W×H suivis par deux sous-blocs de chrominance de taille (W/2)×(H/2)



Sous-échantillonnage chromatique : Format N1:N2:N3

- Calcul de la taille mémoire en octets

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



Taille de la représentation (4:2:0)

YCrCb = 3/2(X*Y) 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= 01abbbbbbcZeed

  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

↓

0010010110101011011000111000

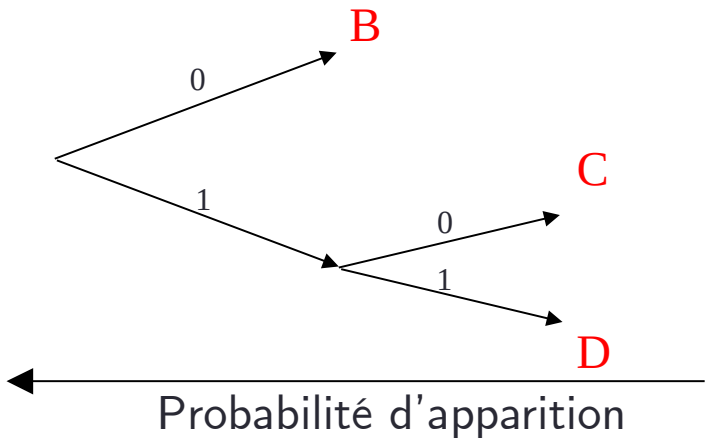
Règle du préfixe : un code est à décodage unique SSI aucun mot code n'est le **préfixe** d'un autre code.

Codage de Huffman

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

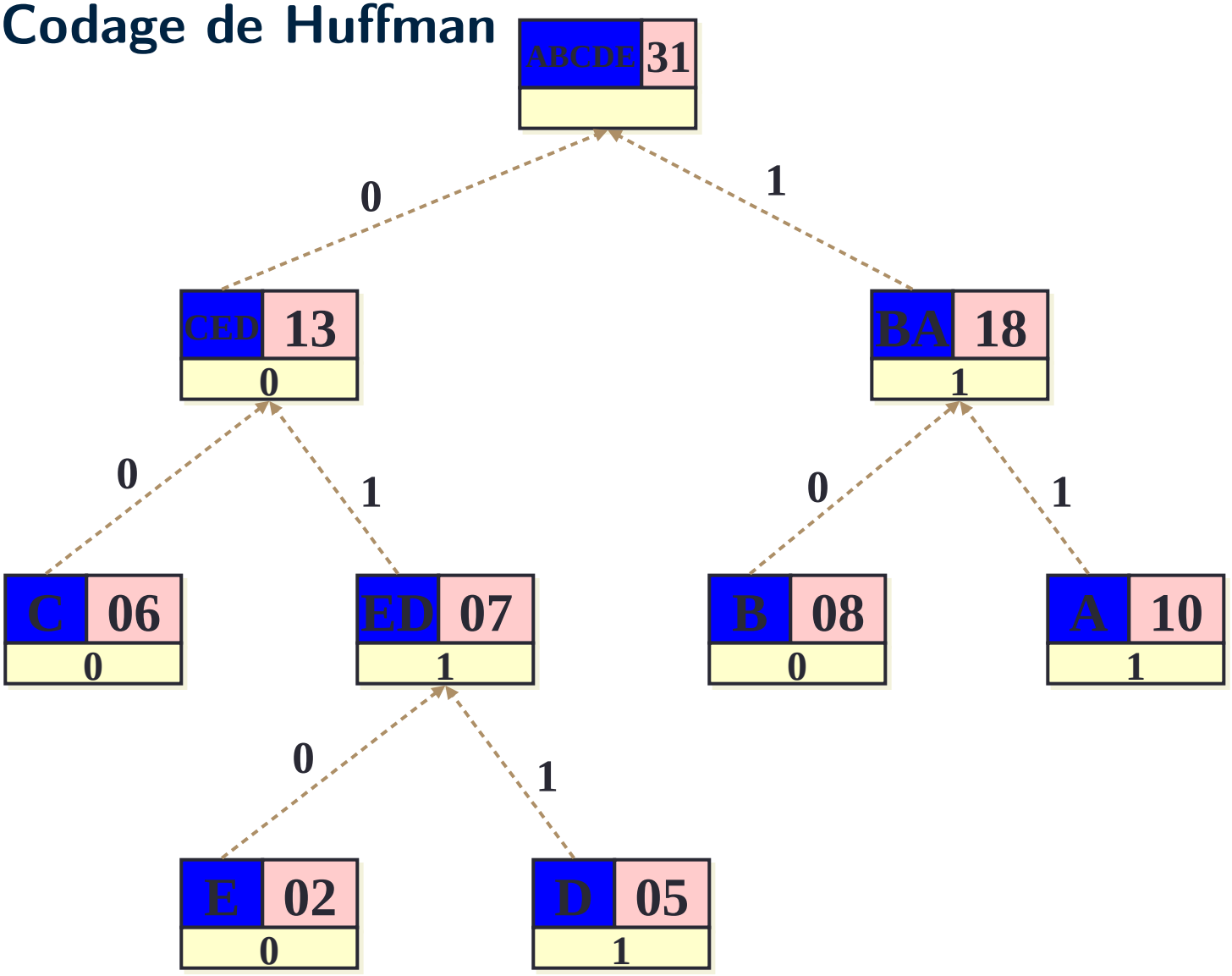
$p_i \rightarrow 1$ $L(C_i)$ faible

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

Codage de Huffman



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

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

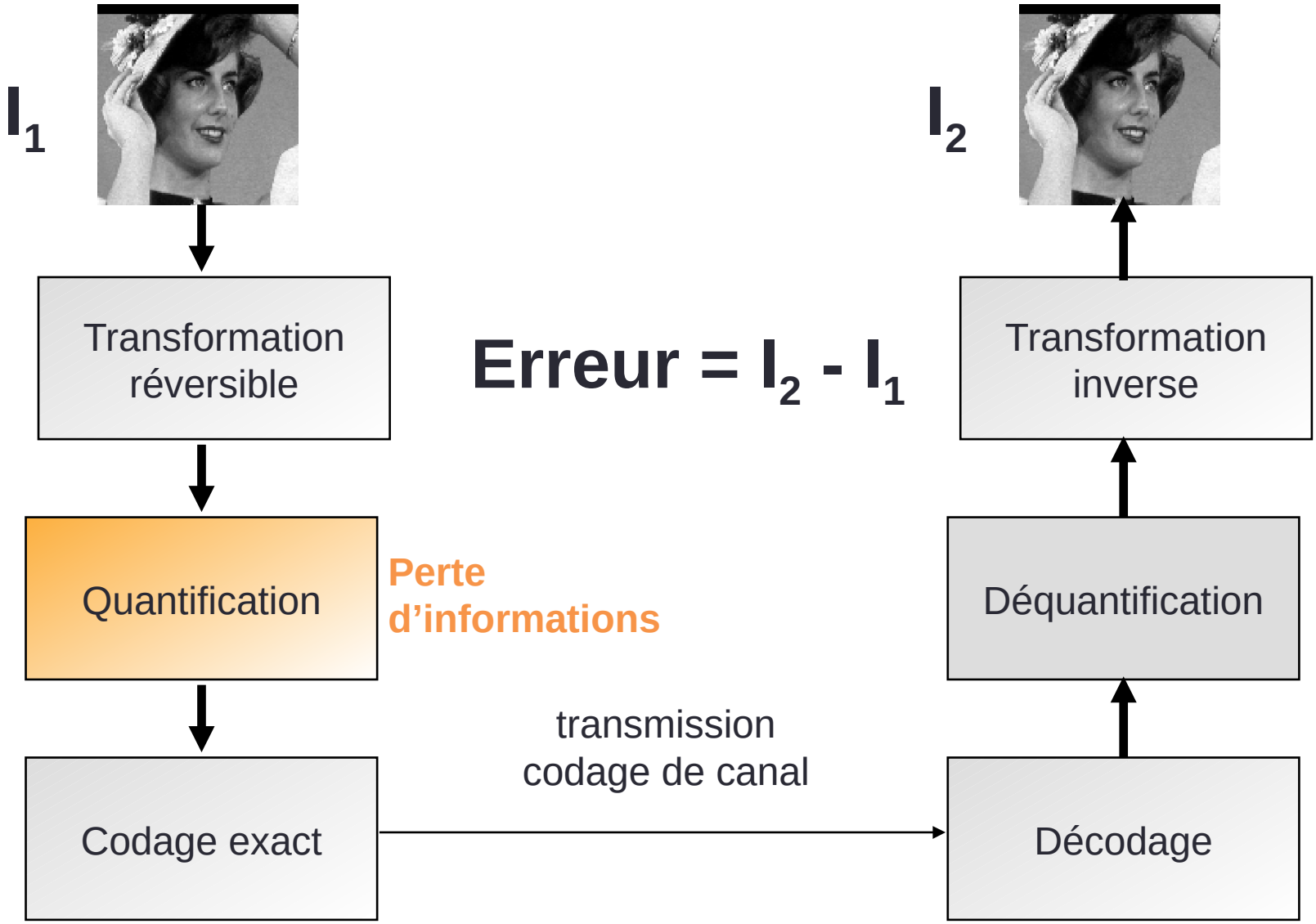
Codage de Huffman

- Analyser le codage de Huffman fourni dans *codage_huffman.m.*
- Finaliser la reconstruction des codes associés aux symboles.
- Calculer la longueur moyenne des codes.
- Comparer cette longueur à l'entropie de la source.

  A-t-on bien approché l'entropie ?

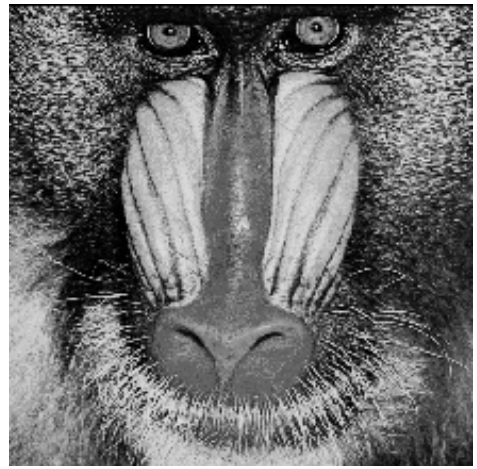
- Appliquer l'algorithme aux images A et D de l'exercice précédent et faire les mêmes observations.

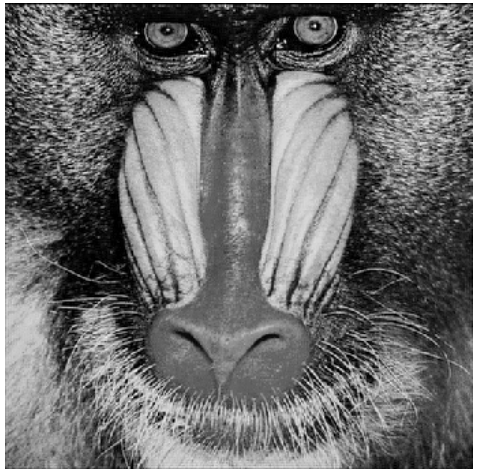
Compression/Décompression

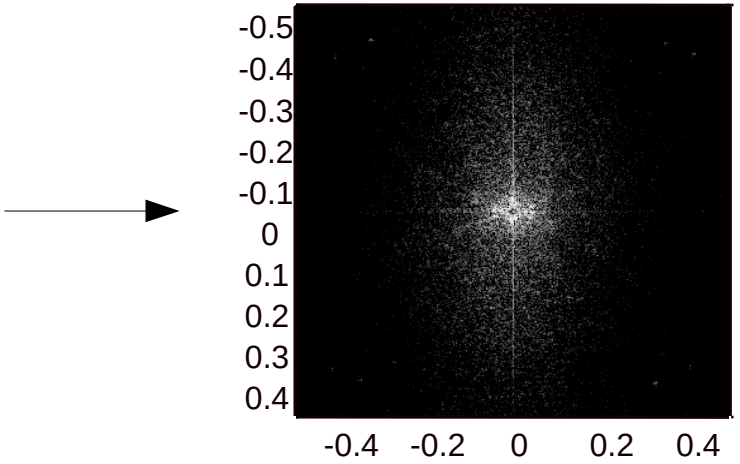


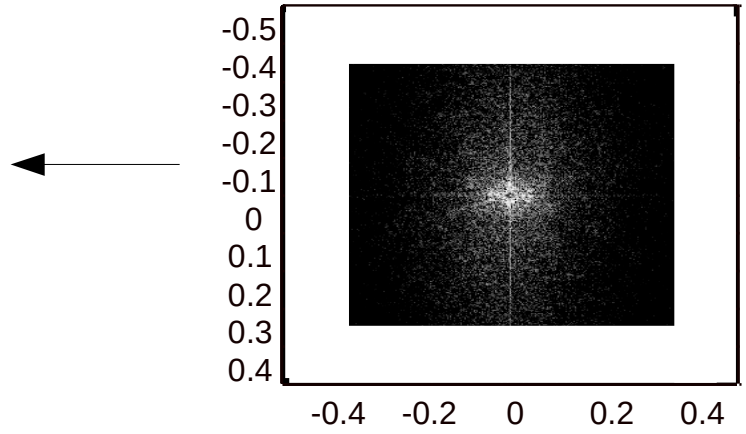
Principe psychovisuel

- La perception est peu sensible aux composantes hautes fréquences









Mise à zéro des coefficients haute fréquence

JPEG

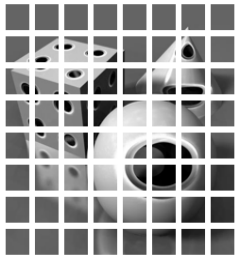
Joint Photographic Expert Group

Norme ISO/IEC CCITT T81

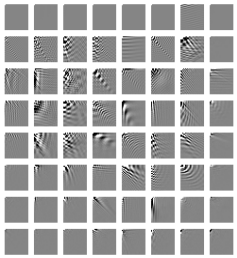
Principe

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

blocs



coefficients DCT



Paramètres :
matrice et facteur de qualité

table de Huffman

Découpage en blocs 8x8

DCT 2D

Quantification

Codage entropique sans pertes

flux binaire (fichier .jpg)

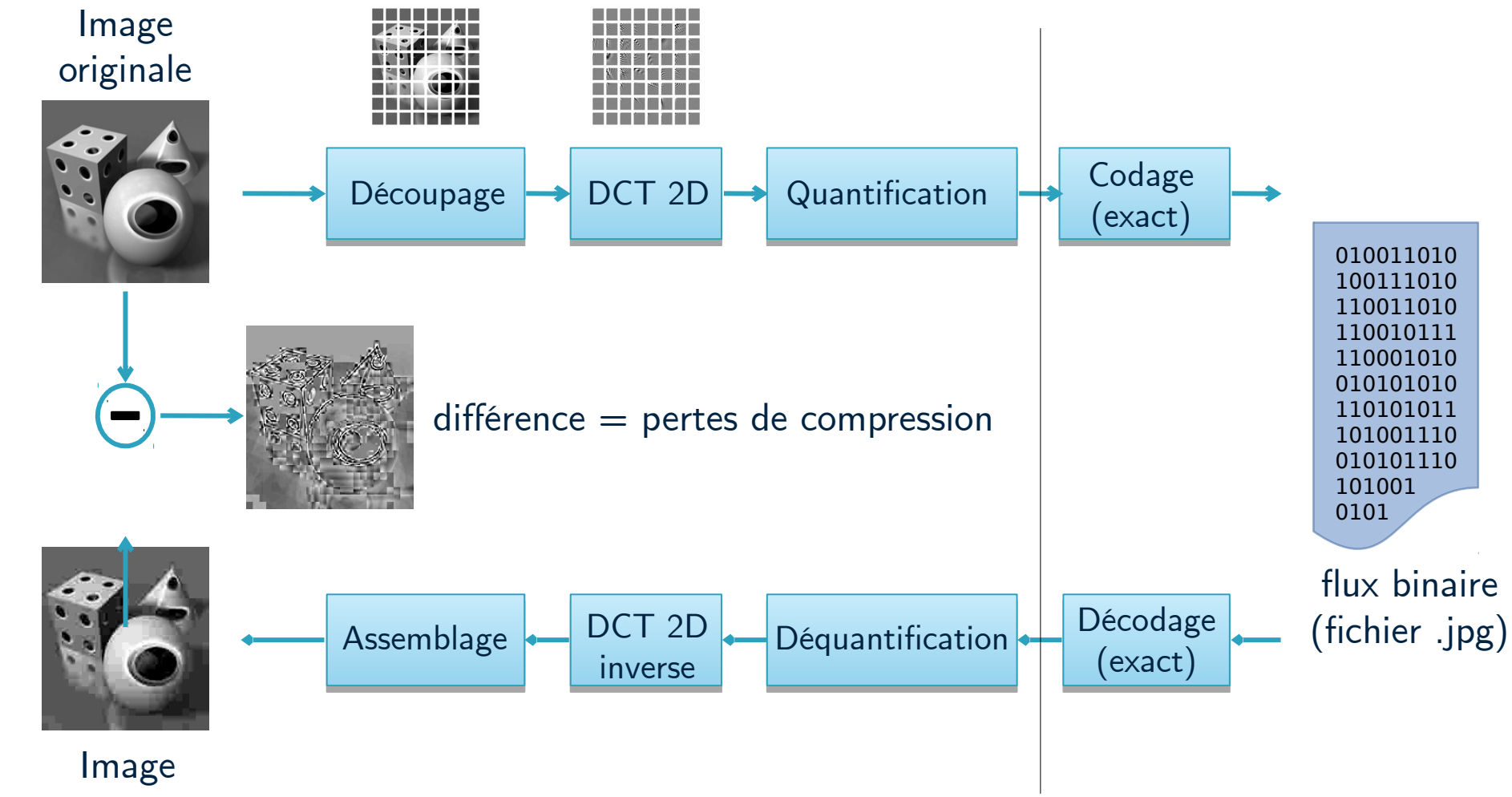
01001101
01001110
10110011
01011001
01111100
01010010
10101011
01010111
01001110
01010111
0101001
0101

changement d'espace (fréquence)

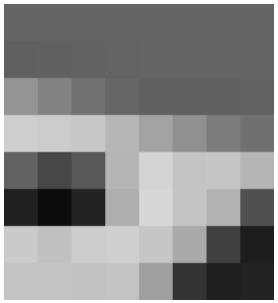
« simplification » par mise à zéro de coefficients (davantage les hautes fréquences)

compression

Chaîne complète



Changement d'espace et « simplification »



Bloc 8x8

94	94	94	94	94	94	94	94
90	91	92	93	94	94	94	94
143	124	107	96	90	90	91	92
203	201	196	179	158	13		106
92	65	82	176	20			177
28	10	31	17		2	175	74
199	188	202	20		165	58	25
192	193	190	192	154	47	27	31

spatial

f

Transformée en cosinus (DCT)

1 coefficient DC

63 coefficients AC

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

fréquentiel

F

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		
1	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

fréquentiel

F_Q

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

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

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

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

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 de Fourier

$$F(k) = \sum_{n=0}^{N-1} f(n) e^{-j 2\pi k \frac{n}{N}}$$

← base de cosinus ← base d'exponentielles

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

- 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

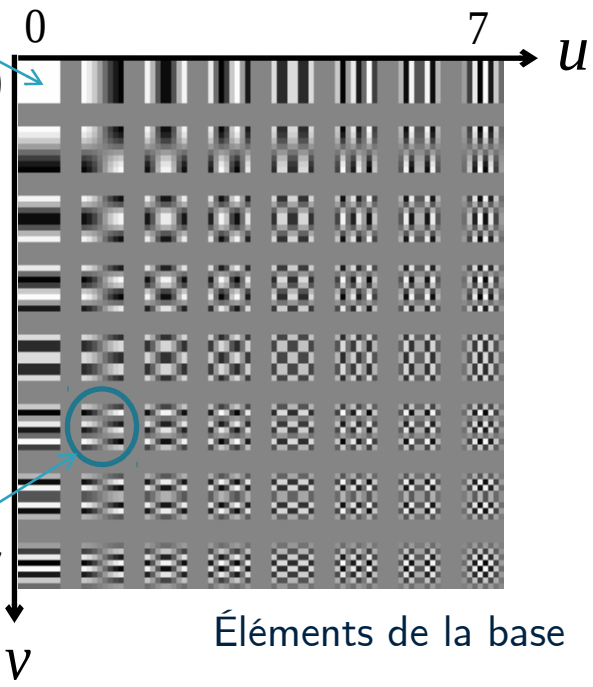
$$F(u,v) \propto \sum_{m=0}^{7}\sum_{n=0}^{7} f(m,n)\cos\left(\pi u \frac{2m+1}{16}\right)\cos\left(\pi v \frac{2n+1}{16}\right)$$

coefficients DCT

bloc 8x8

Base de décomposition

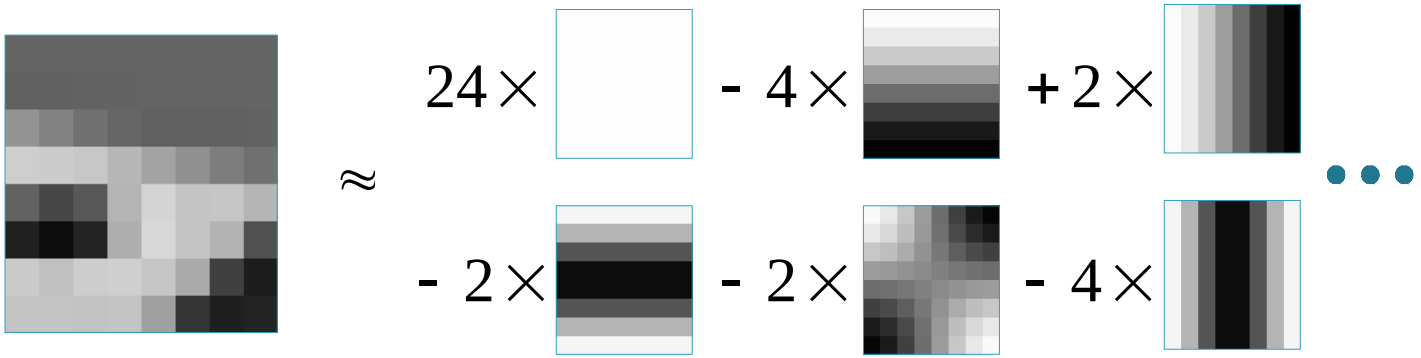
fréquence nulle (moyenne)



$u = 1, v = 5$

Éléments de la base

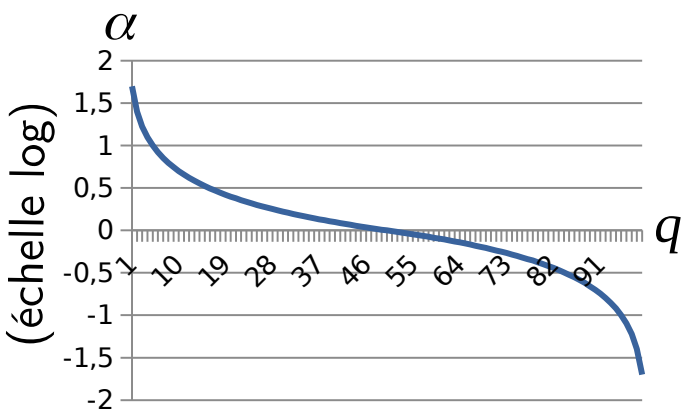
→ Décomposition approchée en une somme de motifs fréquentiels



Phase de quantification

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

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



(facteur de qualité)

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

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)

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

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

Image initiale

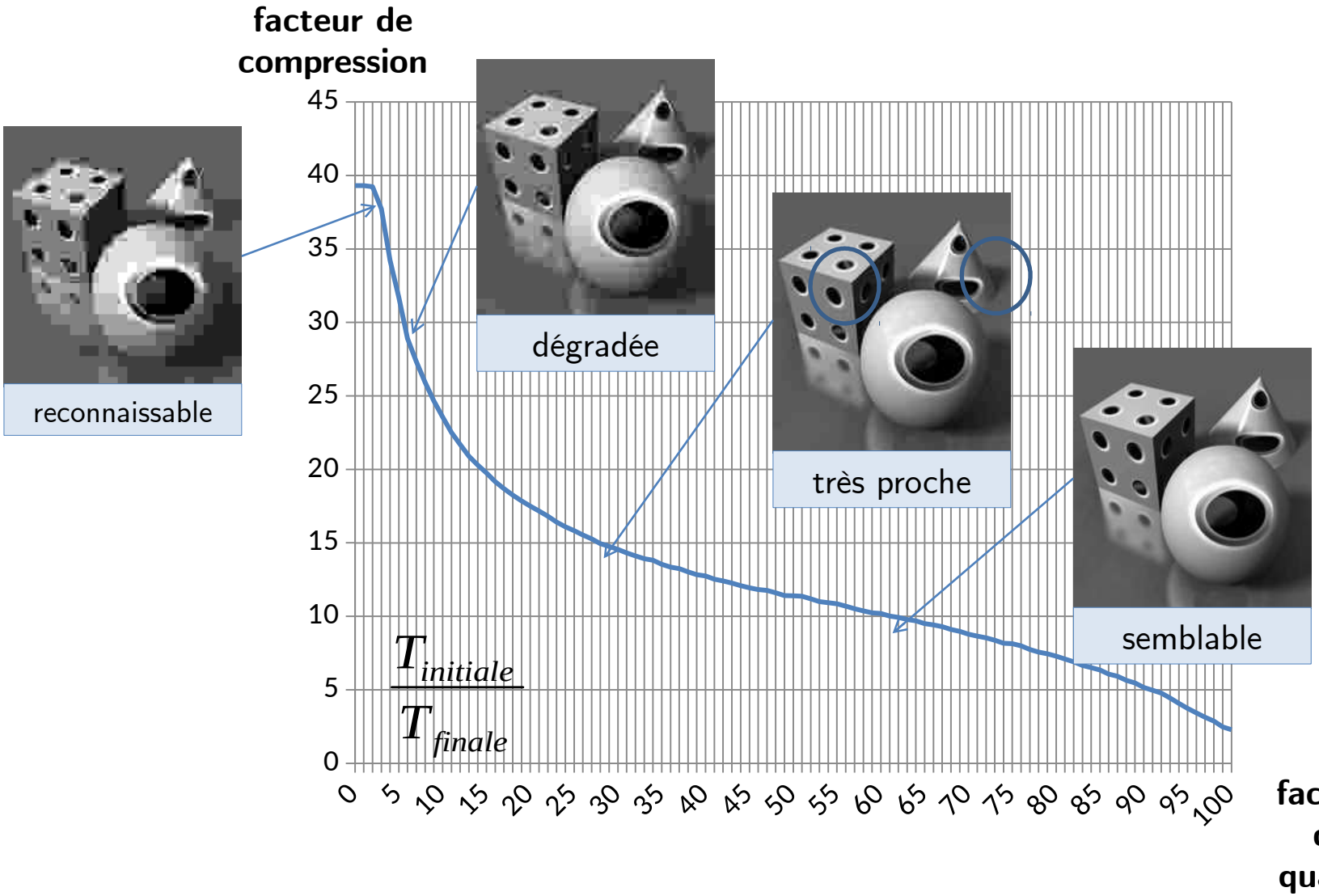
PSNR = 24.85

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) Recentrer l'image par un gain de -128

3) Appliquer la DCT (*dct(dct(...))*)

4) Appliquer la quantification (*fix* pour la troncature)

5) Réaliser la reconstruction : déquantification, DCT inverse (`idct(idct(...))`) et recentrage

6) Appliquer l'algorithme à tous les blocs de l'image (*blockproc*)

7) 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
```

8) Comparer la qualité de manière qualitative en calculant le PSNR entre l'image initiale et l'image compressée