

Algorithmique des images, textes et données

1. Exemple de compression avec perte : le format JPEG

David Parello (Vincent Zucca)

david.parello@univ-perp.fr

Université de Perpignan Via Domitia

S4 Licence 2023-2024



Représentation des données multimédia

- ▶ Discrétisation
- ▶ Stockage des images
- ▶ Limites des algorithmes de compression sans perte

Compression de données multimédia

- ▶ Perception humaine du son
- ▶ Perception humaine des images

Outils pour la compression d'images

- ▶ Transformée en cosinus discrète (TCD)
- ▶ Espaces colorimétriques

Compression des images : le format JPEG

- ▶ Transformation de couleurs et sous-échantillonage
- ▶ DCT et Quantification
- ▶ Encodage

Représentation des données multimédia

- ▶ Discrétisation
- ▶ Stockage des images
- ▶ Limites des algorithmes de compression sans perte

Compression de données multimédia

- ▶ Perception humaine du son
- ▶ Perception humaine des images

Outils pour la compression d'images

- ▶ Transformée en cosinus discrète (TCD)
- ▶ Espaces colorimétriques

Compression des images : le format JPEG

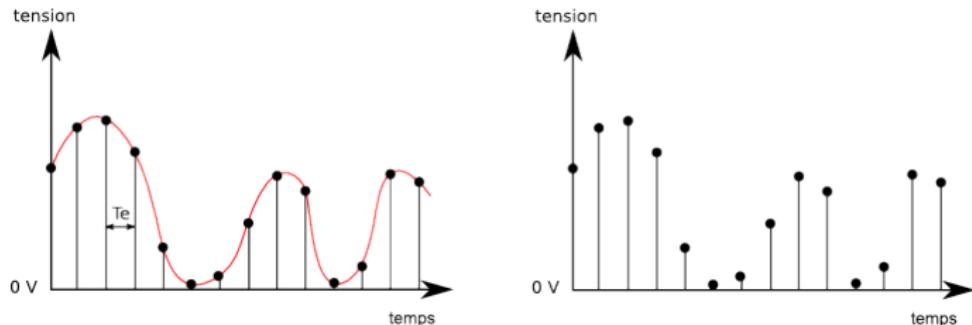
- ▶ Transformation de couleurs et sous-échantillonage
- ▶ DCT et Quantification
- ▶ Encodage

Discrétisation

- Les données multimédia de types son ou image sont des signaux continus.
- Pour pouvoir être retranscrits numériquement, ils doivent être discrétisés, c'est à dire représentés par un nombre fini de valeurs.
- Le nombre de valeurs avec lesquels ces signaux sont représentés va définir la qualité de la représentation et le type de ces valeurs diffèrent selon le média considéré :
 - ▶ Pour le son on parle de **fréquence d'échantillonage**, c'est à dire le nombre de valeurs enregistrées par seconde.
 - ▶ Pour les images on parle de **RÉSOLUTION**, c'est à dire le nombre de pixels.
- Plus l'échantillonage sera important, plus le fichier qui représente le signal sera volumineux.

Discrétisation

- Un son numérique est un signal continu discréteisé.



- Une image est un signal de deux variables x et y : $\text{pixel}(x, y)$.



- Les valeurs sonores proches temporellement et les pixels proches spatialement diffèrent peu en général.

Stockage des images

- Sauf rares exceptions les images sont représentées par un pavage rectangulaire de pixels.
- Les fichiers qui vont stocker les images doivent pouvoir stocker différentes informations :
 - ▶ largeur, hauteur de l'image
 - ▶ valeur de chaque pixel
 - ▶ type de données
 - ▶ auteur
 - ▶ date
- En pratique on va stocker les informations concernant l'image dans l'en-tête du fichier et ensuite on stockera les données.
- Nous allons commencer par étudier les formats PBM, PGM et PPM qui permettent de stocker des images de façon simple mais qui sont bien moins performant en terme de compression que les formats JPEG, GIF ou PNG.

Stockage des images : le format PBM

- Le format PBM (Portable Bit Map) permet de stocker des images en noir et blanc
→ les pixels ne peuvent prendre que deux valeurs 0 ou 1.
- En ouvrant une image enregistrée au format PBM en ASCII avec un éditeur de texte simple (bloc note ou gedit) on observera la structure suivante

```
P1 <— Format PBM en ASCII
# Une ou plusieurs lignes de commentaires
7 10 <— largeur puis hauteur de l'image
0 0 0 0 0 0
0 0 0 0 0 1 0
0 0 0 0 0 1 0
0 0 0 0 0 1 0
0 0 0 0 0 1 0
0 0 0 0 0 1 0
0 0 0 0 0 1 0
0 0 0 0 0 1 0
0 1 0 0 0 1 0
0 0 1 1 1 0 0
0 0 0 0 0 0 0
```

Stockage des images : le format PGM

- Le format PGM (Portable Gray Map) permet de stocker des images en niveau de gris dont les pixels ont des valeurs comprises entre 0 et MAX (MAX à 65536 en ASCII) selon leur niveau d'intensité.
→ un pixel noir est codé par la valeur 0 et un pixel blanc est codé par la valeur MAX (généralement 255)
- En ouvrant une image enregistrée au format PGM en ASCII avec un éditeur de texte simple (bloc note ou gedit) on observera la structure suivante

```
P2 <-- Format PGM en ASCII
# Une ou plusieurs lignes de commentaires
24 7 <-- largeur puis hauteur de l'image
15 <-- valeur de MAX
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 3 3 3 3 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0 0 0 0 15 0 0 15 0 0 15 0
0 3 3 3 0 0 0 7 7 7 0 0 0 11 11 11 0 0 0 15 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0 0 0 0 15 0 0 0 15 0 0 0 0
0 3 0 0 0 0 0 7 7 7 7 0 0 11 11 11 11 0 0 0 15 0 0 0 15 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
```

Stockage des images : le format PPM

- Le format PPM (Portable Pixel Map) permet de stocker des images en couleur.
- Chaque pixel a pour valeur un triplet (R,G,B) composé d'une composante rouge, une composante verte et d'une composante bleue. Chaque composante est représentée par une valeur comprise entre 0 et MAX (généralement 255).
- En ouvrant une image enregistrée au format PPM en ASCII avec un éditeur de texte simple (bloc note ou gedit) on observera une structure de type :

```
P3 <-- Format PPM en ASCII
# Une ou plusieurs lignes de commentaires
3 2 <-- largeur puis hauteur de l'image
255 <-- valeur de MAX
255 0 0 <-- triplet RGB du pixel en position (0,0) (en haut à gauche)
    0 255 0 <-- triplet RGB du pixel en position (0,1)
    0 0 255 <-- triplet RGB du pixel en position (0,2)
255 255 0 <-- triplet RGB du pixel en position (1,0)
255 255 255 <-- triplet RGB du pixel en position (1,1)
    0 0 0 <-- triplet RGB du pixel en position (1,2)
```

Stockage des images

- Regardons maintenant la taille des fichiers codés en ASCII (sous linux avec la commande `wc -c`)



Format PBM : 266 Ko Format PGM : 901 Ko Format PPM : 2.7 Mo

- Ces images sont volumineuses : une valeur entre 0 et 255 est codée par 3 caractères ASCII soit 3 octets alors qu'elle tient sur 1 octet ($255 = 2^8 - 1$).
- Pour gagner en place on peut coder les valeurs des pixels en binaire (l'en tête du fichier reste généralement en ASCII)

Stockage des images

- En binaire le format PBM a le nombre P4 dans l'en-tête de son fichier, PGM a le nombre P5 et PPM a le nombre P6.
- Une image enregistrée en binaire ne pourra pas être ouverte avec un éditeur de texte standard.
- En C la lecture et écriture de données binaires s'effectuent avec

```
size_t fread(void *data, size_t size, size_t nobj, FILE *stream)  
size_t fwrite(const void *data, size_t size, size_t nobj, FILE *stream)
```

- Attention également à l'ordre d'encodage ! En binaire les données peuvent être encodées dans deux sens différents
 - ▶ little endian : octets de poids faibles stockés aux petites adresses (ex : processeur Intel)
 - ▶ big endian : octets de poids forts stockés aux petites adresses (ex : processeur Motorola, ARM)
- Les formats PBM, PGM et PPM utilisent l'ordre little endian.

Stockage des images



P1

PBM ASCII : 266 Ko

P2

PGM ASCII : 901 Ko

P3

PPM ASCII : 2.7 Mo

P4

PBM binaire : 33 Ko

P5

PGM binaire : 262 Ko

P6

PPM binaire : 786 Ko

Limites des algorithmes de compression sans perte

- Les formats PBM, PGM et PPM encodent donc des images sans aucune forme de compression.
 - ▶ pas de perte de données
 - ▶ aucun algorithme de codage ou décodage pour la lecture ou l'écriture
- D'autres formats utilisent des algorithmes de compression sans pertes
 - ▶ GIF : LZW en réduisant le nombre de couleurs à 256.
 - ▶ PNG : utilise l'algorithme DEFLATE (LZ77 + Huffman)
- Malheureusement les algorithmes de compression sans perte ne sont pas ou peu adaptés à la compression de données multimédia :
 - ▶ RLE est assez efficaces pour les images binaires (blanc/noir) mais perd en efficacité lorsque le nombre de symboles (couleurs) augmente.
 - ▶ L'efficacité des algorithmes à dictionnaire type LZ dépend de la présence de motifs répétitifs (peu probable pour une image)
 - ▶ Les méthodes statistiques comme Huffman donnent de bons résultats lorsque les symboles à compresser possèdent des probabilités d'apparition significativement différentes.

Limites des algorithmes de compression sans perte

- La compression GIF est sans perte si seulement si l'image contient moins de 256 couleurs !



PBM ASCII : 266 Ko	PGM ASCII : 901 Ko	PPM ASCII : 2.7 Mo
PBM binaire : 33 Ko	PGM binaire : 262 Ko	PPM binaire : 786 Ko
GIF : 21 Ko	GIF : 170 Ko	GIF : 184 Ko
PNG : 26 Ko	PNG : 156 Ko	PNG : 436 Ko

- En revanche si l'on accepte de perdre des données on peut obtenir des taux de compression beaucoup plus intéressants (exemple JPEG)

Limites des algorithmes de compression sans perte



PBM ASCII : 266 Ko

PGM ASCII : 901 Ko

PPM ASCII : 2.7 Mo

PBM binaire : 33 Ko

PGM binaire : 262 Ko

PPM binaire : 786 Ko

GIF : 21 Ko

GIF : 170 Ko

GIF : 184 Ko

PNG : 26 Ko

PNG : 156 Ko

PNG : 436 Ko

JPEG (90%) : 132 Ko

JPEG (90%) : 63 Ko

JPEG (90%) : 102 Ko

JPEG (50%) : 77 Ko

JPEG (50%) : 23 Ko

JPEG (50%) : 34 Ko

Représentation des données multimédia

- ▶ Discrétisation
- ▶ Stockage des images
- ▶ Limites des algorithmes de compression sans perte

Compression de données multimédia

- ▶ Perception humaine du son
- ▶ Perception humaine des images

Outils pour la compression d'images

- ▶ Transformée en cosinus discrète (TCD)
- ▶ Espaces colorimétriques

Compression des images : le format JPEG

- ▶ Transformation de couleurs et sous-échantillonage
- ▶ DCT et Quantification
- ▶ Encodage

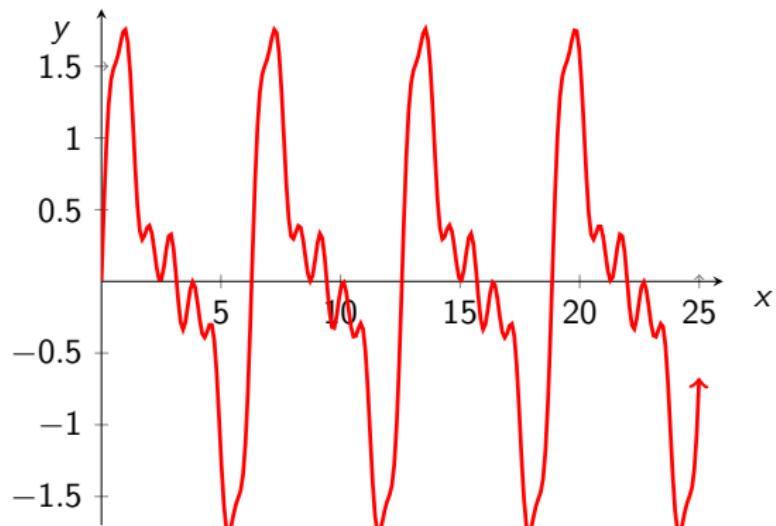
Perception humaine du son

Le son est une vibration mécanique d'un fluide qui se déplace grâce à la déformation élastique de ce fluide.

La vitesse de l'onde sonore détermine sa fréquence. En pratique, les sons que l'on entend sont la superposition de plusieurs ondes sonores.

Exemple :

→ ?

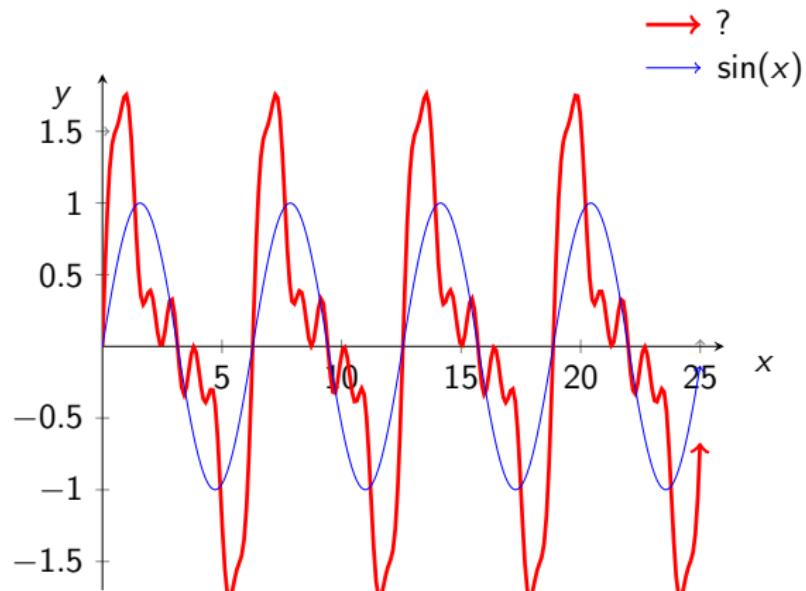


Perception humaine du son

Le son est une vibration mécanique d'un fluide qui se déplace grâce à la déformation élastique de ce fluide.

La vitesse de l'onde sonore détermine sa fréquence. En pratique, les sons que l'on entend sont la superposition de plusieurs ondes sonores.

Exemple :

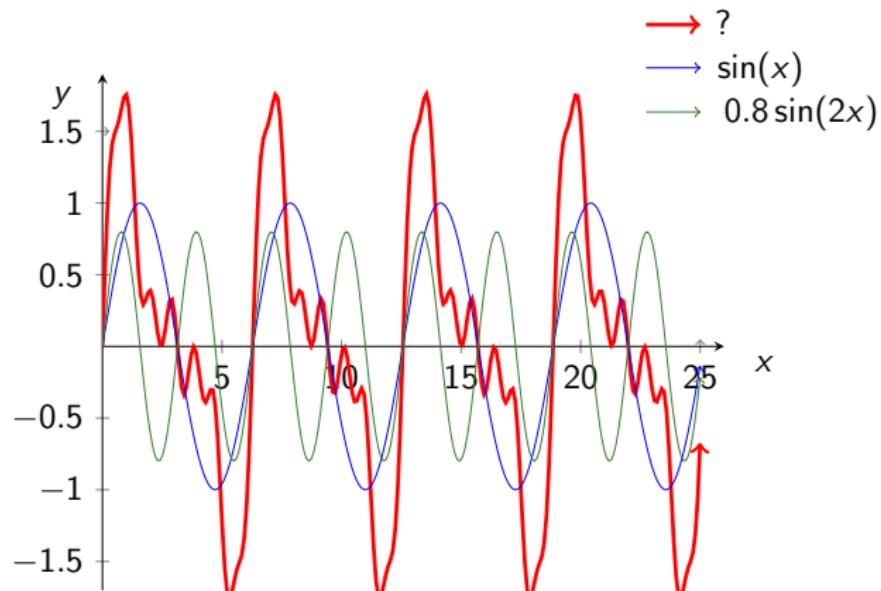


Perception humaine du son

Le son est une vibration mécanique d'un fluide qui se déplace grâce à la déformation élastique de ce fluide.

La vitesse de l'onde sonore détermine sa fréquence. En pratique, les sons que l'on entend sont la superposition de plusieurs ondes sonores.

Exemple :

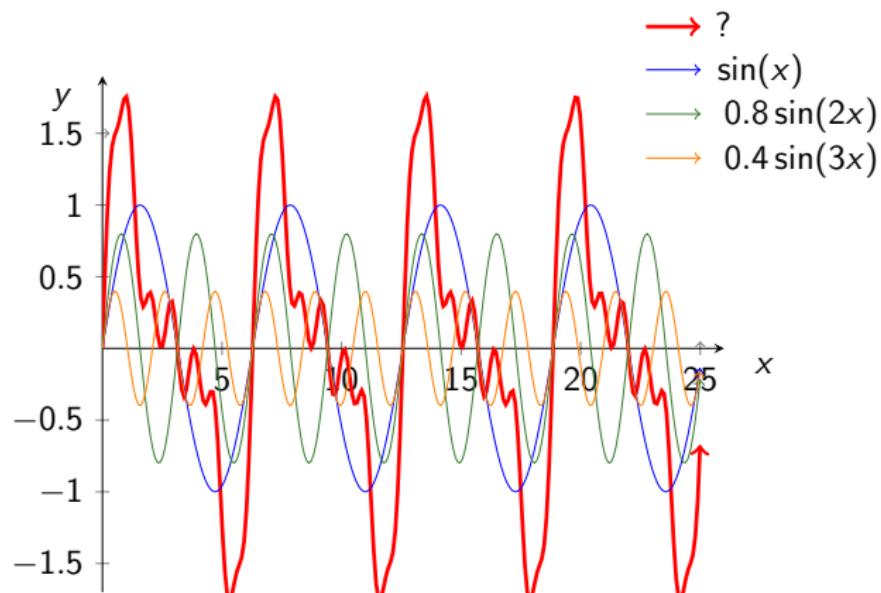


Perception humaine du son

Le son est une vibration mécanique d'un fluide qui se déplace grâce à la déformation élastique de ce fluide.

La vitesse de l'onde sonore détermine sa fréquence. En pratique, les sons que l'on entend sont la superposition de plusieurs ondes sonores.

Exemple :

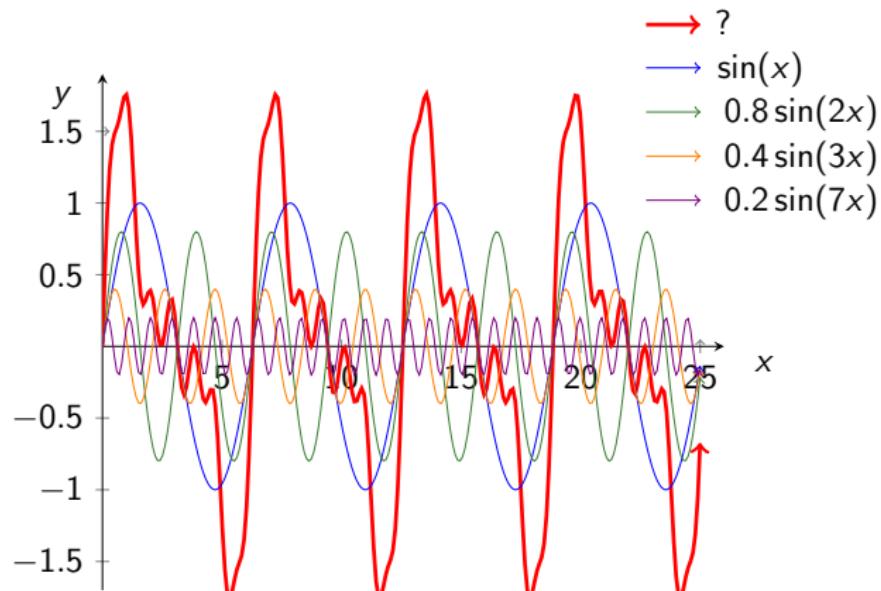


Perception humaine du son

Le son est une vibration mécanique d'un fluide qui se déplace grâce à la déformation élastique de ce fluide.

La vitesse de l'onde sonore détermine sa fréquence. En pratique, les sons que l'on entend sont la superposition de plusieurs ondes sonores.

Exemple :



Perception humaine du son

Cette superposition est traduite mathématiquement par la décomposition en série de Fourier des fonctions périodiques.

Décomposition en série de Fourier : soit $f : I \rightarrow \mathbb{R}$ une fonction T périodique de classe \mathcal{C}^1 sur un intervalle I , i.e. $f(x + T) = f(x)$ pour tout $x \in I$, alors :

$$f(x) = \sum_{n=0}^{+\infty} \left(a_n(f) \cdot \cos \left(\frac{2\pi n x}{T} \right) + b_n(f) \cdot \sin \left(\frac{2\pi n x}{T} \right) \right),$$

où les coefficients a_n et b_n sont appelés les coefficients de Fourier de f .

Cela signifie qu'en ajoutant des sinusoïdes et en ajustant leur amplitudes, fréquences et phase il est possible d'approximer n'importe quel signal périodique.

Rappel : la fréquence d'une fonction périodique correspond à l'inverse de sa période T .

Perception humaine du son

Principe de la compression de signal sonore :

Il est assez connu que pour une onde sonore :

- les hautes fréquences correspondent aux sons aigus ;
- les basses fréquences correspondent aux sons graves.

Cependant, l'oreille humaine est moins sensible aux basses fréquences.

→ on peut perdre ou approximer les composantes de basse fréquence, c'est à dire les premiers coefficients de Fourier.

Perception humaine des images

Pour le son, les fréquences sont définies à partir des variations du signal au cours du temps.

De façon analogue, on peut considérer les variations spatiales du signal le long des lignes et des colonnes d'une image.

- Les hautes fréquences spatiales correspondent à un niveau de gris qui varie fortement sur quelques pixels, par exemple, dans une zone de l'image très texturée.
- Les basses fréquences spatiales correspondent à une variation lente du niveau de gris, par exemple dans une zone de dégradé.

De même que pour le son, on peut analyser une image pour connaître les fréquences que contient l'image.

Perception humaine des images

Propriété : Toute image numérique I de taille M lignes et N colonnes peut être obtenue en additionnant $M \times N$ images sinusoïdales de différentes fréquences.

$$I(x, y) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} a(x, y) \cos \left(2\pi \left(\frac{ix}{M} + \frac{jy}{N} \right) \right) + b(x, y) \sin \left(2\pi \left(\frac{ix}{M} + \frac{jy}{N} \right) \right)$$

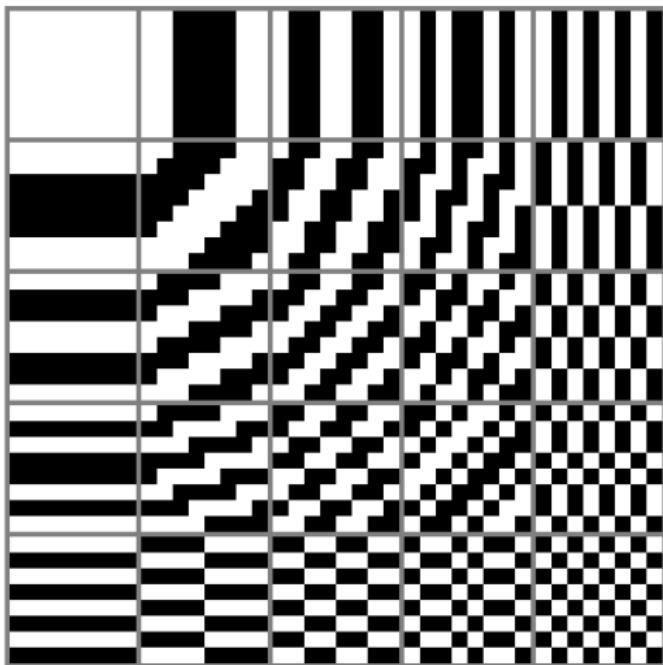
- La ligne d'indice i est une sinusoïde de période M/i pixels.
- La colonne d'indice j est une sinusoïde de période N/j pixels.

Exemple :



Images sinusoïdales classées par ordre croissant sur leur fréquence.

Perception humaine des images



Grille de 5×5 images avec $N = M = 8$, l'axe horizontale augmente la fréquence spatiale des colonnes et l'axe vertical augmente la fréquence spatiale des lignes.

Perception humaine des images

Principes de la compression d'image :

- L'oeil humain est peu sensible aux hautes fréquences qui décrivent les détails d'une image.
 - on peut perdre ou approximer les composantes de haute fréquence.
- L'oeil humain est plus sensible aux variations de luminosité que de couleurs.
 - on peut approximer plus grossièrement les composantes codant les couleurs.

Perception humaine des images

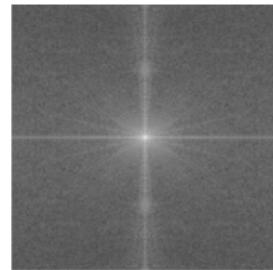
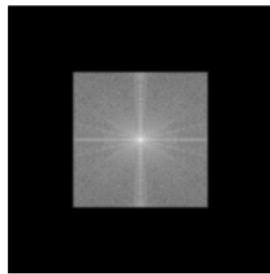


Image naturelle et sa transformée de Fourier



Application d'un filtre passe bas sur la transformée de Fourier et reconstruction de l'image correspondante

Représentation des données multimédia

- ▶ Discrétisation
- ▶ Stockage des images
- ▶ Limites des algorithmes de compression sans perte

Compression de données multimédia

- ▶ Perception humaine du son
- ▶ Perception humaine des images

Outils pour la compression d'images

- ▶ Transformée en cosinus discrète (TCD)
- ▶ Espaces colorimétriques

Compression des images : le format JPEG

- ▶ Transformation de couleurs et sous-échantillonage
- ▶ DCT et Quantification
- ▶ Encodage

Outils pour la compression d'images

Afin de pouvoir compresser efficacement des images, il est nécessaire d'avoir recourt à des transformations permettant :

- d'identifier les redondances de l'image ;
- d'identifier les parties les “moins importantes” de l'image.

En pratique il est préférable que ces transformations soient rapides et faciles à implémenter. C'est pourquoi on utilise généralement des transformations linéaires du type :

$$C = W \cdot D \cdot W^T,$$

avec D la matrice contenant les pixels de l'image d'origine, et C celle contenant les coefficients de la transformée et W la matrice de la transformée.

La nature de la transformation est donc entièrement définie à partir de la matrice W ou des coefficients qui la composent.

Transformée en cosinus discrète (TCD)

De manière globale, si l'on subdivise une image en bloc de $M \times N$ pixels et que M et N sont suffisamment petits les pixels à l'intérieur de chaque bloc vont être très fortement corrélés.

L'idée est donc d'utiliser une transformation permettant d'obtenir de nouvelles variables **décorrélées** les unes des autres, de sorte à ce que l'information importante, dans notre cas les basses fréquences, soit seulement portée par un **petit** nombre de ces nouvelles variables.

Théoriquement, la décorrération optimale est obtenue à l'aide de l'analyse en composantes principales également appelée transformation de Karhunen-Loève.

Malheureusement cette transformation n'est pas linéaire et dépend de l'image à traiter ce qui la rend beaucoup trop coûteuse en calcul.

Transformée en cosinus discrète (TCD)

En pratique on préfère donc utiliser la **Transformée en Cosinus Discrète** (TCD) qui est la transformation linéaire qui se rapproche le plus de celle de Karhunen-Loève.

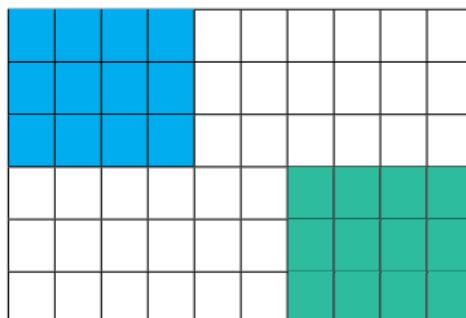
La TCD d'un bloc de $M \times N$ pixels est calculée comme :

$$\text{TCD}(i,j) = \frac{2}{\sqrt{MN}} C(i)C(j) \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \text{pixel}(x,y) \cos\left(\frac{(2x+1)i\pi}{2M}\right) \cos\left(\frac{(2y+1)j\pi}{2N}\right)$$

avec $C(0) = 1/\sqrt{2}$ et $C(i) = 1$ pour $i > 0$.

La TCD regroupe les **basses** fréquences dans les **premiers** coefficients :

basses fréquences →



← hautes fréquences

Transformée en cosinus discrète (TCD)

La TCD est une transformation linéaire **inversible** elle ne cause donc pas de perte d'information en théorie. Sa transformée inverse est donnée par :

$$\text{pixel}(i,j) = \frac{2}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} C(x)C(y)\text{TCD}(x,y) \cos\left(\frac{(2i+1)x\pi}{2M}\right) \cos\left(\frac{(2j+1)y\pi}{2N}\right)$$

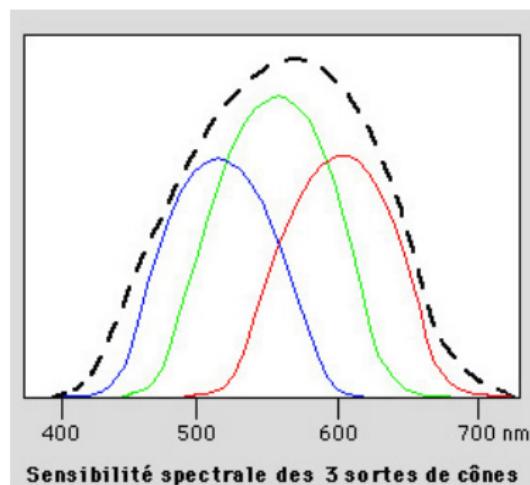
avec C la même constante que précédemment.

Remarque : en pratique le calcul de la TCD cause de petites erreurs d'approximations numériques qui peuvent causer une perte de l'information.

Espaces colorimétriques

Un espace colorimétrique est une représentation numérique des couleurs dans un procédé de synthèse des couleurs.

L'œil humain distingue les couleurs à travers **trois** types de récepteurs photosensibles, appelés **cônes**.



Toute couleur discernable par l'homme est caractérisée par un point dans un espace à **3** dimensions, par exemple l'espace RGB (Red, Green, Blue).

Espaces colorimétriques

Comme l'oeil humain est assez sensible à la luminance mais peu à la chrominance, l'espace RGB n'est pas le plus adapté pour la compression d'image.

Afin d'exploiter cette propriété il faut utiliser un espace type luminance/chrominance comme YUV (aussi appelé Y'CrCb). Dans ce système :

- U, V sont les coordonnées sur le cercle chromatique (chrominance).
- Y sert à éclaircir ou foncer la couleur (luminance).



Espaces colorimétriques

On peut transformer les coordonnées RGB avec $(R, G, B) \in [0, 1]^3$ en $(Y, U, V) \in [0, 1] \times [-0.436, 0.436] \times [-0.615, 0.615]$ avec :

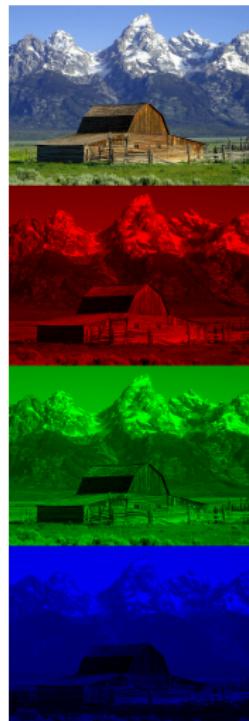
$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & 0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

et la transformation inverse est donnée par :

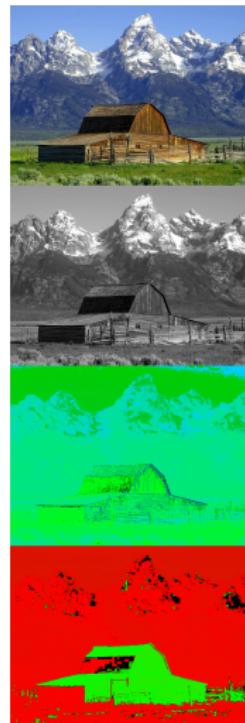
$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1.140 \\ 1 & -0.395 & -0.581 \\ 1 & 2.032 & 0 \end{pmatrix} \cdot \begin{pmatrix} Y \\ U \\ V \end{pmatrix}$$

Espaces colorimétriques

Exemple : décompositions RGB et YUV



Décomposition RGB



Décomposition YUV

Représentation des données multimédia

- ▶ Discrétisation
- ▶ Stockage des images
- ▶ Limites des algorithmes de compression sans perte

Compression de données multimédia

- ▶ Perception humaine du son
- ▶ Perception humaine des images

Outils pour la compression d'images

- ▶ Transformée en cosinus discrète (TCD)
- ▶ Espaces colorimétriques

Compression des images : le format JPEG

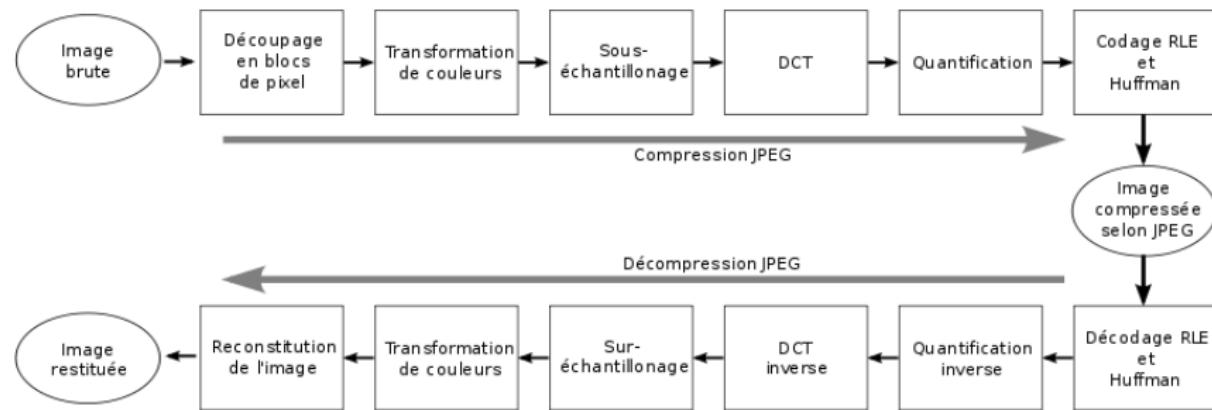
- ▶ Transformation de couleurs et sous-échantillonage
- ▶ DCT et Quantification
- ▶ Encodage

Compression des images : le format JPEG

JPEG (Joint Photographic Experts Group) est une norme qui définit le format d'enregistrement et l'algorithme de **décodage** pour une représentation numérique compressée d'une image fixe.

Les extensions de nom de fichiers les plus communes pour les fichiers employant la compression JPEG sont .jpg et .jpeg.

La compression et décompression se déroulent chacune en 6 étapes.



Transformation de couleurs et sous-échantillonage

L'oeil humain étant plus sensible aux variations de luminance que de chrominance on commence par convertir l'image depuis son espace colorimétrique d'origine (généralement RGB) vers l'espace YUV.

→ Une fois en YUV, on va pouvoir sous-échantillonner les composantes de chrominance (U et V) afin d'exploiter cette propriété.

Le principe de l'opération est de réduire de la taille de plusieurs blocs de chrominance en un seul bloc. Le sous-échantillonnage peut être réalisé selon plusieurs modes différents. Le type de mode utilisé dans une opération de compression est spécifié par la notation "J :a :b".

- J : représente la largeur de la plus petite matrice de pixels considérée (généralement 4)
- a : le nombre de composantes de chrominance dans la première ligne.
- b : le nombre de composantes de chrominance dans la deuxième ligne.

Transformation de couleurs et sous-échantillonage

Exemple :

	4:1:1	4:2:0	4:2:2	4:4:4																																																												
$Y'CrCb$																																																																
=	=	=	=	↪ =																																																												
Y'																																																																
+	+	+	+	+																																																												
(Cr, Cb)	<table border="1"><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>J = 4</td></tr><tr><td>1</td><td>1</td><td>a = 1</td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td>b = 1</td><td></td></tr></table>	1	2	3	4	J = 4	1	1	a = 1			1			b = 1		<table border="1"><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>J = 4</td></tr><tr><td>1</td><td>1</td><td>2</td><td>a = 2</td><td></td></tr><tr><td></td><td></td><td>b = 0</td><td></td><td></td></tr></table>	1	2	3	4	J = 4	1	1	2	a = 2				b = 0			<table border="1"><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>J = 4</td></tr><tr><td>1</td><td>1</td><td>2</td><td>a = 2</td><td></td></tr><tr><td></td><td></td><td>b = 2</td><td></td><td></td></tr></table>	1	2	3	4	J = 4	1	1	2	a = 2				b = 2			<table border="1"><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>J = 4</td></tr><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>a = 4</td></tr><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>b = 4</td></tr></table>	1	2	3	4	J = 4	1	2	3	4	a = 4	1	2	3	4	b = 4
1	2	3	4	J = 4																																																												
1	1	a = 1																																																														
1			b = 1																																																													
1	2	3	4	J = 4																																																												
1	1	2	a = 2																																																													
		b = 0																																																														
1	2	3	4	J = 4																																																												
1	1	2	a = 2																																																													
		b = 2																																																														
1	2	3	4	J = 4																																																												
1	2	3	4	a = 4																																																												
1	2	3	4	b = 4																																																												
$\frac{1}{4}$ horizontal resolution, full vertical resolution	$\frac{1}{2}$ horizontal resolution, $\frac{1}{2}$ vertical resolution	$\frac{1}{2}$ horizontal resolution, full vertical resolution	full horizontal resolution, full vertical resolution																																																													

- Cas 4 :4 :4 : absence d'application du sous-échantillonnage.
- Cas 4 :2 :2 : on calcule la moyenne de chrominance de deux pixels voisins horizontalement on a donc une réduction d'un facteur 1/2.
- Cas 4 :2 :0 : on calcule la moyenne de la chrominance d'un bloc carré de dimension 2×2 . Le sous-échantillonnage 4 :2 :0 ajoute donc une réduction verticale à la réduction horizontale du mode 4 :2 :2.
- Cas 4 :1 :1 : ce dernier cas est rare en compression JPEG car il provoque une perte souvent jugée trop importante. Il est néanmoins parfois utilisé dans certains formats vidéo.

DCT et Quantification

Une fois le sous-échantillonage des composantes U et V effectués on découpe l'image en blocs carrés de $N \times N$ pixels.

Augmenter la taille de N grand permet d'obtenir de meilleurs taux de compression jusqu'à une certaine limite. En effet, si N devient trop grand les pixels du bloc seront moins corrélés et la qualité de la compression s'en ressentira.

Le calcul de chaque coefficient de la DCT requièrent N^2 multiplications, et il y en a N^2 à calculer. Le coût de la décomposition devient vite prohibitif.

→ En pratique il a été choisi d'utiliser $N = 8$ qui est un bon compromis.

On applique donc la DCT à chaque bloc de 8×8 pixels.

Remarque : avec l'augmentation de la puissance de calcul des ordinateurs, on commence à envisager de passer à $N = 16$.

DCT et Quantification

Exemple : le bloc de pixels :

$$\text{pixels} = \begin{bmatrix} 139 & 144 & 149 & 153 & 155 & 155 & 155 & 155 \\ 144 & 151 & 153 & 156 & 159 & 156 & 156 & 156 \\ 150 & 155 & 160 & 163 & 158 & 156 & 156 & 156 \\ 159 & 161 & 162 & 160 & 160 & 159 & 159 & 159 \\ 159 & 160 & 161 & 162 & 162 & 155 & 155 & 155 \\ 161 & 161 & 161 & 161 & 160 & 157 & 157 & 157 \\ 162 & 162 & 161 & 163 & 162 & 157 & 157 & 157 \\ 162 & 162 & 161 & 161 & 163 & 158 & 158 & 158 \end{bmatrix}$$

devient (arrondi à l'entier le plus proche pour l'exemple) :

$$\text{DCT} = \begin{bmatrix} 1260 & -1 & -12 & -5 & 2 & -2 & -3 & 1 \\ -23 & -17 & -6 & -3 & -3 & 0 & 0 & -1 \\ -11 & -9 & -2 & 2 & 0 & -1 & -1 & 0 \\ -7 & -2 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 2 & 0 & -1 & 1 & 1 \\ 2 & 0 & 2 & 0 & -1 & 1 & 1 & -1 \\ -1 & 0 & 0 & -1 & 0 & 2 & 1 & -1 \\ -3 & 2 & -4 & -2 & 2 & 1 & -1 & 0 \end{bmatrix}$$

DCT et Quantification

La DCT étant inversible nous n'avons pas perdu d'information. C'est lors de la **quantification** que le gros de l'information est perdu.

Le but est d'atténuer les hautes fréquences auxquelles l'oeil humain est peu sensible. Chaque coefficient du bloc de DCT est donc divisé par le coefficient correspondant de la matrice de quantification Q :

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

Le résultat est ensuite arrondi à l'entier le plus proche : $\hat{DCT} = \left\lfloor \frac{DCT(i,j)}{Q(i,j)} \right\rfloor$

DCT et Quantification

Exemple : en reprenant l'exemple précédent :

$$\text{DCT} = \begin{bmatrix} 1260 & -1 & -12 & -5 & 2 & -2 & -3 & 1 \\ -23 & -17 & -6 & -3 & -3 & 0 & 0 & -1 \\ -11 & -9 & -2 & 2 & 0 & -1 & -1 & 0 \\ -7 & -2 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 2 & 0 & -1 & 1 & 1 \\ 2 & 0 & 2 & 0 & -1 & 1 & 1 & -1 \\ -1 & 0 & 0 & -1 & 0 & 2 & 1 & -1 \\ -3 & 2 & -4 & -2 & 2 & 1 & -1 & 0 \end{bmatrix}$$

on obtient :

$$\hat{\text{DCT}} = \begin{bmatrix} 79 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

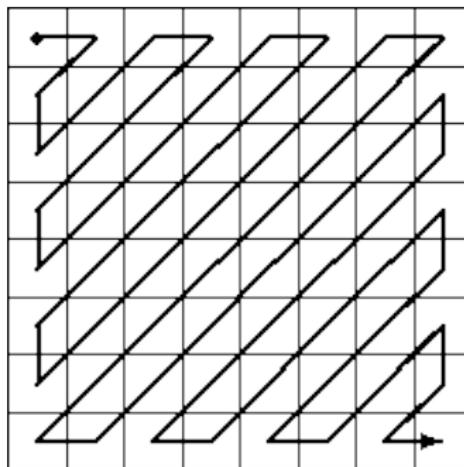
DCT et Quantification

- La quantification ramène beaucoup de coefficients à 0.
 - La longue suite de zéros nécessitera très peu de place !
- Mais si la quantification est trop forte (taux de compression trop élevé), il y aura trop peu de coefficients non nuls pour représenter fidèlement le bloc de 8x8 pixels.
 - la division en blocs devient visible, et l'image apparaît pixelisée.

C'est pourquoi la matrice de quantification a été déterminée de manière **empirique** après avoir effectué de nombreux tests.

Encodage

Afin d'extraire la suite de zéros la plus longue possible on procède à une extraction des coefficients en zigzag :



Les successions de zéros sont ensuite compressées avec un algorithme RLE.

Enfin le résultat (global) est compressé à l'aide d'un codage statistique (codage de Huffman ou codage arithmétique).

Compression des images : le format JPEG

Les étapes de la décompression s'effectuent dans l'ordre inverse de la compression suivant les méthodes définies précédemment. Elles permettent de récupérer une image légèrement altérée.

Par exemple sur l'exemple précédent on récupèrera un bloc de pixels avec une erreur de :

$$e = \begin{bmatrix} -5 & -2 & 0 & 1 & 1 & -1 & -1 & -1 \\ -4 & 1 & 1 & 2 & 3 & 0 & 0 & 0 \\ -5 & -1 & 3 & 5 & 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & -2 & -1 & 0 & 2 & 4 \\ -1 & 0 & 1 & -2 & -1 & 0 & 2 & 4 \\ -2 & -2 & -3 & -3 & -2 & -3 & -1 & 0 \\ 2 & 1 & -1 & 1 & 0 & -4 & -2 & -1 \\ 4 & 3 & 0 & 0 & 1 & -3 & -1 & 0 \end{bmatrix}$$

L'erreur moyenne est d'environ 1% pour un passage de 64 à 10 valeurs.

Globalement, le JPEG permet des taux de compression allant de 3 à 100.