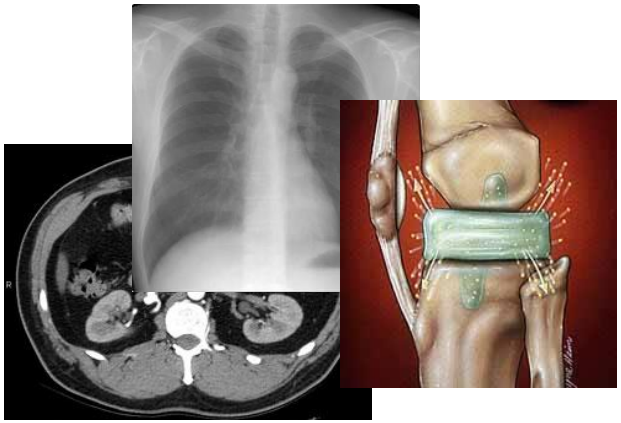
Compression : sans perte et avec perte

Images et filtres
ELEC3

Lionel Fillatre
2018-2019

Omniprésence des images numériques



Objectifs

- Stocker et transférer efficacement les images
- Idéalement :
  - Maximiser la qualité des images
  - Minimiser l'espace de stockage et les ressources nécessaires aux traitements
- Difficile de satisfaire ces deux "idéaux"
- Comment définir un bon compromis ?

Deux types de compression

- Sans perte
  - L'image originale peut être reproduite exactement
  - Nécessite de l'espace
  - Utilise généralement les codages entropiques (Huffman par exemple)
  - Exemples: BMP, TIFF, GIF
- Avec perte
  - L'image originale est reproduite avec quelques dégradations
  - Utilise souvent de la quantification
  - Utilise souvent des transformations de l'image
  - Exemples: JPEG, JPEG-2000

Sommaire

- Compression sans perte
- Compression avec perte

COMPRESSION SANS PERTE

Code à longueur fixe

Principe :

- Supposons que l'on ait un signal :
  - par exemple un fichier de texte, composé de caractères codés sur 8 bits (256 caractères possibles),
  - une image en niveaux de gris, chaque niveau de gris est codé sur 8 bits (256 valeurs possibles).
- Si le nombre de caractères (pixels) du signal est N_c, alors le signal occupera $8*N_c$ bits.

Inconvénient :

- Il est possible de faire mieux que 8 bits à condition d'adopter une longueur de code variable, fonction de l'occurrence du symbole : on a tout intérêt à choisir un code court pour les symboles les plus fréquents, et réserver les longueurs plus importantes aux symboles apparaissant rarement.

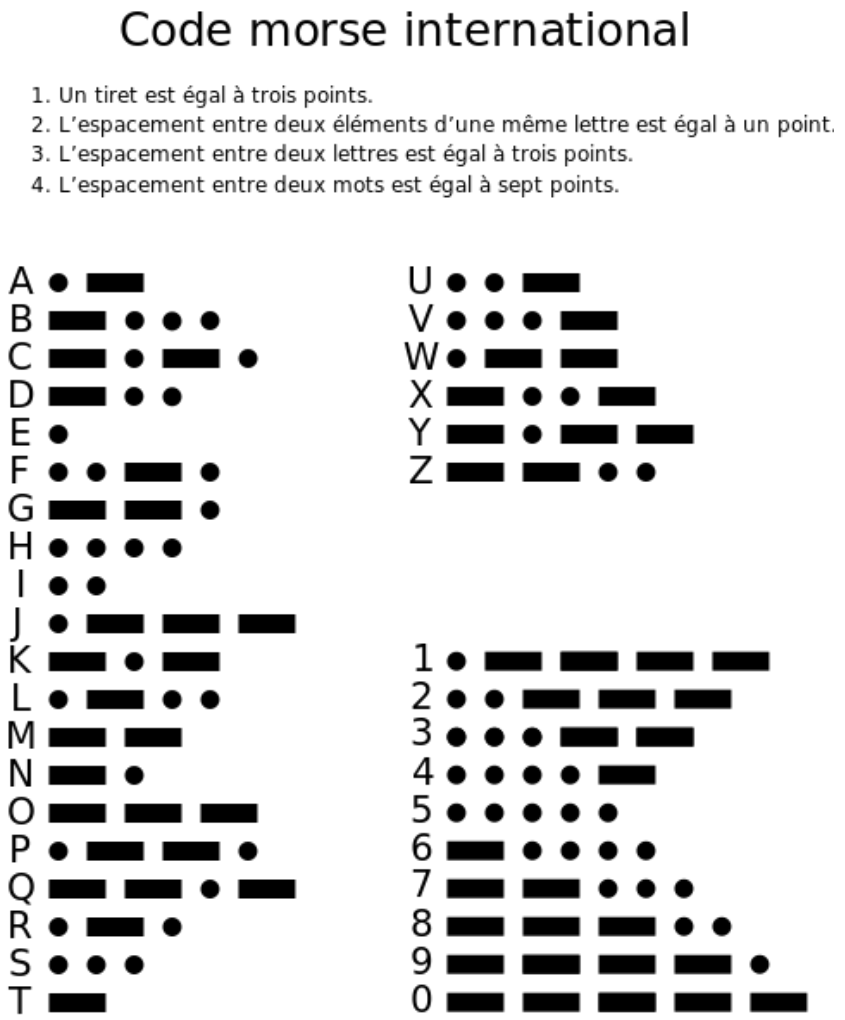
Code à longueur variable

Principe :

- Lorsque tous les mots ont une longueur fixe (8 bits par exemple), le décodage ne pose pas de problème, chaque octet code un symbole. Lorsque les codes sont à longueur variable, il faut distinguer un symbole du suivant.

- Cette opération peut être faite par insertion d'un caractère réservé, ce qui augmente la taille résultante (exemple : espace dans le code morse).

- Une solution alternative consiste à utiliser des **codes préfixés**.
  - Chaque nouveau code ne peut être le début (= préfixe) d'un code déjà existant. Chaque nouveau mot codé devient lui-même préfixe, et ne peut être utilisé.
  - Aucun mot n'est le début d'un autre mot.



Exemple

symbole	Code lg. fixe	VLC1	VLC2
a	000	0	00
b	001	10	01
c	010	110	100
d	011	1110	101
e	100	11110	110
f	101	111110	111
$L_{moyenne}$	3	3,5	2,66

- Dans ce cas équiprobable, le code VLC2 serait meilleur. Ce n'est peut être pas le plus performant si les symboles ne sont plus équiprobables (et peut être que pour une distribution donnée, le code VLC1 serait le plus performant).
- Remarque : les codes préfixés sont des codes « uniquement déchiffrables », c'est-à-dire des codes pour lesquels toute suite de mots ne peut être déchiffrée que d'une seule manière. Exemple : {0, 11, 010} est uniquement déchiffrable mais n'est pas un code préfixé.
- Différents algorithmes permettent de déterminer des codes à longueurs variables pour une source donnée en garantissant ce « préfixage » : le codage de Huffman, le code de Shannon-Fano , etc.

Algorithme de Huffman

- L'algorithme de Huffman est certainement le plus utilisé en raison de sa gratuité, sa simplicité et son efficacité.

- Un grand nombre de normes de compression d'images le préconisent. C'est le cas de JPEG et MPEG en particulier.

- L'algorithme crée des codes de longueurs variables en fonction des probabilités fournies par un modèle, ou la connaissance de la séquence complète.

Algorithme de Huffman

Exemple :

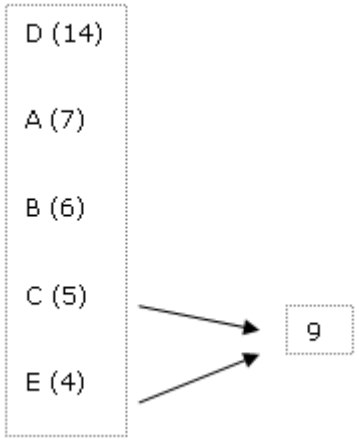
Soit un message de 36 caractères, composé des caractères A, B, C, D et E qui apparaissent selon les fréquences suivantes :

Symboles : A B C D E

Fréquences : 7 6 5 14 4

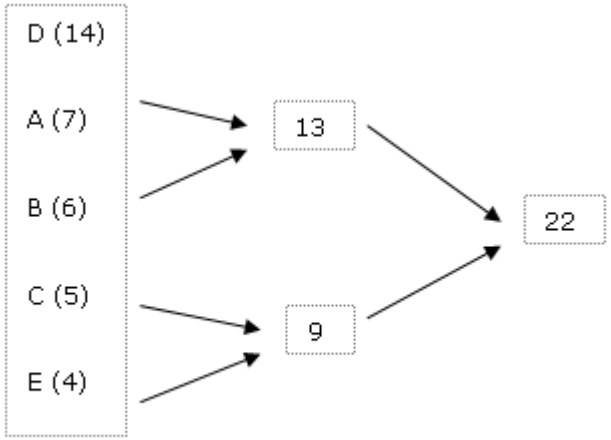
Le graphe ci-dessous fait apparaître les symboles depuis le plus fréquent (D) jusqu'au moins fréquent (E).

Les symboles C et E fusionnent vers un même nœud de poids 5+4=9

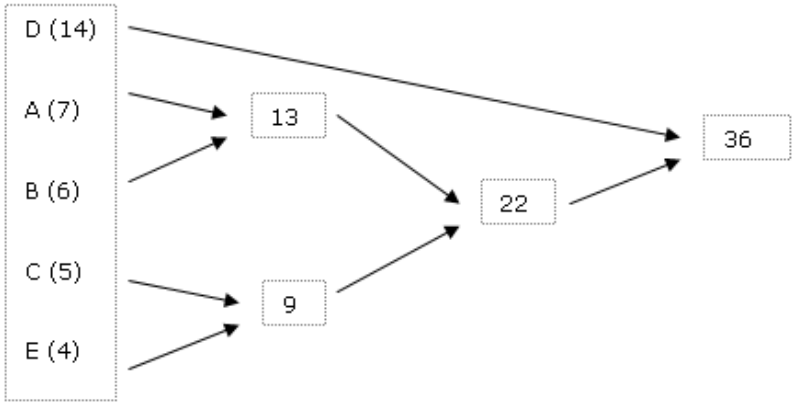


Algorithme de Huffman

Puis les symboles ou nœuds de poids les plus faibles sont à nouveau considérés.
Dans notre exemple, il s'agit ici de 2 symboles : A et B.
Le processus est renouvelé avec le nœud de poids 9 et celui de poids 13.



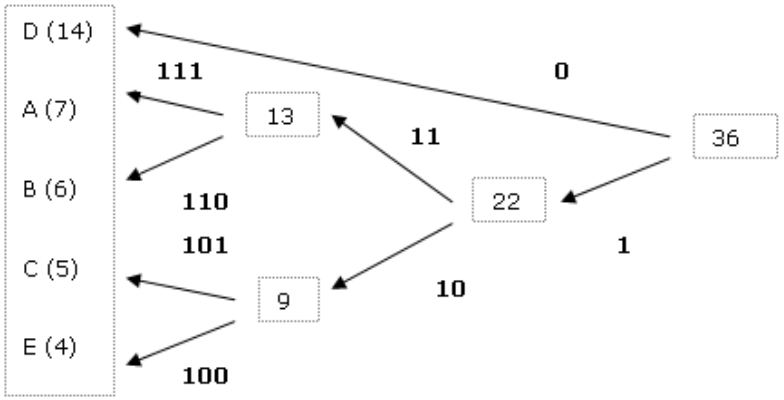
Il reste enfin à fusionner le nœud de poids 22 et le symbole restant, de poids 14.



Algorithme de Huffman

La deuxième phase de l'algorithme consiste à « redescendre » ce graphe binaire, en affectant à la branche conduisant au poids le plus faible, la valeur 0 par exemple, et 1 à l'autre.

Sur l'exemple, on obtient :



Codage obtenu

- Le codage obtenu est donc :

  D (occurrence =14) : 0
  A (occurrence =7) : 111
  B (occurrence =6) : 110
  C (occurrence =5) : 101
  E (occurrence =4) : 100

- Le symbole le plus fréquent a obtenu le code le plus court, les autres ont, dans cet exemple, des longueurs équivalentes.

- Quelle est la longueur moyenne de code ? 1 bit pour D qui apparaît 14 fois dans le message, 3 bits pour les autres qui apparaissent 22 fois en tout.

  Longueur moyenne = (1*14 + 3*7 + 3*6 + 3*5 + 3*4) / 36 =80/36 = 2.22

Taux de compression

- Le taux de compression peut se définir comme le rapport entre la taille du message dans son code d'origine, et la taille du message après re-codage :

  Taux de compression = [Volume final] / [Volume initial]

- On utilise aussi le quotient de compression :

  Quotient de compression = [Volume initial] / [Volume final]

- Remarque : lorsque le nombre de mots à coder est élevé, le volume final est proportionnel à la longueur moyenne des mots du code.

- La longueur moyenne $\bar{L}$ des mots de tout code binaire déchiffrable est bornée inférieurement

  $$\bar{L} \geq H$$

  où H est l'entropie de la source.

Entropie d’une source aléatoire

- Les événements émis par une source ne sont généralement pas équiprobables, la quantité d’information reçue est fonction des événements.

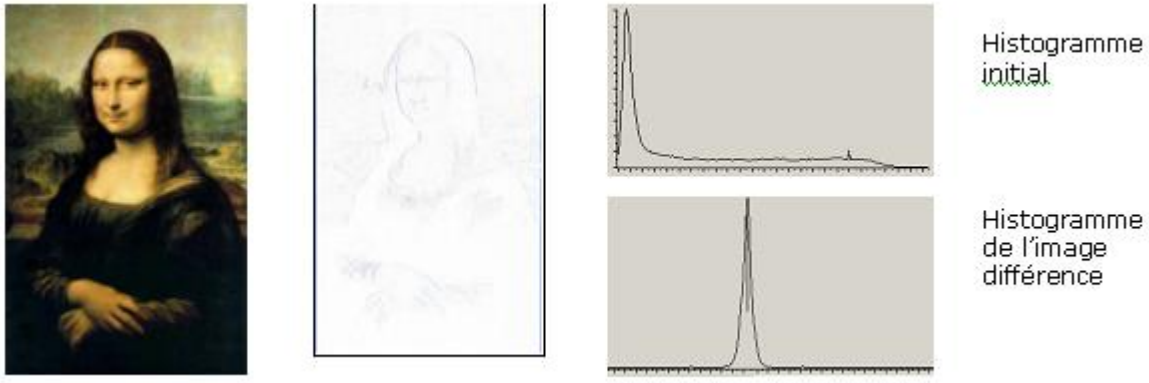
- Pour évaluer la quantité d’informations moyenne reçue, il faut faire une moyenne sur tous les événements possibles : si P_i est la probabilité d’un événement i, la quantité moyenne d’information est :

$$H = -\sum_i P_i \log_2(P_i)$$

- Cette quantité ne dépend que de la source (sans mémoire), c’est son **entropie**.

Application aux images

- Les algorithmes précédents sont-ils adaptés à la compression des images ?
- On peut remarquer que l'hypothèse d'une source sans mémoire n'est pas respectée, une première redondance existe et il faut la diminuer. Il est possible de s'approcher de l'hypothèse d'une source sans mémoire, si sur chaque ligne d'une image, on calcule simplement l'erreur entre le niveau de gris du pixel courant s(x,y) et celui du précédent sur la ligne s(x-1,y) : e(x,y) = s(x,y) – s(x-1,y).



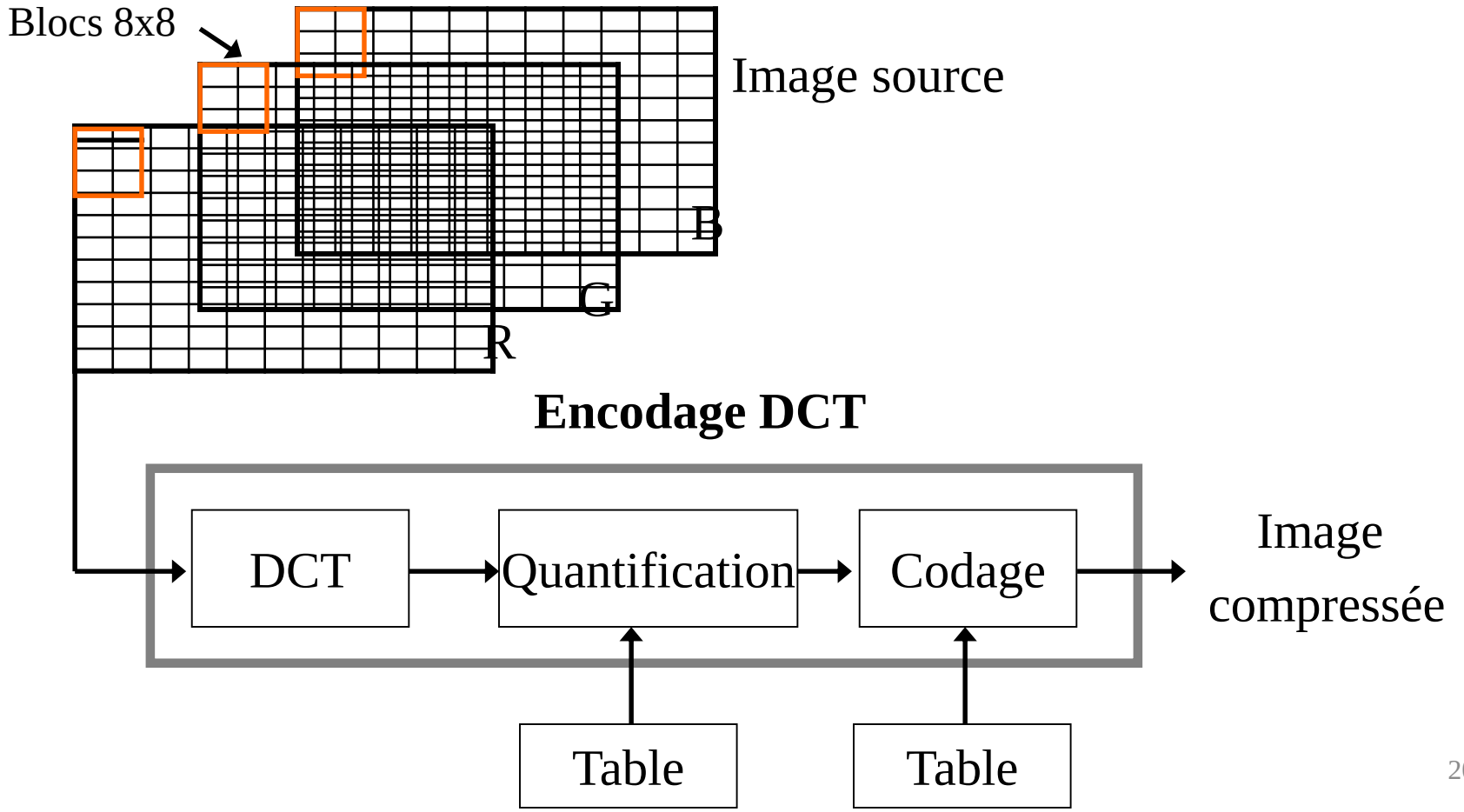
- Dans la pratique, les algorithmes de codage entropique ne sont pas utilisés directement sur les valeurs des pixels de l'image, mais sur des grandeurs obtenues après transformation des valeurs des pixels (comme la DCT par exemple).

COMPRESSION AVEC PERTE

JPEG (Joint Photographic Experts Group)

- Format dominant
- Taille d'environ 10% par rapport au format d'origine non compressé
- Spécialement adapté aux photographies

Diagramme JPEG



ETAPE 1 : Transformation DCT "Discrete Cosine Transform"

- Passage du domaine spatial au domaine fréquentiel
  - L'intensité des pixels est transformée en une somme pondérée de fonctions périodiques (cosinus)
  - Identification des bandes spectrales qui peuvent être éliminées sans trop de perte de qualité
- Les intensités varient souvent de façon modérée dans un même canal de couleur.
- Produit seulement des réels (et non des complexes), plus rapide à calculer
- Possède une qualité naturelle : compacter l'énergie (plus d'informations sont contenues dans moins de coefficients).

Comprendre la DCT

- Dans $\mathbb{R}^3$, (5, 2, 9) peut s'écrire sous la forme d'une somme pondérée de vecteurs de base

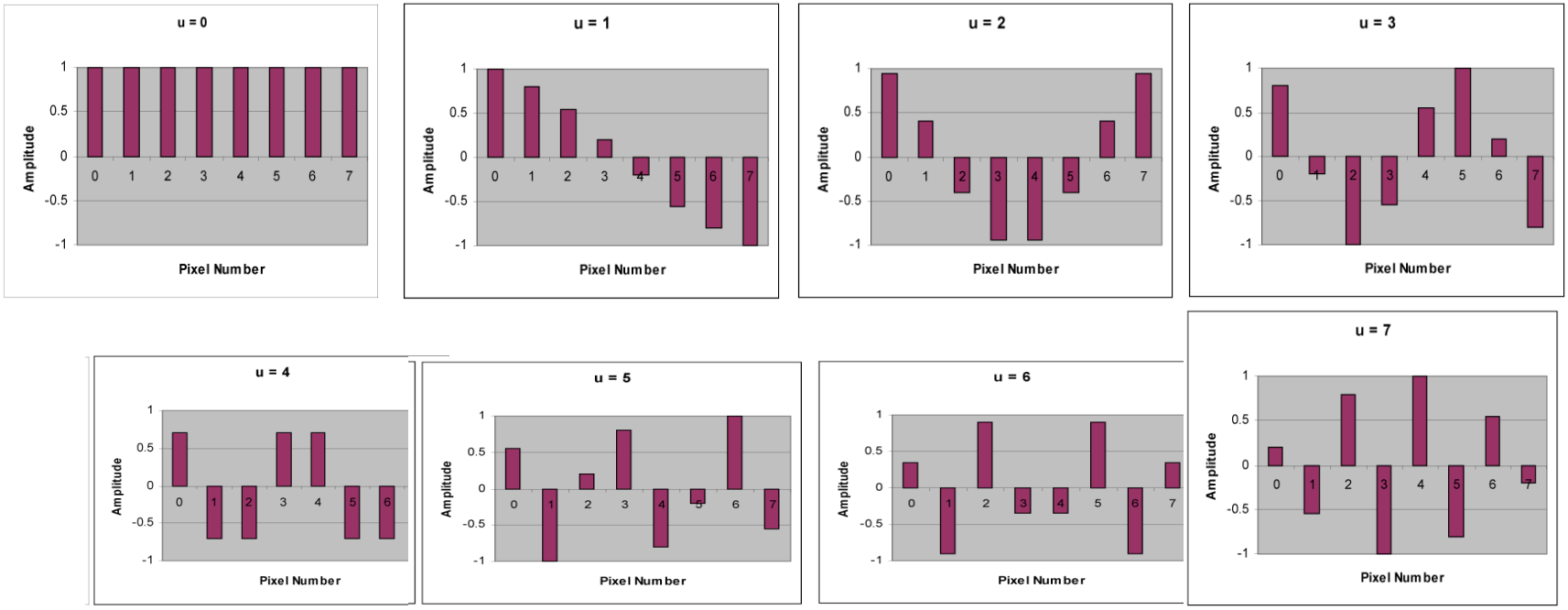
- $\{(1,0,0), (0,1,0), (0,0,1)\}$ est une base de $\mathbb{R}^3$:

$$(5,2,9) = 5 * (1,0,0) + 2 * (0,1,0) + 9 * (0,0,1)$$

- DCT est une "base" dans le domaine des fonctions

Fonctions de base DCT

- Fonction du type “cosinus”



Vision alternative

DCT 1D

- Pour une liste de n valeurs d'intensité $I(x)$, où $x = 0, \dots, n-1$

- Calculer les n coefficients DCT :

$$F(u) = \sqrt{\frac{2}{n}} \; C(u) \sum_{x=0}^{n-1} I(x) \cos \frac{(2x+1)\, u\, \pi}{2n}, u = 0, \dots, n-1$$

où

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{pour} \quad u = 0, \\ 1 & \text{sinon} \end{cases}$$

DCT 1D Inverse

- Pour une liste de n coefficients DCT $F(u)$, où $u = 0, \dots, n-1$

- Calculer les n valeurs:

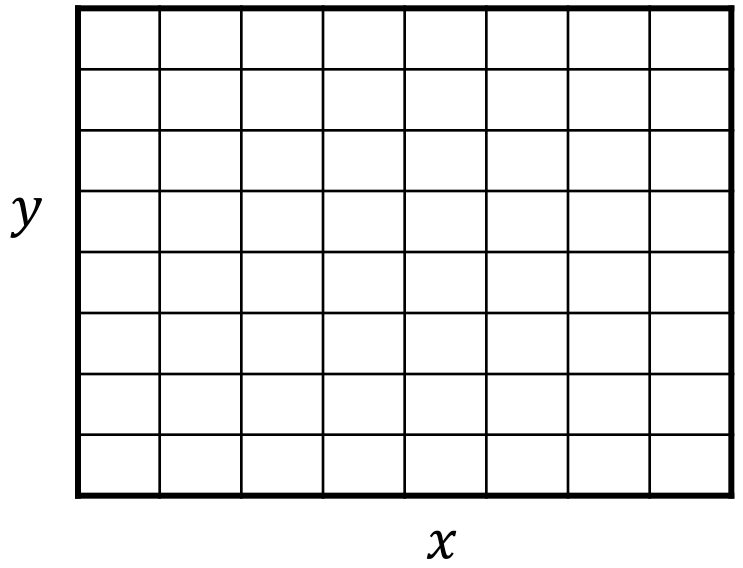
$$I(x) = \sqrt{\frac{2}{n}} \sum_{u=0}^{n-1} F(u)C(u) \cos \frac{(2x+1)\, u\, \pi}{2n}, x = 0, \dots, n-1$$

où

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{pour} \quad u = 0, \\ 1 & \text{sinon} \end{cases}$$

Extension DCT du 1D au 2D

- Calculer la DCT 1D sur chaque ligne d'un bloc
- Puis sur chaque colonne de coefficients 1D



Equations pour DCT 2D

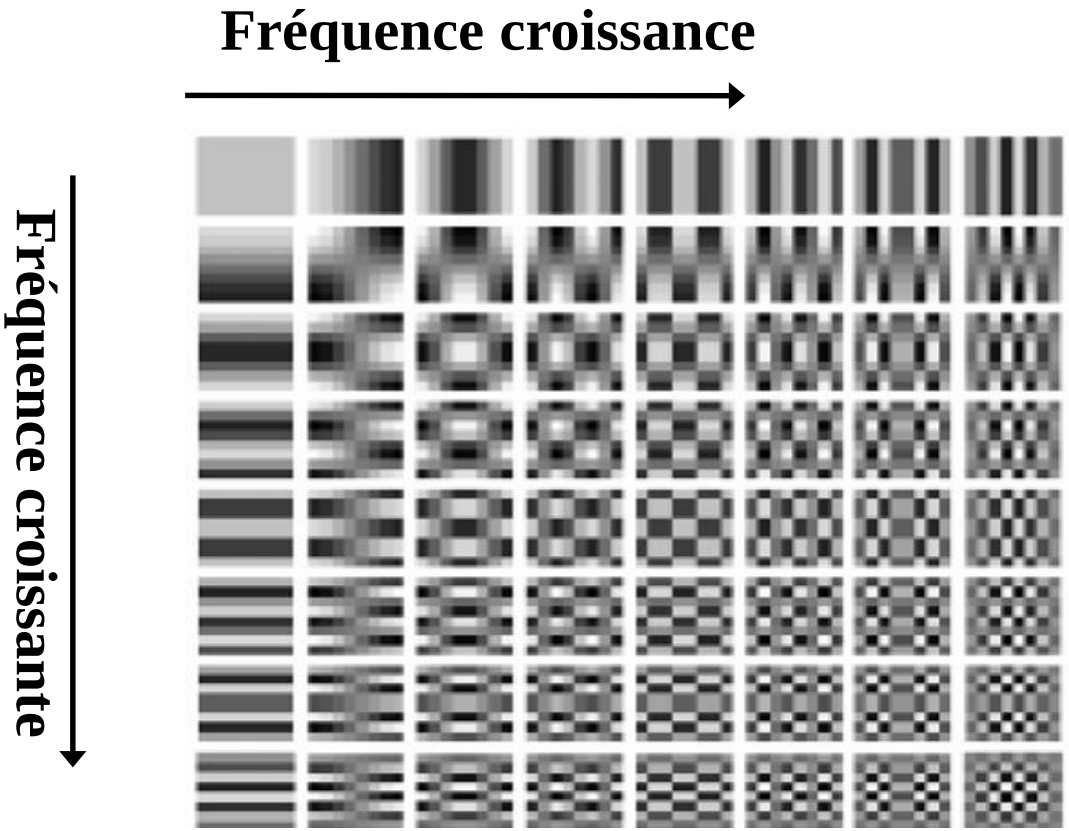
- DCT :

$$F(u,v) = \frac{2}{\sqrt{n\,m}}\;\; C(u)C(v)\sum_{x=0}^{n-1}\sum_{y=0}^{m-1} I(x,y)\cos\left(\frac{(2x+1)\,u\,\pi}{2n}\right)\cos\left(\frac{(2y+1)\,v\,\pi}{2m}\right)$$

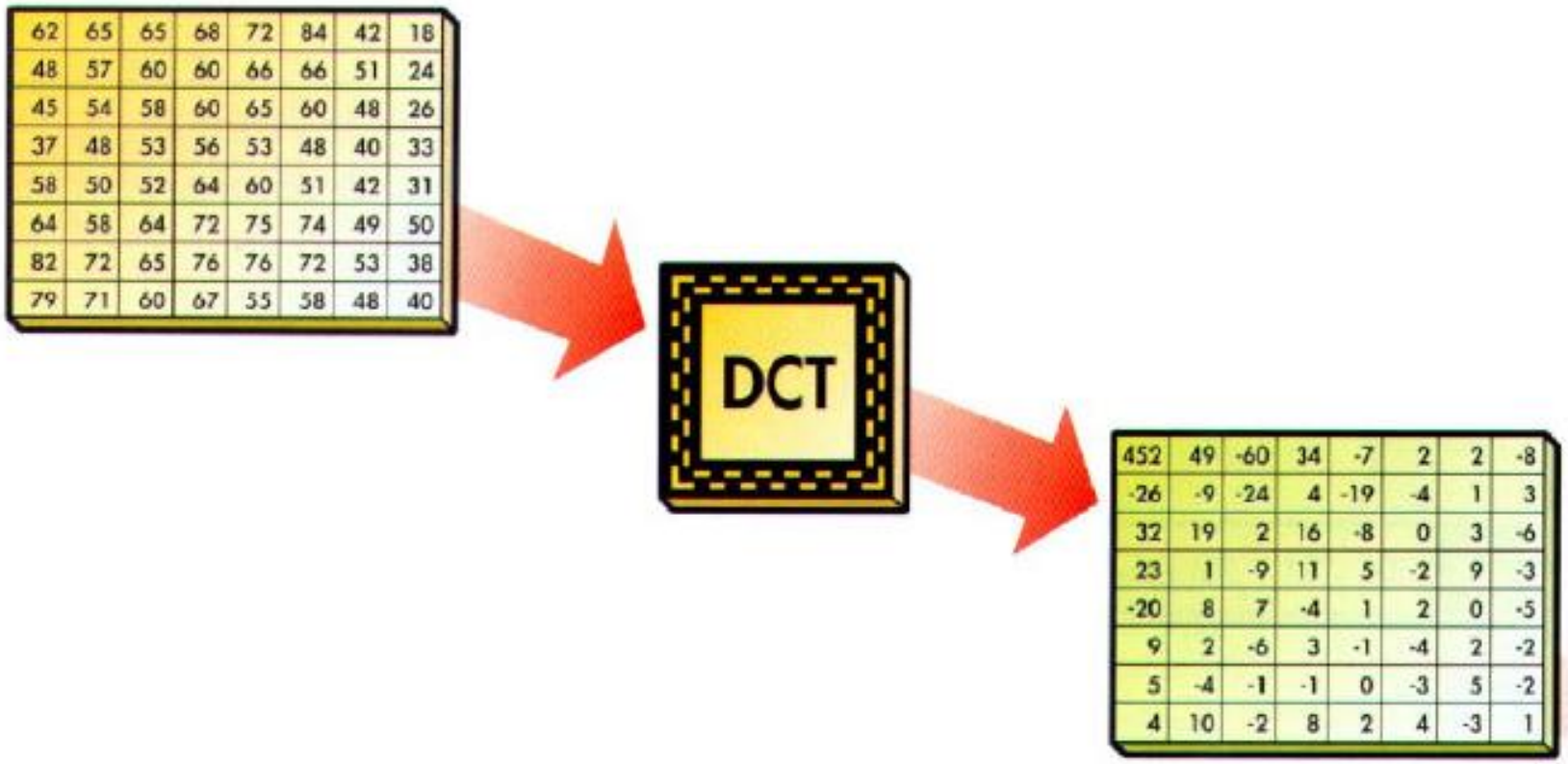
- DCT inverse :

$$I(x,y) \;= \frac{2}{\sqrt{n\,m}}\;\; \sum_{u=0}^{n-1}\sum_{v=0}^{m-1} F(u,v)C(u)C(v)\cos\left(\frac{(2x+1)\,u\,\pi}{2n}\right)\cos\left(\frac{(2y+1)\,v\,\pi}{2m}\right)$$

Visualisation des fonctions de base



Application de la DCT



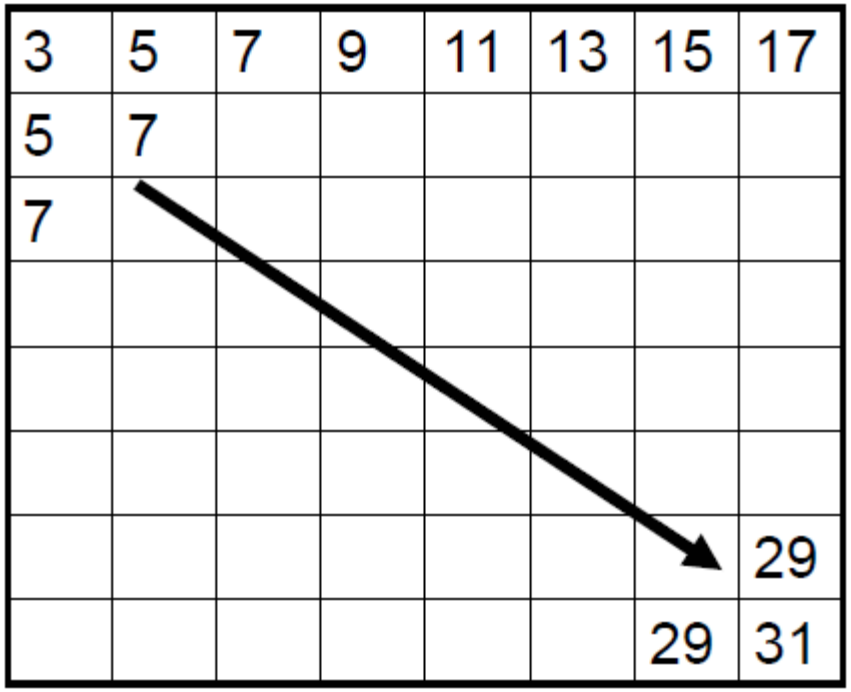
ETAPE 2 : Quantification

- La seconde phase est la « QUANTIFICATION » des coefficients.
- Chaque coefficient DCT est divisé par un facteur de la TABLE DE QUANTIFICATION et ensuite ARRONDI:
  - Coefficient DCT : $n(i,j)$
  - Facteur de quantification : $q(i,j)$
  - Coefficient quantifié : $\mathrm{E}[n(i,j)\ /q(i,j)]$ où $\mathrm{E}[x]$ est la partie entière de x
- Dans le processus de décodage, multiplier par l'entier.
- Un grand pas de quantification conduit à des pertes importantes !

Matrice de quantification

- Facteur de qualité 2

3	5	7	9	11	13	15	17
5	7						
7							
							29
						29	31



Q(i,j)=1+((1+i+j) *qualité)

Bloc 8x8

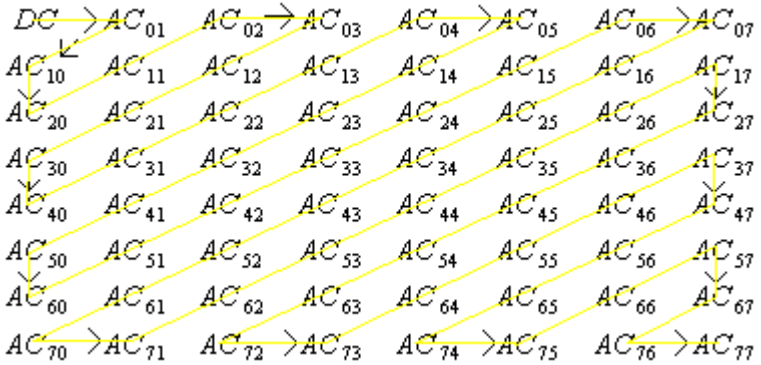
Table de quantification

Vision humaine moins sensible

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

Vision humaine moins sensible

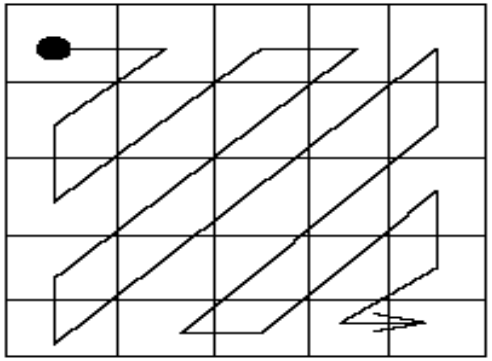
ETAPE 3 : Codage entropique



- Compression sans perte des coefficients AC (Alternating-Current) et DC (Direct-Current)
- Codage différent entre DC et AC
  - Les coefficients DC changent lentement, donc ils seront codés par différence

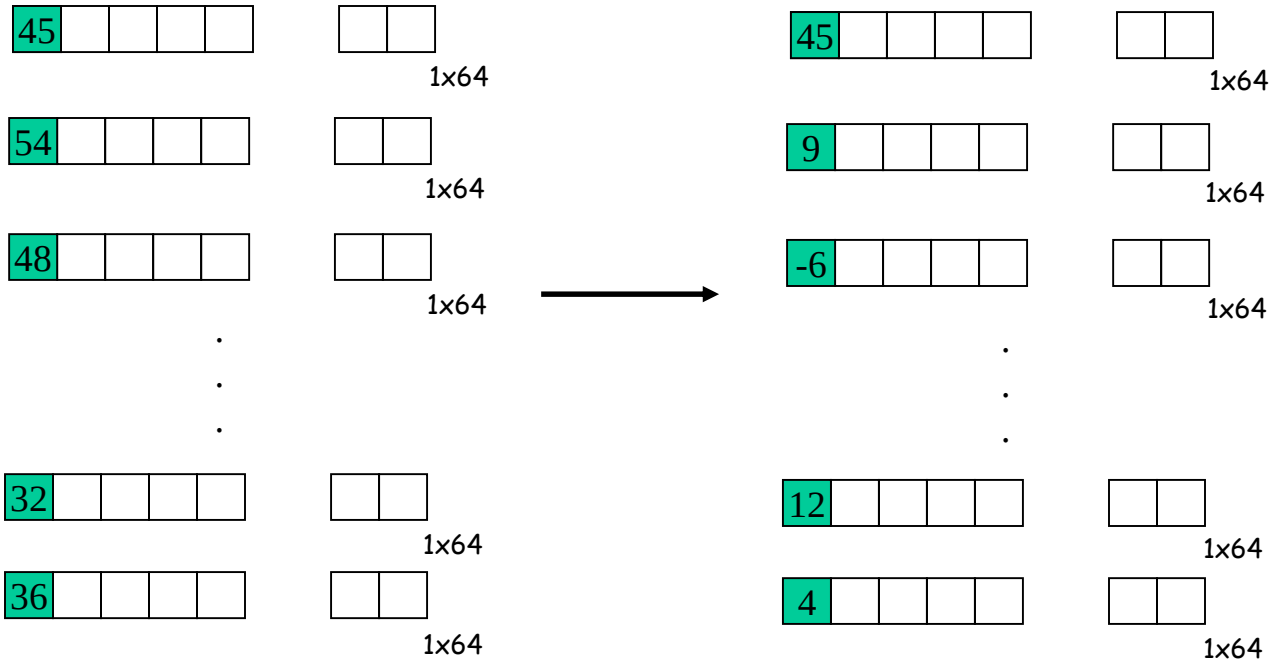
Parcours zig-zag

- Les coefficients sont parcourus des basses fréquence vers les hautes fréquences
- Ceci produit de longues suites de 0 vers la fin du parcours



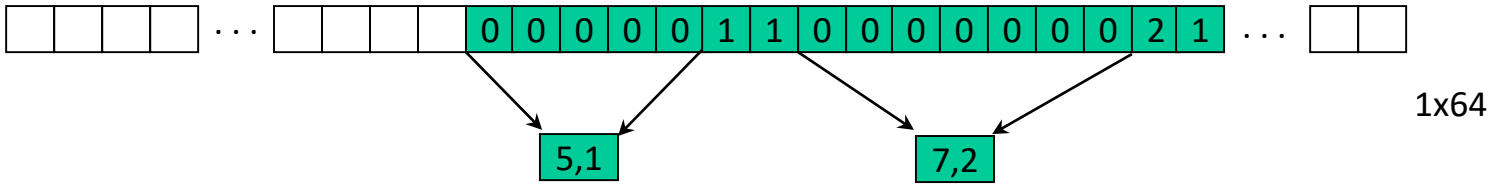
Codage DPCM pour DC

- Le coefficient DC est souvent grand mais varie peu entre les blocs. Le codage “Differential Pulse Code Modulation (DPCM)” code la différence entre les blocs.
- La différence est ensuite codée avec un codage entropique approprié.



RLE sur les coefficients AC

- Le vecteur 1x64 a beaucoup de 0, surtout vers la fin.
  - Les entrées vers la fin capturent surtout les hautes fréquences qui sont moins présentes dans l'image.
  - C'est aussi un résultat de la quantification
- Une suite de 0 est codée comme un couple (*skip*,*value*), où *skip* est le nombre de 0 et *value* est la prochaine valeur non-nulle.
  - Le couple (0,0) désigne la fin du block .
- Les couples sont ensuite codés avec un codage entropique approprié.



Erreur de reconstruction

- L'image finale $K(x, y)$ doit être proche de l'image originale $I(x, y)$
- Cette "proximité" est généralement mesurée avec l'erreur quadratique moyenne (MSE) et le rapport signal à bruit crête (PSNR)

$$MSE = \frac{1}{n\,m}\sum_{x=0}^{n-1}\sum_{y=0}^{m-1}(I(x,y) - K(x,y))^2$$

$$PSNR = 10\log_{10}\left(\frac{\max I^2(x,y)}{MSE}\right) = 20\log_{10}\left(\frac{\max I(x,y)}{\sqrt{MSE}}\right)$$

Exemple – Une photo de 2.76 MB

Exemple – Une photo de 600 KB

Exemple – Une photo de 350 KB

Exemple – Une photo de 240 KB

Exemple – Une photo de 144 KB

Exemple – Une photo de 88 KB

Brève analyse

- Ceci est la présentation du mode **Baseline Sequential Mode** mais il y d'autres modes (progressif,…)
- La précision diminue lorsque la taille diminue
- Faux contours visibles à 144K et 88K
- Quelle est la meilleure taille?
  - Cette question reste sans réponse
  - Tout dépend des besoins et des ressources de l'utilisateur

Conclusion

- La compression d'images est importante pour leur stockage et leur manipulation
- Elle s'étend vers la compression des vidéos
- Il existe de nombreux standards (JPEG2000, MPEG, H264, HEVC, etc.)
- Très important pour l'univers du numérique : les systèmes embarqués (voiture autonome), les objets connectés, l'Internet, les réseaux sociaux, etc.