Données multimedia - Notes de cours

Renaud Keriven
(keriven@cermics.enpc.fr)

10 mars 1995

Plan

- Les types de données
  - Textes
  - Images
  - Sons
  - Animations
- Pourquoi comprimer.
- Comment comprimer.
  - Huffman
  - Compression arithmétique
  - Compression avec dictionnaire
  - JPEG
  - MPEG
  - Le son
- Quelques formats standards

1 Les types de données

1.1 Textes

Codage ASCII: on représente chaque caractère par un entier entre 0 et 255 (un octet = 8 bits = 256 valeurs). Les codes entre 0 et 127 sont standards, les codes supérieurs dépendent de l'alphabet (choix des lettres accentuées, etc.). Exemple:

```
32 sp    48 0    64 @    80 P     96 `    112 p
33 !     49 1    65 A    81 Q     97 a    113 q
34 "     50 2    66 B    82 R     98 b    114 r
35 #     51 3    67 C    83 S     99 c    115 s
36 $     52 4    68 D    84 T    100 d    116 t
37 %     53 5    69 E    85 U    101 e    117 u
38 &     54 6    70 F    86 V    102 f    118 v
39 '     55 7    71 G    87 W    103 g    119 w
40 (     56 8    72 H    88 X    104 h    120 x
41 )     57 9    73 I    89 Y    105 i    121 y
42 *     58 :    74 J    90 Z    106 j    122 z
43 +     59 ;    75 K    91 [    107 k    123 {
44 ,     60 <    76 L    92 \    108 l    124 |
45 -     61 =    77 M    93 ]    109 m    125 }
46 .     62 >    78 N    94 ^    110 n    126 ~
47 /     63 ?    79 O    95 _    111 o    127 del
```

Autres codages:

– sur plus de 8 bits pour les alphabets plus importants, ou pour prendre en compte tous les alphabets en un seul codage. Non encore couramment utilisé.

– Les traitements de textes (Word, ...) parsèment les fichiers d'informations supplémentaires (mise en page, etc.)

1.2 Images

– Discrétisation spatiale: l'image est divisée en points (pixel). Ex: un écran de PC fait 1024x768, une page d'imprimante 2000x3000, une image satellite 10000x10000

– Discrétisation de l'intensité:

  – Noir et Blanc: un bit par pixel (0: noir, 1: blanc)

  – Niveaux de gris: un octet par pixel (de 0=noir à 255=blanc)

  – Couleurs:

    – "Vraies couleurs": un octet par composante (rouge, vert, bleu), soit 24 bits par pixel.

- Avec palette: on choisit convenablement 256 triplets (rouge, vert, bleu) (la palette). Chaque pixel utilise un octet: l'indice du triplet de la palette à afficher. On choisit en général la “meilleure” palette possible pour chaque image à coder (suivant les couleurs les plus utilisées dans l'image).

1.3 Sons

- Discrétisation spatiale: le son est un signal dépendant du temps. On choisit d'échantilonner sa valeur à intervalles de temps réguliers. On parle de fréquence d'échantillonage (sur PC ou Mac: 11.025 KHz, sur CD: 44.1 KHz, sur DAT: 48 KHz)
- Discrétisation de l'intensité: chaque valeur (tension) analysée est transformée en un entier avec plus ou moins de précision par un convertisseur analogique/digital (DAC) (sur PC ou Mac: 8 bits, sur CD ou DAT: 16 bits)

1.4 Animations

En principe, il s'agit juste d'un suite d'images et d'un son, le tout étant synchronisé. Mêmes conventions de discrétisation donc que pour les images et les sons...

2 Pourquoi comprimer

Quelques points de repères:

- Stockage:
  - Une disquette: 1.4Mo (40Ko il y a 10 ans)
  - Un disque dur: 300Mo à 1Go (5Mo il y a 10 ans)
  - Une cartouche de sauvegarde: 500Mo à 4Go sur une petite cassette (30Mo sur une bande de 30cm sur les gros systèmes informatiques il y a 5 ans)
- Transfert sur réseau:
  - Chez un particulier: 14 Kbit/s (minitel: 1200 bits/s)
  - A l'ENPC: 65 Kbits/s à partager entre tous les utilisateurs
  - Ailleurs: 256 Kbits/s à 2 Mbits/s
- Taille des données:
  - Une page de texte: 1Ko
  - Une image écran en vraies couleurs: 2.5Mo ($1024 \times 764 \times 3$)

- 30 secondes de son basse qualité: 300Ko (30×11025)
- 3 minutes de son qualité CD: 31Mo ($180 \times 44100 \times 4$)
- 1 minute de télévision sans le son: 2Go ($720 \times 486 \times 3 \times 30 \times 60$) (11Go pour de la télévision haute définition)

D'où le désir de comprimer pour stocker les données sur les supports et la nécessité absolue de le faire pour les transférer en temps réel!

3 Comment comprimer

3.1 Huffman

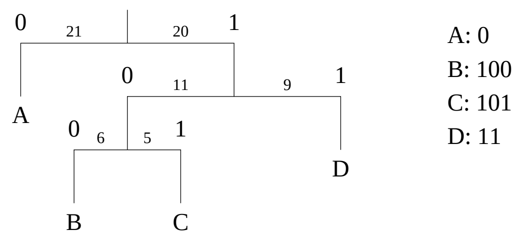

FIG. 1 - *Codage de Huffman*

Au lieu de coder tous les caractères sur le même nombre de bits, il vaut mieux coder sur moins de bits les caractères fréquents quite à augmenter le nombre de bits représentant les caractères les moins rencontrés.

Exemple: on veut comprimer un message écrit dans un alphabet à 4 caractères (A,B,C,D). Sans compression, on devrait coder chaque caractère sur 2 bits. Supposons que le message utilise A 21 fois, B 6 fois, C 5 fois et D 9 fois. On va construire un nouveau codage comme suit. On coupe (A,B,C,D) en deux sous listes telles que les nombres de caractères utilisés dans chaque sous liste sont les plus proches possible: (A;21) et (B,C,D;20). Puis on recoupe: (B,C;11) et (D;9). Enfin (B;6) et (C;5). Ce qui donne l'arbre et le codage de la figure 1. Au lieu de coder le message sur: $2 \times (21 + 6 + 5 + 9) = 82$ bits, on le code sur: $1 \times 21 + 3 \times 6 + 2 \times 5 + 2 \times 2 = 72$ bits. (Noter que le codage est tel qu'il n'y a pas d'ambiguïté dans le flot de bits lors de son décodage).

Améliorations possibles: il faut aussi transmettre le codage dans le message codé, ce qui augmente sa longueur. On utilise plutôt:

- Un codage fixe, basé par exemple sur la fréquence statistique d'utilisation des lettres dans une langue.
- Un codage adaptatif: le codage varie au fur et à mesure de l'apparition des caractères: l'arbre est sans cesse mis à jour. Il devient inutile de transmettre le codage qui est reconstruit à la décompression. En outre, on n'a plus besoin de balayer tout le fichier avant de comprimer.

3.2 Compression arithmétique

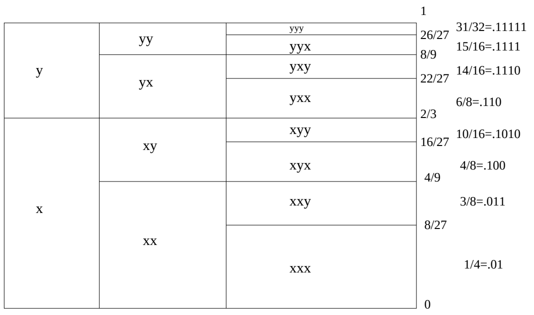

FIG. 2 - *Codage arithmétique*

Le codage de Huffman n'utilise que de manière approchée les fréquences d'utilisation (de manière exacte si elles sont puissances de deux). La compression arithmétique tient compte des fréquence exactes. Supposons qu'on ait à coder un message dans un alphabet à deux caractères (x,y) de fréquences d'apparition respectives $\frac{2}{3}$ et $\frac{1}{3}$. Chaque caractère est représenté par un intervalle de $[0,1]$: $x = [0, \frac{2}{3}], y = [\frac{2}{3}, 1]$. Idem pour les messages à deux caractères: $xx = [0, \frac{4}{9}], xy = [\frac{4}{9}, \frac{2}{3}], yx = [\frac{2}{3}, \frac{8}{9}], yy = [\frac{8}{9}, 1]$, et ainsi de suite (voir figure 2). Enfin, pour le message à coder, on choisit le nombre de l'intervalle le plus court à coder en binaire: $xxx = \frac{1}{4} = .01, xxy = \frac{3}{8} = .011, xyx = \frac{4}{8} = .100, \ldots$

Mêmes améliorations possibles que pour Huffman: probabilités fixes ou codage adaptatif.

3.3 Compression avec dictionnaire

Il s'agit du principe sur lequel sont basés tous les programmes de compressions de textes actuels (compress, gzip, pkzip, lha, sit, ...). Pendant le parcours du message, on mémorise les séquences de plusieurs caratères rencontrées en leur affectant un code. Si on les rencontre à nouveau, on utilise leur code plutôt que la séquence elle-même. Exemple: on comprime le message `/WED/WE/WEE/WEB`.

Entrée	Sortie	Dictionnaire
/W	/	256=/W
E	W	257=WE
D	E	258=ED
/	D	259=D/
WE	256	260=/WE
/	E	261=E/
WEE	260	262=/WEE
/W	261	263=E/W
EB	257	264=WEB
	B	

A la décompression, on reconstruit aussi le dictionnaire:

Entrée	Sortie	Dictionnaire
/	/	
W	W	256=/W
E	E	257=WE
D	D	258=ED
256	/W	259=D/
E	E	260=/WE
260	/WE	261=E/
261	E/	262=/WEE
257	WE	263=E/W
B	B	264=WEB

Améliorations possibles:

- Si on utilise des codes dans un intervalle fixé au départ, on utilise le même nombre de bits pour tous les caractères (Si on veut par exemple un dictionnaire à 4000 entrées, il faut 12 bits par caractère. C'est gâché!). On peut s'arranger pour accroître le nombre des bits au fur et à mesure des besoins (9, puis 10, puis 11, etc.)
- Pour un texte trop long, il faudrait le couper en morceaux pour ne pas manquer de place dans le dictionnaire. On peut utiliser une "fenêtre" glissant sur le texte, ou mieux: détruire les entrées du dictionnaire les moins utilisées.

3.4 JPEG

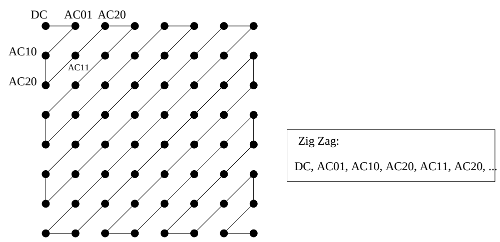

FIG. 3 - *JPEG: le zig-zag des harmoniques*

Les codages précédents marchent parfaitement sur les textes (compression d'un facteur 2 à 10). Il marchent bien sur des petites images à condition toutefois de tenir compte du côté bi-directionnel d'une image. Une image au format GIF est une image de type couleur à palette (1 octet par point) comprimée suivant un codage à dictionnaire. On obtient en général un facteur 2 à 5. Pour les images trop grandes ou en vraies couleur, ca ne marche plus du tout. D'où l'idée d'une compression **avec perte**: la décompression ne redonne pas exactement l'image d'origine mais est telle que la différence n'est pas visible psychologiquement. Le choix du seuil psychlogique est le résultat d'expériences. C'est le principe de la compression JPEG. Ses avantages sont:

- Travaille sur des images en "vraies couleurs"
- Taux de compression plus important
  - 10: image non distingable de l'originale
  - 20: image excellente
  - 50: résultat satisfaisant
  - 100: utilisable pour certaines applications
- Taux de compression réglable suivant le résultat désiré (facteur de qualité).

Le principe de base est d'utiliser une décomposition spectrale de l'image: les hautes fréquences correspondent aux détails les plus fins, qu'on dégradera en premier. On compresse de la façon suivante:

- L'image est décomposée en blocs de 8×8 pixels. On traite chacun de ses blocs séparément.
- Chaque bloc est analysé par une transformation en cosinus discrète (Discrete Cosine Transformation), plus adaptée qu'une transformée de Fourier. On obtient un spectre de 8 sur 8, avec un coefficient continu en $(0, 0)$ et des harmoniques de fréquence grandissante sur les diagonales successives (figure 3).
- Les coefficients (réels) sont discrétisés suivant le choix du facteur de qualité. Le choix du nombre de bits par coefficient (d'autant plus petit que la fréquence est élevée) est le résultat des expériences psychologiques.
- Les coefficients sont rangés suivant un zig-zag (fréquence croissante) et le tout est comprimé avec une méthode style Huffman.

A la décompression, le travail inverse est réalisé.
Il existe aussi une norme JBIG pour les images noir et blanc (Fax).

3.5 MPEG

Pour comprimer une animation, on peut évidemment comprimer chacune de ses images. Mais il est aussi tentant de comprimer l'information dans le temps puisque l'information d'une image à la suivante ne change pas brutalement. C'est l'idéee de la norme MPEG.

- Première idée: au lieu de comprimer les trois canaux rouge, vert et bleu (RGB), on décompose en un signal de luminance et deux signaux de chrominance (YUV). C'était déja le cas en télévision pour des raisons de fiabilité et de compatibilité avec le noir et blanc. Les signaux de chrominance

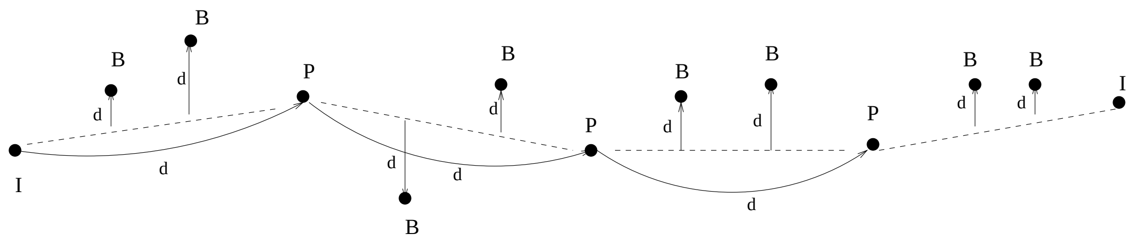

FIG. 4 - *MPEG: prédictions et vecteurs différence*

contenant moins d'information, on les code sur une résolution moitié. D'où un premier gain (déja classique un vidéo numérique non comprimée: norme 4:2:2)

- Pour passer d'une image à la suivante, on ne considère que la différence: *vecteurs différence*. L'information est moindre.

- On fait mieux: on se donne (par les vecteurs) une image et son $n^{\text{ième}}$ successeur. Puis on interpole les images intermédiaires pour enfin ne coder que la différence entre les images intermédiaires réelles et les images interpolées. L'information est encore moindre. Régulièrement, on se redonne une image complète pour éviter les dégradations. Si `I` est une image complète (intraframe), `P` (predictive frame) une image obtenue à partir de `I` ou du `P` précedent et `B` (bidirectionnal frame) les images intermédiaires, on code en général: `IBBPBBPBBPBBIBBP...` (voir figure 4).

- Supposant qu'une partie de l'image en train d'évoluer va continuer à évoluer dans le même sens, on ne code pas les vecteurs différence mais leur propre différence suivant un principe aussi utilisé en compression du son (DPCM: Digital Pulse Code Modulation)

- Enfin, on traite les images I et les vecteurs différence suivant le même principe que du JPEG.

On obtient un facteur de compression de l'ordre de 100 pour une bonne qualité. Pour décomprimer:

- Noter que les données arrivent dans l'ordre: `IPBBPBBPBBI...`

- On refait le travail inverse, en parallélisant:

  - On décode `I`
  - On affiche `I` pendant qu'on décode `P`
  - On décode `B` en l'affichant en même temps
  - Idem avec le deuxième `B`
  - On affiche le premier `P` pendant qu'on décode le suivant.

– ...

La grande complexité de la norme MPEG actuelle (MPEG-2) vient du traitement de l'entrelacement du signal vidéo réel (une trame pour les lignes paires, une trame pour les lignes impaires). De nombreux problèmes d'artefact ont du être résolus. La future norme MPEG-4 concernera les bas débits (visiophone). Une norme MHEG concernera les documents pleinement multimédia (texte, son et images mélangés).

3.6 Le son

Contrairement à ce qu'on pourrait penser, le son est ce qu'il y a de plus difficile à comprimer. L'oreille est très sensible! La compression utilisée pour les DCC (cassette numérique Philips) et mini-disc (mini CD effaçable Sony) ne satisfait pas les professionnels qui en restent au DAT (cassette numérique sans compression, à tête rotative). Elle n'en est pas moins très suffisante pour le grand public (DCC, mini-disc, radio numérique, etc.)

La norme MPEG-2 audio comprime les sons stéreophoniques qualité CD d'un facteur réglable (en général 6 pour passer de 1.5Mbits/s à 256Mbits/s). Il s'agit évidemment d'une compression avec perte. De nombreux tests psycho-accoustiques ont été menés pour quantifier la sensibilité de l'oreille humaine, notamment aux problèmes de masquage (rapport signal/bruit). Par exemple: quand on écoute un son à 1000Hz, un son à 1100Hz d'intensité inférieure de 18dB sera inaudible. Ce n'est pas le cas pour un son à 2000Hz qui ne deviendra inaudible que s'il est inférieur de 45dB. De plus, l'oreille réagit différemment suivant les intensités et les fréquences.

En pratique:

- On divise le spectre de l'oreille humaine en 32 bandes entre 20 et 20000Hz.
- Pour chaque composante du son, on calcul des effets de masque pour sa bande et pour les bandes voisines.
- En superposant tous les masques, on en déduit avec quelle précision coder les composantes dans chaque bande.

Exemple: on a une harmonique dans la bande 8 à 6500Hz d'intensité 60dB. Les tables expérimentales donnent un effet de masque de 35dB sur toute la bande 8. D'où un rapport signal bruit acceptable de 60-35=25dB: on ne code plus les autres composantes de la bande 8 que sur 5 bits ($2^5 = 32 > 25$). On tient aussi compte d'un masque sur les bandes 5 à 7 et 9 à 13.

Améliorations possibles:

- Au lieu de coder les signaux gauche et droite, on code gauche+droite et gauche-droite (moins de perte et compatibilité mono: même idée que YUV)
- Il faut aussi tenir compte des temps de réaction du cerveau:
  - Pre-masquage: 2 à 5 ms

- Post-masquage: 100 ms.

4 Quelques formats standards

- Texte:
  - Un texte brut est en ASCII. Attention: différence entre Pc, Mac et Unix dans le codage des fins de lignes. Il faut donc convertir (comme le fait ftp quand on est en mode ASCII)
  - Chaque traitement de texte a son propre format (ASCII + informations de mise en page). Convertir quand on passe de l'un à l'autre.
- Compression: il existe de nombreux logiciels de compression (compress et gzip sous Unix, pkzip, lha, arj sur Pc, sit sur Mac). Ces logiciels servent en même temps d'archiveur (ils réunissent un ensemble de fichiers en un seul). Quand on récupere un fichier sur Internet, il faut savoir reconnaître quel logiciel l'a comprimé pour l'exploiter.
- Images:
  - .gif: fichier couleur avec palette (8bits/pixel) comprimé suivant un principe de codage avec dictionnaire.
  - .jpg: fichier comprimé avec la norme du même nom. Les plus courants aujourd'hui.
- Sons:
  - .au: fichier de son sur machine Sun: 8KHz, 8bits (avec codage logarithmique de l'intensité pour une meilleure résolution dans les faibles intensités: μ-law). Pas de compression (sauf ADPCM).
  - Sons Mac: en général 11 ou 22KHz, 8bits. Pas de compression.
  - .wav: son type Pc. En général 11.025 ou 22.05KHz (diviseur de la fréquence CD), 8bits (parfois 16). Pas de compression.
  - .mp2: Son MPEG-2. 44.1KHz, 16 bits, stéreo, compression MPEG.
- Animations:
  - .mpg: animation MPEG. Pas de son (à mixer en général avec un .wav).
  - .mov ou .qt: format Macintosh. Comprimé ou non suivant la version. Avec ou sans son.
  - .avi: format Pc (Microsoft). Comprimé MPEG. Avec ou sans son.