Codage
Compression d'images

E. Jeandel

Emmanuel.Jeandel at lif.univ-mrs.fr

Une image

- Une image est un tableau de $w \times h$ pixels, chaque pixel étant représenté par une couleur, codée entre 1 et 4 bits ;

Deux types (principalement) de représentation des couleurs :

- On décrit chaque pixel par trois entiers (R, G, B) précisant chacun l'intensité en rouge/vert/bleu du pixel.
- On décrit initialement une palette de couleurs (couleur 0 = blanc, couleur 1 = jaune...) et on décrit ensuite la couleur de chaque pixel par sa position dans la table

PPM (vu en TD)

On représente ligne après ligne du tableau :

```
P3
16 16
256
```

```
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   0 0 0   0
0 0 0   0 0 0   0 0 0   0 0 0   0 97 0  97 0 62  0 97 0  1100 89  0
0 20 0  19 0 14 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 97 0  1100 89 0 97 0  1100 89  0
0 20 0  19 0 14 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 97 0  1100 89 0 1200 1310 92 0 1200 1310 92  0
0 73 0  79 0 49 0 97 0  1100 89 0 97 0  97 0 62 0 73 0  79 0 49  0
0 0 0   0 0 0   0 97 0  1100 89 0 97 0  1100 89 0 1200 1310 92  0
0 1510  1500 101 0 1510 1500 101 0 1320 1230 87 0 97 0  97 0 62  0
0 97 0  1100 89 0 97 0  1100 89 0 97 0  1100 89 0 97 0  1100 89  0
0 1870  1820 138 0 1580 1550 106 0 1510 1500 101 0 1080 1130 79  0
0 97 0  1100 89 0 97 0  1100 89 0 1510 1500 101 0 1870 1820 138 0
0 1870  1820 138 0 1580 1550 106 0 1510 1500 101 0 1510 1500 101 0
0 0 0   0 0 0   0 0 0   0 0 0   0 1510 1500 101 0 1870 1820 138 0
0 1870  1820 138 0 1580 1550 106 0 1510 1500 101 0 1510 1500 101 0
```

XPM (vu en TD)

On représente d'abord la palette, puis l'image :

```
static char *test[] = {
"16 16 17 1",
"  c #000000",
". c #050503",
"X c #080905",
"o c #14130E",
"O c #292A1C",
"+ c #494F31",
"@ c #61613E",
"# c #6C714F",
"................."
"- c #9D9A6A",
"; c #BAB589",
": c #D4CEA6",
"> c None",
">>>>>>>>>>>>>>>>>",
">>@$$o>%o>>>>>>>",
">>$$$$*Oo>>>>>>>",
">$&&&$$o+$@+>>>>",
">$$&$=$@==*@++>>",
"$$$$;;;;;-=#@+@>",
"$$=;:::;;-==#++>",
">>=;:::;;-==@++>",
">@=;;;;;--=*@++>",
">@=;;;;;--=*@++>",
">>&=-----**@+%+>",
">>&&=-===*@@+O.X",
">>>##&*##@@+O...",
">>>>O+++%++O..  ",
">>>> Xo...X.    ",
">>>>   ...     >"
```

Problèmes majeurs avec les compressions ordinaires

- Les caractères consécutifs ne représentent pas la même couleur. Exemple d'un dégradé de rouge

  0 0 0 1 0 0 2 0 0 3 0 0 4 0 0 5 0 0 6 0 0 7 0 0 ...

- Solution : traiter les 3 couleurs séparément
- Une image est en 2 dimensions : Suivant si on la représente en ligne ou en colonne, le gain en compression n'est pas le même.

Compression d'images

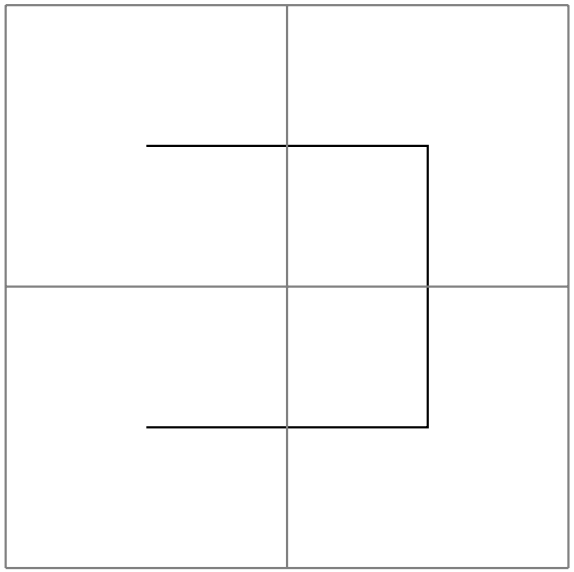
- On peut utiliser les techniques précédentes pour compresser une image
- Les images se compressent souvent mal
- Pourquoi ?
- Une image de 3000 x 2000 pixels avec 3 bits par pixels se compresse en 10 Mo, ce qui est (encore) trop.
- Méthodes toutefois utilisées pour des petites images (PNG, GIF) ou lorsqu'on ne veut pas dégrader l'image.

$2D \mapsto 1D$

- Les pixels voisins ont souvent la même couleur
- Idée : transformer l'image $2D$ en image $1D$ de façon à conserver la localité
- Space-Filling Curves

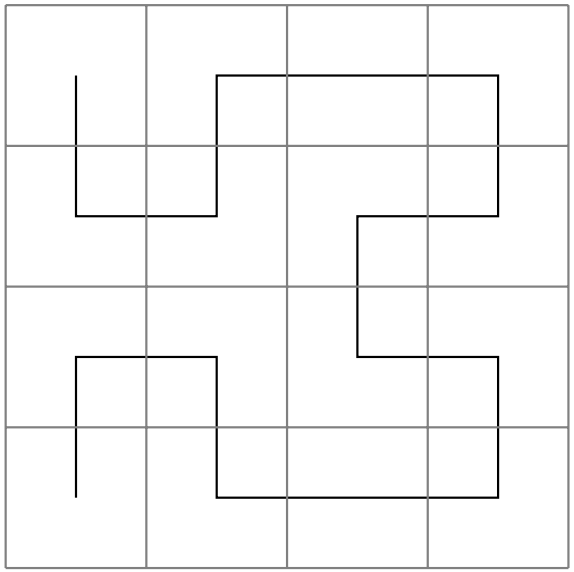
$2D \mapsto 1D$

- Les pixels voisins ont souvent la même couleur
- Idée : transformer l'image $2D$ en image $1D$ de façon à conserver la localité
- Space-Filling Curves



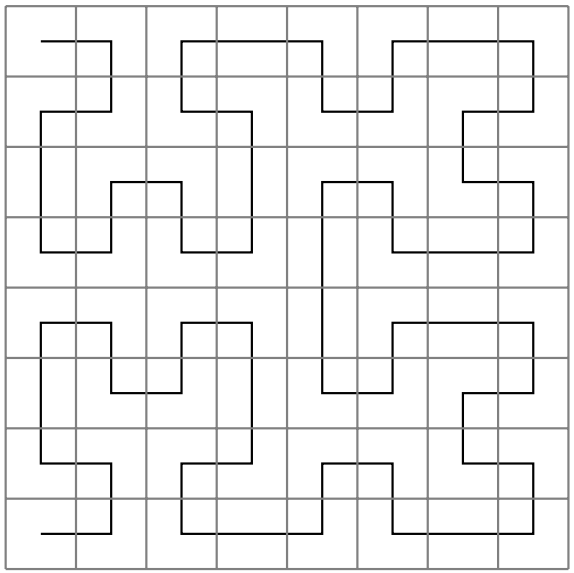
$2D \mapsto 1D$

- Les pixels voisins ont souvent la même couleur
- Idée : transformer l'image $2D$ en image $1D$ de façon à conserver la localité
- Space-Filling Curves



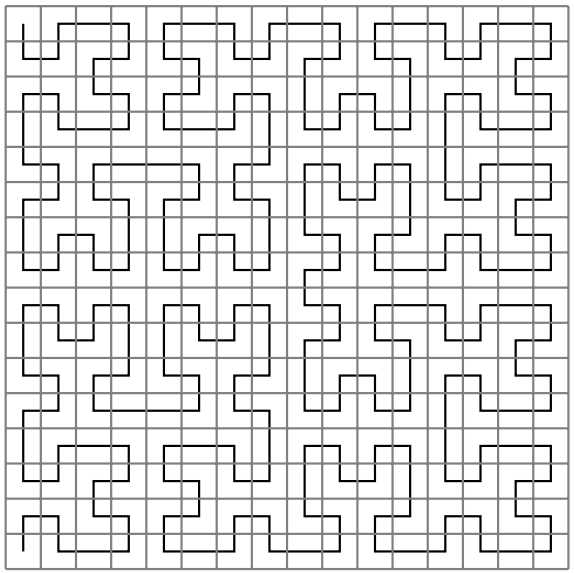
$2D \mapsto 1D$

- Les pixels voisins ont souvent la même couleur
- Idée : transformer l'image $2D$ en image $1D$ de façon à conserver la localité
- Space-Filling Curves



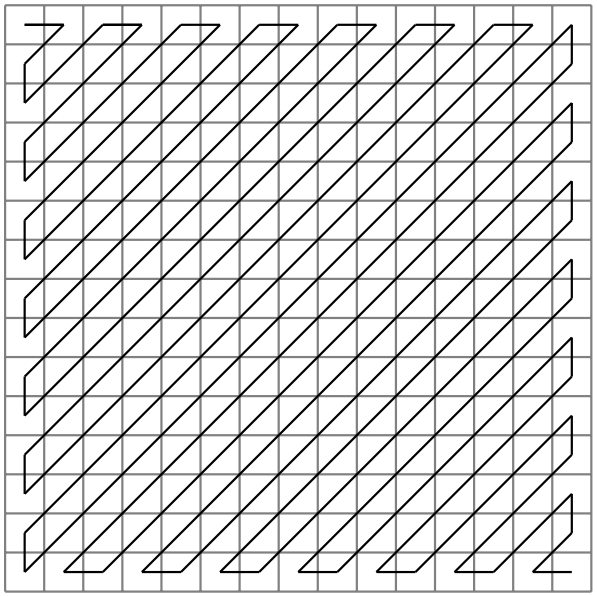
$2D \mapsto 1D$

- Les pixels voisins ont souvent la même couleur
- Idée : transformer l'image $2D$ en image $1D$ de façon à conserver la localité
- Space-Filling Curves



$2D \mapsto 1D$

- Les pixels voisins ont souvent la même couleur
- Idée : transformer l'image $2D$ en image $1D$ de façon à conserver la localité
- Space-Filling Curves



Application d'un filtre (Preprocessing)

Avant d'appliquer un algorithme de compression, on peut "préparer" l'image. Dans PNG :

- On remplace chaque pixel par sa différence avec son prédécesseur horizontal :

$$Q(x,y) = P(x,y) - P(x-1,y)$$

- Ou encore

$$Q(x,y) = P(x,y) - P(x,y-1)$$

- $$Q(x,y) = P(x,y) - \frac{P(x,y-1) + P(x-1,y)}{2}$$

- $$Q(x,y) = P(x,y) - R$$

  où R est, parmi les pixels $a = P(x-1,y)$, $b = P(x,y-1)$ et $c = P(x-1,y-1)$ le plus proche de $a+b-c$.

Exemple

$$\begin{pmatrix} 80 & 53 & 10 & 49 & 48 & 32 & 49 & 48 \\ 10 & 50 & 53 & 53 & 10 & 90 & 91 & 92 \\ 94 & 96 & 99 & 103 & 105 & 107 & 108 & 91 \\ 92 & 94 & 96 & 99 & 104 & 109 & 113 & 115 \\ 115 & 92 & 94 & 96 & 99 & 104 & 110 & 117 \\ 121 & 123 & 122 & 93 & 95 & 98 & 102 & 108 \\ 116 & 122 & 126 & 127 & 126 & 95 & 96 & 100 \\ 105 & 112 & 119 & 123 & 126 & 129 & 129 & 96 \end{pmatrix}$$

Méthode 1

$$\begin{pmatrix} 80 & -27 & -43 & 39 & -1 & -16 & 17 & -1 \\ 10 & 40 & 3 & 0 & -43 & 80 & 1 & 1 \\ 94 & 2 & 3 & 4 & 2 & 2 & 1 & -17 \\ 92 & 2 & 2 & 3 & 5 & 5 & 4 & 2 \\ 115 & -23 & 2 & 2 & 3 & 5 & 6 & 7 \\ 121 & 2 & -1 & -29 & 2 & 3 & 4 & 6 \\ 116 & 6 & 4 & 1 & -1 & -31 & 1 & 4 \\ 105 & 7 & 7 & 4 & 3 & 3 & 0 & -33 \end{pmatrix}$$

Méthode 2

$$\begin{pmatrix} 80 & 53 & 10 & 49 & 48 & 32 & 49 & 48 \\ -70 & -3 & 43 & 4 & -38 & 58 & 42 & 44 \\ 84 & 46 & 46 & 50 & 95 & 17 & 17 & -1 \\ -2 & -2 & -3 & -4 & -1 & 2 & 5 & 24 \\ 23 & -2 & -2 & -3 & -5 & -5 & -3 & 2 \\ 6 & 31 & 28 & -3 & -4 & -6 & -8 & -9 \\ -5 & -1 & 4 & 34 & 31 & -3 & -6 & -8 \\ -11 & -10 & -7 & -4 & 0 & 34 & 33 & -4 \end{pmatrix}$$

Méthode 3

$$\begin{pmatrix} 80 & -27 & -43 & 39 & -1 & -16 & 17 & -1 \\ -70 & 19 & 23 & 2 & -40 & 69 & 22 & 23 \\ 84 & 24 & 25 & 27 & 49 & 10 & 9 & -9 \\ -2 & 0 & 0 & 0 & 2 & 4 & 5 & 13 \\ 23 & -12 & 0 & 0 & -1 & 0 & 2 & 5 \\ 6 & 17 & 14 & -16 & -1 & -1 & -2 & -1 \\ -5 & 3 & 4 & 18 & 15 & -17 & -2 & -2 \\ -11 & -1 & 0 & 0 & 2 & 19 & 17 & -18 \end{pmatrix}$$

Quadtree

- On suppose pour simplifier que l'image est de taille une puissance de 2
- On coupe à chaque fois l'image en 4.
- Si un des quarts de plan est uni, on le représente par un seul noeud, dont la couleur est la couleur du noeud.
- Sinon, on le redécoupe en 4.

Compression avec pertes

- Souvent, on peut se contenter d'obtenir une image qui "ressemble" à l'image d'origine, sans lui être forcément identique
- Diminuer le nombre de couleurs
- Diminuer la taille

Test

Originale

Test

4 fois moins de pixels

Test

16 fois moins de pixels

Test

32 niveaux de gris

Test

16 niveaux de gris

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec son prédécesseur
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention. . .

Original	71	9	87	76	51	28	89	77	71
Différence									
Codage									

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec son prédécesseur
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention...

Original	71	9	87	76	51	28	89	77	71
Différence									
Codage	64								

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec son prédécesseur
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention...

Original	71	9	87	76	51	28	89	77	71
Différence		-62							
Codage	64	-64							

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec son prédécesseur
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention...

Original	71	9	87	76	51	28	89	77	71
Différence		-62	78						
Codage	64	-64	64						

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec son prédécesseur
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention...

Original	71	9	87	76	51	28	89	77	71
Différence		-62	78	-11	-25	-23	61	-12	-6
Codage	64	-64	64						

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec son prédécesseur
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention. . .

Original	71	9	87	76	51	28	89	77	71
Différence		-62	78	-11	-25	-23	61	-12	-6
Codage	64	-64	64	-16	-32	-32	48	-16	-16

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec son prédécesseur
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention. . .

Codage	64	-64	64	-16	-32	-32	48	-16	-16
Décodage									

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec son prédécesseur
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention. . .

Codage	64	-64	64	-16	-32	-32	48	-16	-16
Décodage	64								

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec son prédécesseur
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention. . .

Codage	64	-64	64	-16	-32	-32	48	-16	-16
Décodage	64	0							

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec son prédécesseur
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention...

Codage	64	-64	64	-16	-32	-32	48	-16	-16
Décodage	64	0	64						

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec son prédécesseur
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention. . .

Codage	64	-64	64	-16	-32	-32	48	-16	-16
Décodage	64	0	64	48	16	-16	32	16	0

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec son prédécesseur
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention...

Codage	64	-64	64	-16	-32	-32	48	-16	-16
Décodage	64	0	64	48	16	-16	32	16	0
Original	71	9	87	76	51	28	89	77	71

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec son prédécesseur
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention... à la dérive !

Codage	64	-64	64	-16	-32	-32	48	-16	-16
Décodage	64	0	64	48	16	-16	32	16	0
Original	71	9	87	76	51	28	89	77	71

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec le recalcul du prédécesseur.
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention... à la dérive !

Codage	64	-64	64	-16	-32	-32	48	-16	-16
Décodage	64	0	64	48	16	-16	32	16	0
Original	71	9	87	76	51	28	89	77	71

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec le recalcul du prédécesseur.
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention... à la dérive !

Original	71	9	87	76	51	28	89	77	71
Différence									
Codage									
Recalcul									

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec le recalcul du prédécesseur.
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention... à la dérive !

Original	71	9	87	76	51	28	89	77	71
Différence									
Codage	64								
Recalcul	64								

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec le recalcul du prédécesseur.
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention... à la dérive !

Original	71	9	87	76	51	28	89	77	71
Différence		-62							
Codage	64	-64							
Recalcul	64	0							

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec le recalcul du prédécesseur.
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention... à la dérive !

Original	71	9	87	76	51	28	89	77	71
Différence		-62	87						
Codage	64	-64							
Recalcul	64	0							

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec le recalcul du prédécesseur.
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention... à la dérive !

Original	71	9	87	76	51	28	89	77	71
Différence		-62	87						
Codage	64	-64	80						
Recalcul	64	0	80						

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec le recalcul du prédécesseur.
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention... à la dérive !

Original	71	9	87	76	51	28	89	77	71
Différence		-62	87	-4	-13	-20	73	-3	5
Codage	64	-64	80						
Recalcul	64	0	80						

PCM

- Idée : Au lieu de mettre le pixel, mettre la différence avec le recalcul du prédécesseur.
- Ne pas utiliser 8 bits mais beaucoup moins (par exemple 4 bits)
- Faire attention. . . à la dérive !

Original	71	9	87	76	51	28	89	77	71
Différence		-62	87	-4	-13	-20	73	-3	5
Codage	64	-64	80	-16	-16	-32	64	-16	0
Recalcul	64	0	80	64	48	16	80	64	64

PCM

- Généralisations : Représenter le pixel par la différence entre une valeur précalculée $R(x, y)$ et sa vraie valeur :
- Le pixel précédent

$$R(x, y) = P(x - 1, y)$$

- Interpolation linéaire avec les deux pixels précédents

$$R(x, y) = 2 * P(x - 1, y) - P(x - 2, y)$$

- Interpolation quadratique avec les trois pixels précédents

$$R(x, y) = P(x - 3, y) - 3 * P(x - 2, y) + 3P(x - 1, y)$$

Compression d'images - idée générale

- Pour améliorer la compression, on peut essayer de transporter l'image dans un nouvel espace
- Trouver un espace où l'information (l'intensité lumineuse) est concentrée en un petit nombre de points
- Compresser dans le nouvel espace

Transformée

- Groupons les pixels 2 par 2, et remplaçons (x, y) par $(x + y, x - y)$
- Effectuons cette opération sur les lignes, puis sur les colonnes, afin d'éliminer la "corrélation"

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \mapsto \begin{pmatrix} a+b & a-b \\ c+d & c-d \end{pmatrix} \mapsto \begin{pmatrix} a+b+c+d & a-b+c-d \\ a+b-(c+d) & (a-c)+(d-b) \end{pmatrix}$$

- Fonction inverse :

$$\begin{pmatrix} x & y \\ z & w \end{pmatrix} \mapsto \begin{pmatrix} x+y+z+w & x-y+z-w \\ x+y-(z+w) & (x-z)+(w-y) \end{pmatrix}$$

Exemple

$$\begin{pmatrix} 193 & 73 & 165 & -47 & 180 & -20 & 280 & -86 \\ -13 & 67 & -39 & -39 & -64 & 96 & 0 & 2 \\ 376 & 4 & 397 & 7 & 425 & -1 & 427 & -29 \\ -4 & 0 & -7 & -1 & -7 & 3 & 15 & 19 \\ 451 & -37 & 405 & -25 & 396 & 10 & 437 & 17 \\ 21 & 25 & 27 & -31 & -8 & -2 & -13 & -1 \\ 455 & 21 & 495 & 11 & 476 & -34 & 421 & -29 \\ -13 & 1 & -5 & 3 & 28 & 34 & 29 & -37 \end{pmatrix}$$

Transformée

- Dans la représentation

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \mapsto \begin{pmatrix} a+b+c+d & a-b+c-d \\ a+b-(c+d) & (a-c)+(d-b) \end{pmatrix}$$

  Le coefficient le plus important est le premier
- Ne garder que le premier et diminuer la précision des autres
- Compression ?

Exemple

Originale

Exemple

En ne gardant que 6 bits des coefficients moins pertinents

Exemple

En ne gardant que 2 bits des coefficients moins pertinents

Transformée

- On se place en dimension 1 pour simplifier
- On groupe les pixels k par k.
- On applique une matrice (application linéaire) pour transformer les k valeurs en k nouvelles valeurs
- Quelle matrice choisir ?
- Méthode optimale (en un certain sens) compliquée à mettre en oeuvre.

Transformées

- Il faut choisir une matrice M qui est inversible
- Avoir une matrice orthogonale ($M^T = M^{-1}$) a ses avantages

Transformée de Hadamard

- C'est une matrice dont les lignes sont numérotées par des mots sur $\{0,1\}$ (c'est à dire pour 4 lignes, par $00, 01, 10, 11$ plutôt que par $1, 2, 3, 4$
- Le coefficient en v, w est $(-1)^{\sum_i v_i w_i}$
- Le coefficient en v, w est (-1) si $\sum_i v_i w_i$ est impair, 1 sinon.

$$\begin{pmatrix} & 00 & 01 & 10 & 11 \\ 00 & 1 & 1 & 1 & 1 \\ 01 & 1 & -1 & 1 & -1 \\ 10 & 1 & 1 & -1 & -1 \\ 11 & 1 & -1 & -1 & 1 \end{pmatrix}$$

Transformée de Hadamard - Exemple - Ligne

$$\begin{pmatrix} 369 & 5 & 57 & 81 & 15 & -29 & 91 & 51 \\ 449 & -121 & -129 & -119 & -117 & 41 & 37 & 39 \\ 803 & 9 & 1 & -17 & -19 & -21 & -25 & 21 \\ 822 & -12 & -24 & -2 & -60 & 2 & 6 & 4 \\ 827 & 9 & -7 & 27 & -33 & 33 & 41 & 23 \\ 862 & 18 & 12 & -28 & 56 & 36 & 46 & -34 \\ 908 & 20 & 10 & 30 & 74 & -34 & -40 & -40 \\ 939 & 19 & 5 & -39 & -21 & -41 & -55 & 33 \end{pmatrix}$$

Transformée de Hadamard - Exemple - Ligne puis Colonne

$$\begin{pmatrix} 5979 & -53 & -75 & -67 & -105 & -13 & 101 & 97 \\ -165 & 139 & 197 & 309 & 179 & -89 & 33 & 13 \\ -965 & -125 & -59 & -11 & -53 & 175 & 329 & 61 \\ -65 & 95 & 137 & 201 & -93 & -57 & 65 & 125 \\ -1093 & -185 & -115 & -47 & -257 & -1 & 117 & 133 \\ -33 & 155 & 225 & 61 & 167 & -97 & 13 & 45 \\ -649 & -101 & -39 & -27 & 7 & -113 & -35 & 69 \\ -57 & 115 & 185 & 229 & 275 & -37 & 105 & -135 \end{pmatrix}$$

Transformée de Haar

- La transformée de Haar de niveau k travaille sur 2^k pixels
- Dans chaque bloc de 2^k, on remplace chaque couple de deux pixels par
  - Leur moyenne
  - La différence entre le premier pixel et la moyenne
- On obtient ainsi un premier bloc contenant toutes les moyennes, et un deuxième bloc contenant la différene
- On exécute ensuite la transformée de Haar de niveau $k-1$ sur le bloc contenant les moyennes
- Exécution sur

$$\begin{pmatrix} 15 & 25 & 80 & 90 & 18 & 18 & 68 & 14 \end{pmatrix}$$

Haar - Exemple

$$\begin{pmatrix} 48 & 41 & 45 & 70 & 18 & -11 & -5 & -21 \\ 94 & 99 & 106 & 106 & 1 & 1 & 0 & -7 \\ 112 & 101 & 99 & 109 & -9 & -6 & 2 & 4 \\ 113 & 123 & 119 & 105 & 5 & 2 & -8 & -7 \\ -3 & -9 & -16 & 0 & 16 & -9 & 24 & 0 \\ -1 & -1 & -1 & 3 & 0 & 0 & 0 & 4 \\ 5 & 6 & -2 & -3 & 6 & -7 & 0 & 0 \\ -3 & -1 & 7 & 7 & 0 & 0 & 8 & -9 \end{pmatrix}$$

Haar - Exemple

$$\begin{pmatrix} 70 & 81 & -26 & -24 & 18 & -11 & -5 & -21 \\ 112 & 108 & -5 & -4 & 1 & 1 & 0 & -7 \\ 0 & -6 & 3 & -6 & -9 & -6 & 2 & 4 \\ 0 & 1 & 5 & -6 & 5 & 2 & -8 & -7 \\ -3 & -9 & -16 & 0 & 16 & -9 & 24 & 0 \\ -1 & -1 & -1 & 3 & 0 & 0 & 0 & 4 \\ 5 & 6 & -2 & -3 & 6 & -7 & 0 & 0 \\ -3 & -1 & 7 & 7 & 0 & 0 & 8 & -9 \end{pmatrix}$$

Haar - Exemple

$$\begin{pmatrix} 92 & -17 & -26 & -24 & 18 & -11 & -5 & -21 \\ -1 & -3 & -5 & -4 & 1 & 1 & 0 & -7 \\ 0 & -6 & 3 & -6 & -9 & -6 & 2 & 4 \\ 0 & 1 & 5 & -6 & 5 & 2 & -8 & -7 \\ -3 & -9 & -16 & 0 & 16 & -9 & 24 & 0 \\ -1 & -1 & -1 & 3 & 0 & 0 & 0 & 4 \\ 5 & 6 & -2 & -3 & 6 & -7 & 0 & 0 \\ -3 & -1 & 7 & 7 & 0 & 0 & 8 & -9 \end{pmatrix}$$

Transformée de Haar

- La transformée de Haar de niveau k s'écrit sous forme matricielle (Exercice)
- Intérêt : connaître les 2^n premiers pixels du résultat permet d'obtenir une "approximation" raisonnable de l'image (décode progressif)

Exercice : dans la formule, on met $\frac{a+b}{2}$ et $a - \frac{a+b}{2}$. Pourquoi ne met-on pas $\frac{a-b}{2}$ à la place ?

Transformée en cosinus

- La transformée en cosinus (DCT) consiste à prendre la matrice de taille n

$$\left(\cos(\frac{2\pi}{n}(i+\frac{1}{2})j)\right)_{i,j}$$

Compression

- Après avoir appliqué la transformée, un seul coefficient est gros
- On traite ce coefficient à part

Il reste à traiter les autres

- On les code sur moins de bits
- On utilise un seuil ϵ : Tout nombre en dessous de ϵ est considéré comme nul

Compression (suite)

- On compresse ensuite tous les “gros” coefficients ensemble
- Les petits coefficients sont compressés (Huffman)

Méthode utilisée dans JPEG.

JPEG

- On exprime chaque pixel dans un espace cdY (avec la luminance) comme paramètre plutôt que RGB
- On compresse séparément chacune des composantes
- On compresse moins la composante Y que les deux autres.
- On diminue la résolution par deux pour les deux composantes c et d.
- On compresse ensuite chaque bloc 8×8 de l'image
- On applique sur chaque bloc la transformation DCT ;
- On diminue la précision des coefficients (facteur *qualité* de JPEG)
- On compresse les gros coefficients ensemble (Comme du PCM pour simplifier)
- On compresse chaque bloc séparément (Huffman)

Test

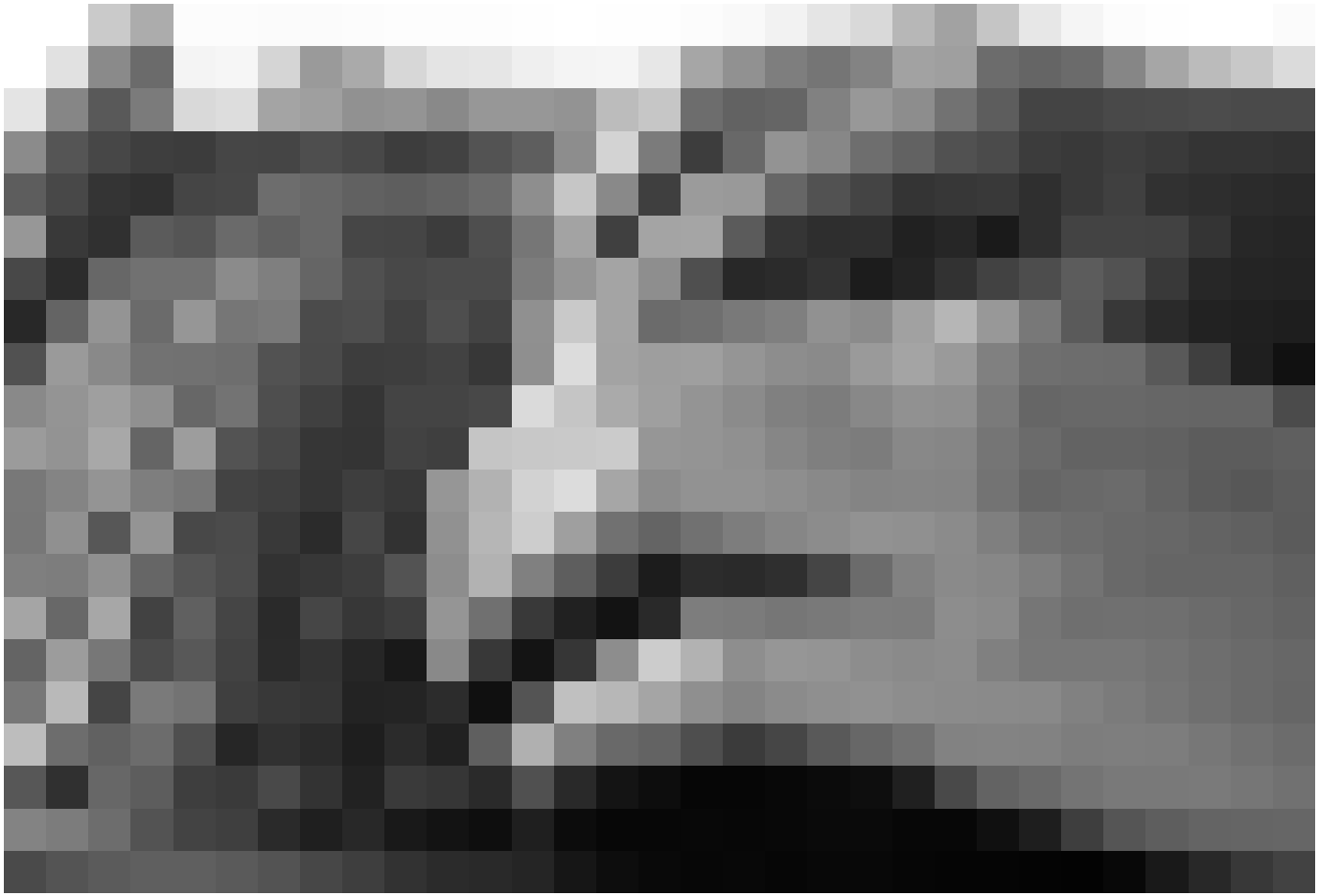
Originale

Test

JPEG, qualité 0

Test

Image en résolution 1/8



Test

JPEG, qualité 50

Test

JPEG, qualité 75

Test

JPEG, qualité 100

Test

- Attention, JPEG même en qualité 100 ne restitue pas exactement la même image.
- On peut tester en partant d'une image PPM un peu particulière..

Test

- Attention, JPEG même en qualité 100 ne restitue pas exactement la même image.
- On peut tester en partant d'une image PPM un peu particulière..

```
\section{Compression}

On essaie ici de compresser des images, c'est à dire des tableaux de pixel
chaque pixel pouvant prendre une couleur entre $0$ et $255$. On ne
s'intéresse ici qu'à compresser des images en niveaux de gris. En effet, l
plupart des techniques de compression, pour compresser une image au format
RVB, compressent séparemment les trois composantes bleue, rouge et verte.

\subsection{Techniques précédentes}

La plupart des techniques précédentes sont encore applicables. Il fa..
```

Test

- Attention, JPEG même en qualité 100 ne restitue pas exactement la même image.
- On peut tester en partant d'une image PPM un peu particulière..

```
\uecuhpn{Cokssdttino}

On duqcje ibk dd!dmoprdtsds!bfs!gmafhq,!c(eru!P"dhpf"cet!scakfaww!ee!ohxfj
di‘qvd qgyfk!oovwamt psfkgrcykf"bpumdvr fntse $/%eu$347$. Oo oe
s&iotêqeuse kbh#qv&à!bpnptdsrds cgrmjcies gl!mltf‘ww de esis. En!dfgds- l_
rltparu bgr teegpfruft ed!bnposetqlol-potsdnmqqfsqfsunfhnbfe!awjmtn^t
TVA*!enmtqfqtgmvrìo‘rdmmdmt!ifq!vqmjrdqnnrs‘nucv‘jfvc-qpydf"dt sfrvb1

_pvdqfcrhpo{Tfbgpjnyfp#qoécédcoucs~

Jbsluq‘xs ccs tgagqgssgr!pqêeçedntes"ppmvep_ppg!arokjdackfp,!Kj g^
```

Son

Des méthodes similaires existent pour compresser le son

- $1D$ au lieu de $2D$.
- Même principe d'application de transformation (MPEG)
- Même principe d'application de filtre (PCM/ADPCM)