

# Compression d'images fixes (Avec perte)

INF4710 Introduction aux technologies  
multimédia

# Plan

- ▶ Compression avec perte
  - Espace des couleurs
  - Domaine transformé
    - Transformée KL
    - Transformée en cosinus discrète
    - Transformée en ondelettes

# Compression de l'espace des couleurs

- ▶ Tramage Floyd–Steinberg (dithering)
  - Appliqué sur les images couleurs et en tons de gris;
  - Réduit le nombre requis de couleurs pour représenter une image en utilisant un tramage;
  - Basé sur le principe de la diffusion des erreurs.

# Compression de l'espace des couleurs

## ▶ Tramage Floyd–Steinberg

### ◦ Algorithme:

- Pour chaque symbole (de haut en bas, et de gauche à droite)
  1. Sauvegarder l'ancienne valeur du symbole;
  2. Trouver la couleur la plus semblable dans la palette de couleur choisie;
  3. Assigner au symbole cette couleur;
  4. Calculer l'erreur
  5. Diffuser l'erreur aux pixels voisins selon la matrice

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 7/16 \\ 3/16 & 5/16 & 1/16 \end{bmatrix}$$

# Compression de l'espace des couleurs

## ▶ Tramage Floyd–Steinberg

- Exemple:

100 50 100 45  
200 10 70 80

- Trouver le tramage Floyd–Steinberg avec une quantification sur 1 bit.

## ▶ Exemple MATLAB:

- *CodageFloydSteinberg.m*

# Compression de l'espace des couleurs

- ▶ Tramage Floyd–Steinberg
  - On atteint le niveau de compression que l'on souhaite, mais la qualité de l'image diminue en conséquence.
  - On peut avoir au maximum un taux de compression de 8:1 -> 8 bits pour 1 bit

# Compression de l'espace des couleurs

- ▶ Sous-échantillonnage de la chrominance (Chroma subsampling)
  - Utilise le fait que le système visuel humain est plus sensible aux variations de luminosité qu'aux variations de couleur.
  - On peut donc compresser une image en consacrant plus de bits à la luminance qu'à la chrominance.
  - Il faut changer de système de couleurs, car RGB mélange la chrominance et la luminance dans chaque canal (R, G, et B).
  - Le système de couleur Y'C<sub>b</sub>C<sub>r</sub> concentre la luminance dans la composante Y'.

# Compression de l'espace des couleurs

- ▶ Sous-échantillonnage de la chrominance
  - Y'CbCr est calculé à partir des RGB après correction Gamma.
  - La correction gamma tient compte des caractéristiques des caméras et écrans (qui ne sont pas linéaires dans la capture et l'affichage de la luminance).

$$V_{sortie} = V_{entrée}^{\gamma}$$

$$Y' = 0.299R' + 0.587G' + 0.114B'$$

$$Cb = 128 - 0.1687R' - 0.3313G' + 0.5B'$$

$$Cr = 128 + 0.5R' - 0.4187G' - 0.08131B'$$

- $Cb$ : bleu – luminance
- $Cr$ : rouge - luminance

# Compression de l'espace des couleurs

- ▶ Sous-échantillonnage de la chrominance
  - Ensuite, le sous-échantillonnage est exprimé de la façon suivante:
    - Un ratio en trois parties  $J:a:b$  (par exemple 4:2:2) qui décrit le nombre d'échantillons de luminance et de chrominance dans une région conceptuelle qui est de  $J$  pixels de large et de 2 pixels de hauteur.
    - $J$  : l'échantillonnage de la référence horizontale (largeur de la région conceptuelle). Habituellement, 4.
    - $a$  : le nombre d'échantillons  $Cr$  et  $Cb$  dans la première rangée de  $j$  pixels.
    - $b$  : le nombre d'échantillons  $Cr$  et  $Cb$  dans la deuxième rangée de  $j$  pixels.

# Compression de l'espace des couleurs

- ▶ Sous-échantillonnage de la chrominance
  - Exemple:
    - Donnez l'échantillonnage et le taux de compression pour
      - 4:4:4
      - 4:2:2
      - 4:1:1
      - 4:2:0

# Compression de l'espace des couleurs

- ▶ Sous-échantillonnage de la chrominance
  - Utilisation:
    - 4:2:2: DVCPRO-HD, DVCPRO50 (Panasonic), XDCAM HD422 (Sony), DIGITAL-S (JVC)
    - 4:1:1: DVCPRO, DV, DVCAM
    - 4:2:0: MPEG, MPEG-2, H.262, H.264, MPEG-4, MPEG-4 AVC, HDV, AVCHD, VC-1, JPEG, MJPEG

# Compression par transformée

- ▶ Codage d'images par transformée
  - Famille de méthodes qui transforme une image de façon à en décorrélérer les données.
  - Par exemple, pour une image RGB, si on élimine B, l'apparence de l'image change complètement, car la couleur perçue est corrélée avec chaque composante R, G, et B. Il y a une relation de dépendance entre les composantes et la couleur perçue. Il y a aussi une dépendance au niveau spatial.
  - Si on souhaite enlever des composantes, il faut qu'elles soient indépendantes entre elles.

# Compression par transformée

- ▶ Codage d'images par transformée
  - Les méthodes de codage d'images par transformée expriment l'image dans d'autres bases (référentiels) où chaque composante est indépendante.
  - En faisant cette transformation, certaines composantes contiennent plus d'information que d'autres. On peut donc éliminer les composantes qui contiennent le moins d'information.

# Compression par transformée

- ▶ Transformée KL (Karhunen–Loève)
  - Idée: Trouver la base optimale, celle qui concentre l'information dans le moins de composantes possibles.
  - Méthode: Calculer la covariance des composantes originales (exemple: R,G,B), et trouver les vecteurs propres de la matrice de covariance. Les vecteurs propres correspondent à des bases orthogonales où les composantes sont non-corrélées.

# Compression par transformée

## ▶ Transformée KL

- Détails de la méthode pour les composantes R,G,B

1. Trouver la moyenne des R,G,B pour obtenir le vecteur

$$\vec{m}_{rgb} = \begin{bmatrix} \bar{R} \\ \bar{G} \\ \bar{B} \end{bmatrix}$$

2. On calcule la covariance des  $n$  (tous les) RGB dans l'image.

$$Cov_{rgb} = \sum_{i=1}^n (\vec{RGB}_i \cdot \vec{RGB}_i^T) - (\vec{m}_{rgb} \cdot \vec{m}_{rgb}^T)$$

# Compression par transformée

## ▶ Transformée KL

3. On trouve les vecteurs propres de  $Cov_{rgb}$

$$Cov_{rgb} \cdot v = \lambda v$$

$v$  correspond aux vecteurs propres, et  $\lambda$  aux valeurs propres. Les valeurs propres indiquent l'énergie associée à chaque base (vecteur propre).

4. On exprime ensuite chaque vecteur RGB de l'image dans la nouvelle base en conservant les vecteurs propres ayant le plus d'énergie.

$$\vec{Y}_i = \begin{bmatrix} v(1) \\ v(2) \\ v(3) \end{bmatrix} \cdot (\vec{R}GB_i - \vec{m}_{rgb})$$

# Compression par transformée

## ▶ Transformée KL

- Si on élimine les vecteurs propres de faibles énergies, les  $Y_i$  auront des composantes à zéro. Donc, au lieu de stocker 3 nombres, on peut en stocker seulement 2 ou 1. D'où la compression.
- En faisant la transformation des  $Y_i$  vers les  $RGB_i$ , on récupère une représentation avec perte de l'image.
- Inconvénient: La base change pour chaque image. Demande beaucoup de calculs.

## ▶ Exemple MATLAB:

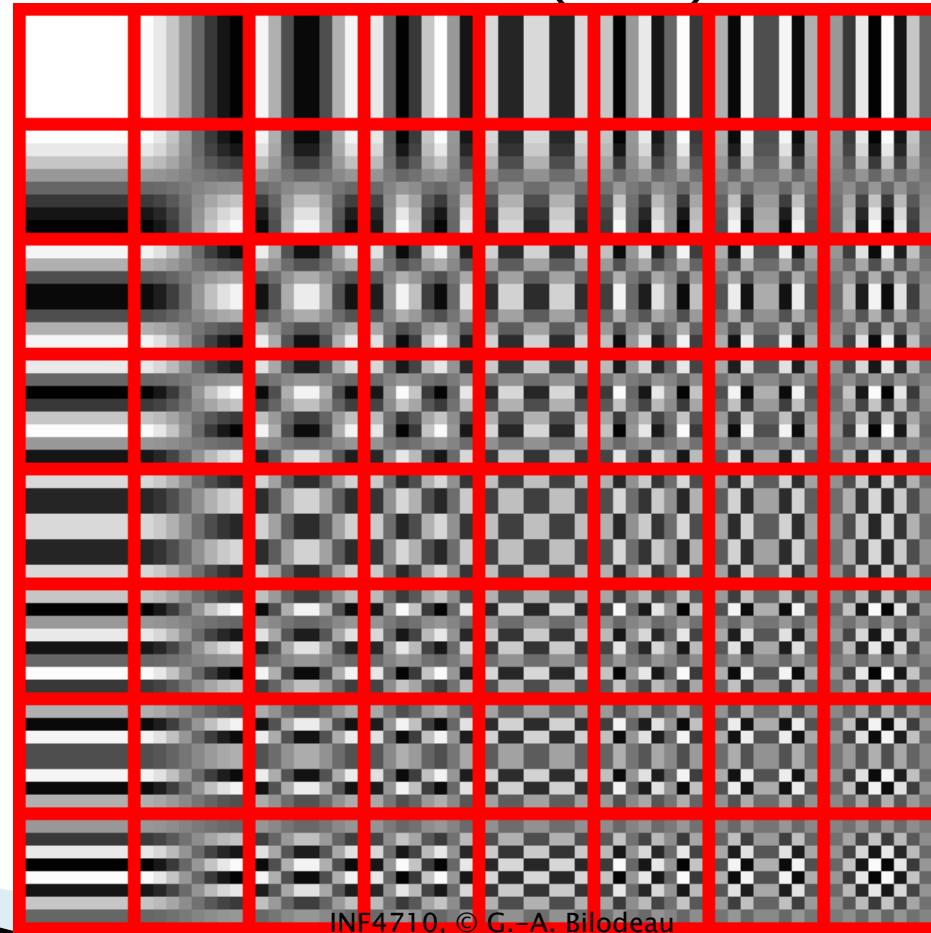
- *TransformeeKL.m*

# Compression par transformée

- ▶ Transformée en cosinus discrète (Discrete Cosine Transform, DCT)
  - Cette transformée vise aussi à exprimer des composantes corrélées dans une base où elles sont moins corrélées.
  - Cette transformée n'est pas optimale, mais se calcule rapidement et est séparable (on peut traiter les deux dimensions d'une image de façon indépendante, et donc en parallèle). Les bases ne changent pas.
  - Comme son nom l'indique, les bases sont des cosinus à des fréquences différentes. Aucun cosinus ne peut être créé par une combinaison d'autres cosinus. Ce qui décorrèle les données.

# Compression par transformée

- ▶ Transformée en cosinus discrète
  - Voici une illustration des bases en 2D ( $N=8$ ).



# Compression par transformée

## ▶ Transformée en cosinus discrète

- Elle est calculée par:

$$C(u, v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos\left(\frac{\pi(2x+1)u}{2N}\right) \cos\left(\frac{\pi(2y+1)v}{2N}\right)$$

- où  $\alpha(u)$  et  $\alpha(v)$  sont données par

$$\begin{aligned} \sqrt{\frac{1}{N}} & \text{ si } u = 0 \quad \text{ ou } \quad v = 0 \\ \sqrt{\frac{2}{N}} & \text{ si } u \neq 0 \quad \text{ ou } \quad v \neq 0 \end{aligned}$$

- $f(x, y)$  est la valeur d'un pixel,  $u$  et  $v$  sont dans l'intervalle de 0 à  $N-1$ , et  $N$  est la taille du bloc à transformer.

# Compression par transformée

- ▶ Transformée en cosinus discrète
  - Les  $C(u,v)$  correspondent à la proportion de chaque cosinus pour représenter le bloc de données.
  - Habituellement, seulement quelques  $C(u,v)$  sont grands.

# Compression par transformée

- ▶ Transformée en cosinus discrète
  - La transformée inverse est calculée par:

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v)C(u, v) \cos\left(\frac{\pi(2x+1)u}{2N}\right) \cos\left(\frac{\pi(2y+1)v}{2N}\right)$$

# Compression par transformée

## ▶ Format JPEG

- Utilise la DCT et le codage Huffman pour faire la compression de l'image.
- Constitué d'une séquence de segments. Exemple:
  - SOI (0xFF,0xD8): Début de l'image
  - SOF0 (0xFF,0XC0): Hauteur, largeur, nombre de composantes, et le type de sous-échantillonnage de la chrominance utilisé.
  - DHT (0xFF,0XC4): Tables de Huffman utilisées.
  - DQT (0xFF,0XDB): Tables de quantification utilisées.
  - SOS (0xFF,0XDA): Les données de l'image.
  - etc.

# Compression par transformée

## ▶ Format JPEG

- Encodage:
  1. Conversion de RGB vers Y'CbCr
  2. Sous-échantillonnage de la chrominance
  3. Décomposition en bloc de 8 x 8, et application de la DCT
  4. Quantification: moins de bits pour les hautes fréquences que les pour basses fréquences. Œil moins sensible aux hautes fréquences. Détermine le niveau de compression.
  5. Encodage avec Huffman et RLE pour la matrice arrangée en zigzag.

# Compression par transformée

## ▶ Format JPEG

- Encodage:
  - Quantification

DCT

$$\begin{bmatrix} -415 & -33 & -58 & 35 & 58 & -51 & -15 & -12 \\ 5 & -34 & 49 & 18 & 27 & 1 & -5 & 3 \\ -46 & 14 & 80 & -35 & -50 & 19 & 7 & -18 \\ -53 & 21 & 34 & -20 & 2 & 34 & 36 & 12 \\ 9 & -2 & 9 & -5 & -32 & -15 & 45 & 37 \\ -8 & 15 & -16 & 7 & -8 & 11 & 4 & 7 \\ 19 & -28 & -2 & -26 & -2 & 7 & -44 & -21 \\ 18 & 25 & -12 & -44 & 35 & 48 & -37 & -3 \end{bmatrix}$$

DCT  
quantifiée

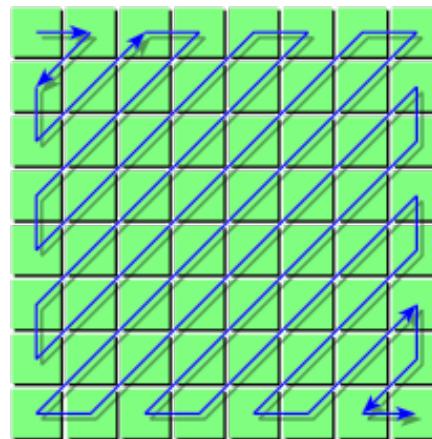
Matrice de quantification

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

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -3 & 4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

# Compression par transformée

- ▶ Format JPEG
  - Encodage RLE et Huffman



- |     |    |   |    |    |    |    |    |   |    |   |   |   |   |   |    |   |    |
|-----|----|---|----|----|----|----|----|---|----|---|---|---|---|---|----|---|----|
| -26 | -3 | 0 | -3 | -2 | -6 | 2  | -4 | 1 | -3 | 1 | 1 | 5 | 1 | 2 | -1 | 1 | -1 |
| 2   | 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  |

# Compression par transformée

- ▶ Format JPEG
  - Encodage RLE et Huffman
  - $-26 -3 0 -3 -2 -6 2 -4 1 -3 1 1 5 1 2 -1 1 -1 2 0 0 0 0 0$   
 $-1 -1 0$   
 $0 0 0 0 0 0 0 0 0$
- ▶ La composante DC n'est pas codée avec RLE et Huffman.
- ▶ Le codage RLE et Huffman produit la séquence suivante qui code les valeurs  $V_i$  différentes de zéro. (nombre de zéros précédent  $V_i$ , nb bits pour représenter  $V_i$  selon table Huffman)( $V_i$ ):
  - $(0, 2)(-3); (1, 2)(-3); (0, 2)(-2); (0, 3)(-6); (0, 2)(2); (0, 2)(-4); (0, 1)(1); (0, 2)(-3); (0, 1)(1); (0, 1)(1); (0, 3)(5); (0, 1)(1); (0, 2)(2); (0, 1)(-1); (0, 1)(1); (0, 2)(2); (5, 1)(-1); (0, 1)(-1); (0, 0)$
- ▶ Note les couples (a,b) sont sur 8 bits. a et b sur 4 bits.  
Code spécial quand 16 zéros ou plus.

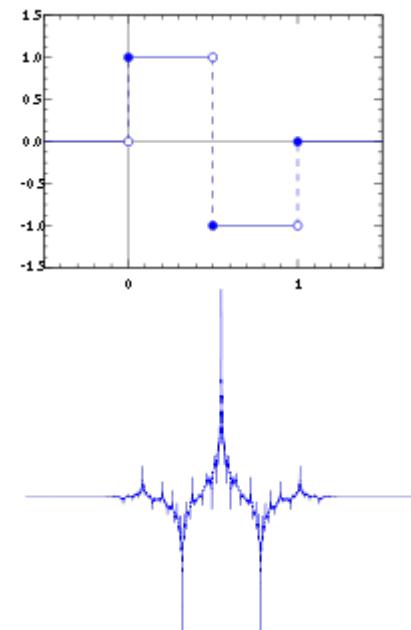
# Compression par transformée

- ▶ Format JPEG
  - Encodage de la valeur DC
    - Par codage prédictif.
- ▶ Exemple MATLAB:
  - *TransformeeDCT.p* (P-code)

# Compression par transformée

## ▶ Transformée en ondelettes (wavelet transform)

- Pour cette transformée, plutôt qu'utiliser des cosinus à différentes fréquences comme bases, on utilise de petites ondes (ondelettes) de fréquences variées et de durées limitées.
- Une variété d'ondelettes existe:
  - Haar
  - Daubechies
  - Cohen–Daubechies–Feauveau



# Compression par transformée

- ▶ Transformée en ondelettes
  - Elle est calculée par

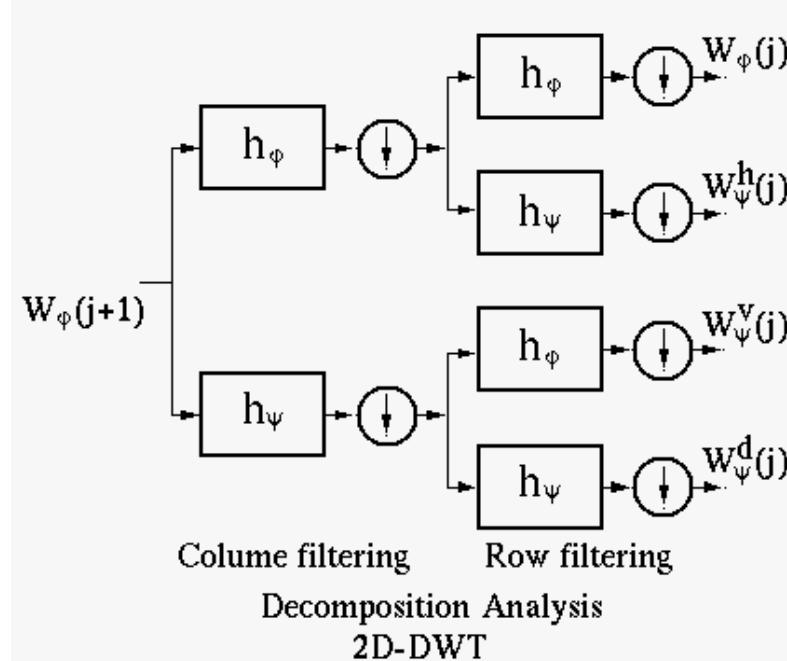
$$W_\varphi(j_0, m, n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \varphi_{j_0, m, n}(x, y)$$

$$W_\psi(j, m, n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \psi_{j, m, n}^i(x, y) \quad i = \{H, V, D\}$$

- Avec  $\psi$  les fonctions d'ondelette et  $\varphi$  la fonction de mise à l'échelle

# Compression par transformée

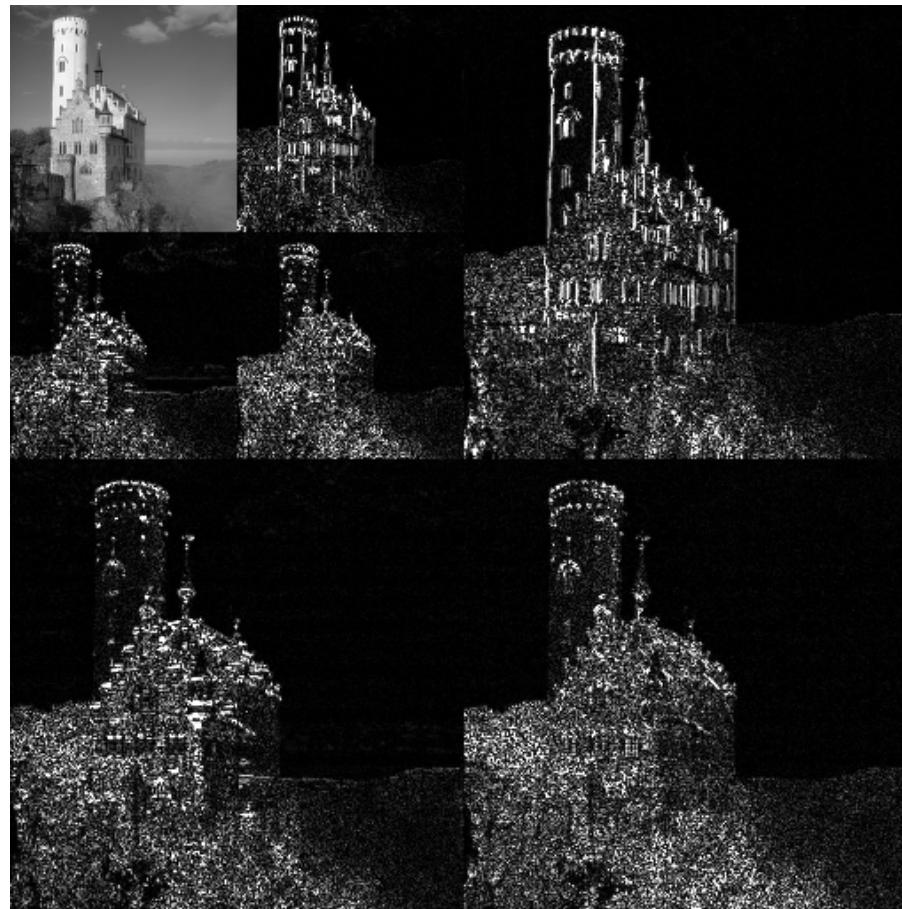
- ▶ Transformée en ondelettes
  - Calculée sous forme de banques de filtres:



- $h\psi$  et  $h\phi$ , les vecteurs et coefficients des ondelettes.

# Compression par transformée

## ▶ Transformée en ondelettes



# Compression par transformée

- ▶ Transformée en ondelettes
  - Meilleure performance que DCT
    - Moins d'artéfacts
    - Moins d'effets de bloc
    - Permet de prévoir le taux de compression
    - Reconstruction progressive de l'image
    - Par contre, effets de flou

# Compression par transformée

## ▶ Format JPEG2000

- Utilise la transformée en ondelettes
- Étapes d'encodage:
  1. Transformation vers Y'CbCr ou YUV. Facultatif: Sous-échantillonnage de la chrominance.
  2. Optionnel: Découpage en tuiles (tiles) pour une plus faible utilisation de la mémoire.
  3. Transformée en ondelettes
  4. Quantification
  5. Codage par blocs avec codage arithmétique.