

## I. Compression par DWT de Haar

### 1. Découpage d'une image en blocs 8x8



- Déclarer une fonction **decoupageEnBlocs** définie de la manière suivante :
  - **Entrée** : une image noir et blanc de définition  $8n \times 8n$
  - **Sortie** : la liste des blocs  $8 \times 8$
- Déclarer une fonction **assemblageDesBlocs** définie de la manière suivante :
  - **Entrée** : une liste de blocs  $8 \times 8$
  - **Sortie** : l'image noir et blanc reconstituée

### 2. Transformation d'une ligne d'un bloc

La transformée discrète en ondelette (DWT) de Haar décompose un signal discret  $x = (x_0, \dots, x_{2N-1})$  en deux composantes :

- un signal d'approximation  $s = (s_0, \dots, s_{N-1})$  où  $s_k = \frac{x_{2k} + x_{2k+1}}{2}$
- un signal de détails  $d = (d_0, \dots, d_{N-1})$  où  $d_k = \frac{x_{2k} - x_{2k+1}}{2}$

En continuant à décomposer le signal d'approximation  $s$ , on obtient un signal décomposé en plusieurs échelles :

- échelle 1 :  $x^{(1)} = (s_0^{(1)}, s_1^{(1)}, s_2^{(1)}, s_3^{(1)}, d_0^{(1)}, d_1^{(1)}, d_2^{(1)}, d_3^{(1)})$
- échelle 2 :  $x^{(2)} = (s_0^{(2)}, s_1^{(2)}, d_0^{(2)}, d_1^{(2)}, d_0^{(1)}, d_1^{(1)}, d_2^{(1)}, d_3^{(1)})$
- échelle 3 :  $x^{(3)} = (s_0^{(3)}, d_0^{(3)}, d_0^{(2)}, d_1^{(2)}, d_0^{(1)}, d_1^{(1)}, d_2^{(1)}, d_3^{(1)})$



- Déterminer la matrice  $W_1$  telle que  $x^{(1)} = x \times W_1$ .
- Déterminer la matrice  $W_2$  telle que  $x^{(2)} = x^{(1)} \times W_2$ .
- Déterminer la matrice  $W_3$  telle que  $x^{(3)} = x^{(2)} \times W_3$ .
- En déduire que  $x^{(3)} = x \times W$  avec
 
$$W = \begin{pmatrix} 1/8 & 1/8 & 1/4 & 0 & 1/2 & 0 & 0 & 0 \\ 1/8 & 1/8 & 1/4 & 0 & -1/2 & 0 & 0 & 0 \\ 1/8 & 1/8 & -1/4 & 0 & 0 & 1/2 & 0 & 0 \\ 1/8 & 1/8 & -1/4 & 0 & 0 & -1/2 & 0 & 0 \\ 1/8 & -1/8 & 0 & 1/4 & 0 & 0 & 1/2 & 0 \\ 1/8 & -1/8 & 0 & 1/4 & 0 & 0 & -1/2 & 0 \\ 1/8 & -1/8 & 0 & -1/4 & 0 & 0 & 0 & 1/2 \\ 1/8 & -1/8 & 0 & -1/4 & 0 & 0 & 0 & -1/2 \end{pmatrix}$$

### 3. Transformation d'un bloc

En répétant la transformation précédente pour toutes les lignes d'un bloc  $A$ , il vient :

$$B = AW$$

Puis en agissant de la même manière sur toutes les colonnes du bloc transformé  $B$ , il vient :

$$C = W^t B = W^t AW$$

où  $W^t$  désigne la matrice transposée de  $W$  définie par  $W_{i,j}^t = W_{j,i}$ .



Déclarer une fonction **transformationBloc** définie de la manière suivante :

- **Entrée** : un bloc  $8 \times 8$
- **Sortie** : le bloc transformé

### 4. Récupération d'un bloc transformé

Pour récupérer le bloc  $A$  à partir du bloc transformé  $C$ , il suffit d'utiliser la matrice inverse<sup>[1]</sup> de  $W$  :

$$A = (W^t)^{-1} C W^{-1} = (W^{-1})^t C W^{-1}$$



#### A propos de la matrice transposée

- La transposition ne change pas le déterminant
- Si  $W$  est inversible, alors  $W^t$  l'est aussi et  $(W^t)^{-1} = (W^{-1})^t$



Déclarer une fonction **recuperationBloc** définie de la manière suivante :

- **Entrée** : un bloc transformé  $8 \times 8$
- **Sortie** : le bloc initial

### 5. Utilisation de l'algèbre linéaire pour améliorer le procédé



#### A propos du produit scalaire

Le produit scalaire<sup>a</sup> dans  $\mathbb{R}^n$  défini par  $(u_1, \dots, u_n) \cdot (v_1, \dots, v_n) = \sum_{i=1}^n u_i v_i$  permet de :

- définir deux vecteurs orthogonaux :

$$(u_1, \dots, u_n) \perp (v_1, \dots, v_n) \Leftrightarrow (u_1, \dots, u_n) \cdot (v_1, \dots, v_n) = 0$$

- mesurer la longueur d'un vecteur en définissant sa norme :

$$\|(u_1, \dots, u_n)\| = \sqrt{(u_1, \dots, u_n) \cdot (u_1, \dots, u_n)} = \sqrt{\sum_{i=1}^n u_i^2}$$

- mesurer l'angle entre deux vecteurs  $u = (u_1, \dots, u_n)$  et  $v = (v_1, \dots, v_n)$  en définissant le cosinus de l'angle :

$$\cos(u, v) = \frac{u \cdot v}{\|u\| \|v\|}$$

a. Généralisation du cas  $n = 2$  que vous connaissez déjà

[1]. On pourra vérifier l'inversibilité de  $W$  en calculant son déterminant

## A propos des bases orthonormales (BON)

- Une famille de vecteurs est orthogonale si les vecteurs sont deux à deux orthogonaux.
- Si une famille de vecteurs non nuls est orthogonale, alors elle est libre.
- Une famille orthogonale de  $n$  vecteurs non nuls est une base dite orthogonale (BO) de  $\mathbb{R}^n$ .  
Si de plus les vecteurs sont unitaires c'est à dire de norme 1, alors la base est dite orthonormale (BON).
- La base canonique de  $\mathbb{R}^n$  est une BON.
- La matrice de passage  $P$  d'une BON vers une autre BON vérifie :

$$P^{-1} = P^t$$

Une matrice dont l'inverse est la transposée est une matrice dite orthogonale.



1. Montrer que  $W$  est la matrice de passage de la base canonique vers une BO de  $\mathbb{R}^8$ .
2. En déduire que  $W$  est bien inversible.
3. A-t-on  $W^{-1} = W^t$  ?
4. Comment peut-on transformer  $W$  pour que  $W^{-1} = W^t$  ?

En notant encore  $W$  la nouvelle matrice, il vient :

$$C = W^t A W \quad \text{et} \quad A = W C W^t$$



1. Déclarer deux nouvelles fonctions **transformationBloc2** et **recuperationBloc2** d'après ce qui précède
2. Comparer les temps d'exécution de **recuperationBloc** et **recuperationBloc2** à l'aide de la méthode `time.clock()`



1. Déclarer une fonction **compressionHaar** définie de la manière suivante :
  - **Entrée** : une image noir et blanc de définition  $8n \times 8n$
  - **Sortie** : l'image compressée
2. Déclarer une fonction **decompressionHaar** définie de la manière suivante :
  - **Entrée** : une image compressée
  - **Sortie** : l'image initiale



- Des zéros apparaissent dans le bloc transformé<sup>a</sup> qui occupe alors moins de place que le bloc initial.  
Pour augmenter le nombre de valeurs nulles et donc améliorer la compression, on peut utiliser un seuillage, mais une partie de l'image initiale est définitivement perdue...
- Dans le même style, vous avez la [compression JPEG](#) avec tous les détails [ici](#)!

<sup>a</sup>. Traduisant une faible variation entre les pixels

## II. Compression par SVD

On note  $A$  la matrice de l'image noir et blanc à compresser.

### 1. Décomposition en valeurs propres d'une matrice (réelle) symétrique

#### Théorème spectral

Si  $A$  est symétrique<sup>a</sup>, alors il existe une matrice orthogonale  $P$  et une matrice diagonale  $D$  telles que :

$$A = PDP^t$$

a.  $A^t = A$



Il s'agit en fait d'une diagonalisation particulière : la base formée de vecteurs propres est orthonormale!

### 2. Décomposition en valeurs singulières d'une matrice quelconque

La condition  $A$  symétrique étant très restrictive, c'est la généralisation du théorème spectral que nous allons utiliser :

#### Théorème de Eckart et Young

Il existe des matrices orthogonales  $U, V$  et une matrice  $S$  nulle partout sauf sur sa diagonale principale qui contient les valeurs singulières de  $A$  dans l'ordre décroissant telles que :

$$A = USV^t$$

Plus précisément, en notant  $r$  le rang de  $A$  de taille  $m \times n$  et  $U_i, V_i$  les colonnes de  $U, V$ , la décomposition s'écrit :

$$A = \begin{pmatrix} U_1 & \dots & U_r & \dots & U_m \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \ddots & & \\ & & \sigma_r & \ddots \\ \vdots & & \ddots & 0 \\ 0 & \dots & & \ddots & 0 \\ 0 & \dots & 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^t \\ \vdots \\ V_r^t \\ \vdots \\ V_n^t \end{pmatrix}$$

ou encore

$$A = \sigma_1 U_1 V_1^t + \sigma_2 U_2 V_2^t + \dots + \sigma_r U_r V_r^t$$



- Ce théorème n'a aucune condition restrictive, pas même  $A$  carrée!
- Le problème est d'obtenir cette décomposition, mais Python s'en charge grâce à la fonction `linalg.svd(mat)`!
- On peut améliorer la compression en ne gardant que les premiers termes de la somme (les derniers sont "petits") mais, comme pour le seuillage, une partie de l'image est alors définitivement perdue...



Déclarer une fonction **compressionSVD** définie de la manière suivante :

- **Entrée** : une image noir et blanc, un niveau de compression (nombre de termes dans la somme)
- **Sortie** : l'image compressée