IHT3

2D and 3D Visual Data Compression

Control subjects

1. Transformées Cosinus et Sinus (Mehdi, Zhejie et Arthur)
2. Transformée de KL (Mathis et Leeven)
3. Transformée en ondelettes (Malo et Pierre)
4. Quantification vectorielle LBG
5. Codage de séquences d'images (Nicolas, Martin, Pâcome, Matheus)
6. Codage ZTW
7. Quantificateur Max-Lloyd (Théophile, Melissa et Victor)
8. Codage prédictif AR d'images couleur (Kessel, Marion, Wei, Yujia)

1. Implantation des transformées de **Sinus, Cosinus**

1.a

Implantez une **transformée de Cosinus discrète**, en partant de la définition:

a. transformée directe
b. transformée inverse

Prototypes:
int dct2d(double **x; double **xt, int Width, int Height);
int dct2d_inv(double **x; double **xrec, int Width, int Height);

// x -l'image initiale
// xt - image transformée
// xrec - l'image reconstruite
// Width, Height : taille de l'image

Appliquez tout d'abord la transformée sur: l'image entière. Quantifier uniformément les coefficients de la transformée (en gardant le coefficient DC inchangé) et calculez l'entropie des coefficients quantifiés. Les pas de quantification suivant seront utilisés: *delta* = 1, 5, 10, 20, 50.

Effectuez une partition de l'image en blocs carrées de taille (N_B x N_B). Dans le cas où l'image n'est pas carrée, construisez les blocs de bords en ajoutant des zéros. Appliquez ensuite la transformée Cosinus à chaque bloc. Quantifier uniformément les coefficients (à l'exception du coefficient DC de chaque bloc) et calculez l'entropie des coefficients sur l'ensemble de l'image. Comparez, pour $N_B = 8$, $N_B = 16$, avec celle obtenue précédemment, lorsque la transformée a été effectuée sur l'ensemble de l'image.

1.b Implantez une **transformée Sinus discrète**, en partant de la définition:

a. transformée directe
b. transformée inverse

Prototypes:
int tsin2d(double **x; double **xt, int Width, int Height);
int tsin2d_inv(double **xt; double **xrec, int Width, int Height);

// x -l'image initiale
// xt - image transformée
// xrec - l'image reconstruite
// Width, Height : taille de l'image

Utilisez la relation entre la transformée sinus et la transformée de Fourier discrète, pour implanter une version rapide de la transformée de sinus (via la transformée de Fourier rapide) dans le cas où les dimensions de l'image sont pouvoirs entiers de 2.

Implantez les mêmes fonctions que celles du sujet 2 pour partition en bloc, calcul de la transformée sur chaque bloc, quantification, etc. Comparez les résultats avec ceux obtenus avec la transformée de Cosinus discrète.

2. Implantez une **transformée de Karhunen-Loève**

a. Calcul des matrices de covariance $r(k,l) = E\{u(m, n)\, u(m-k, n-l)\}$
b. Calcul de la transformée directe
c. Calcul de la transformée inverse

Prototypes (C++ - like):
int tkl2d(double **x; double **xt, int Width, int Height, int N_B, double **FI);
int tkl2d_inv(double **xt; double **xrec, int Width, int Height, int N_B, double **FI);

// x -l'image initiale
// xt - image transformée
// xrec - l'image reconstruite
// Width, Height : taille de l'image
// FI - la matrice de la transformée
// N_B - la taille des blocs carrés

La transformé sera implanté par blocs de taille 8x8 et 16x16.

Etapes de calcul de la transformée de KL directe :

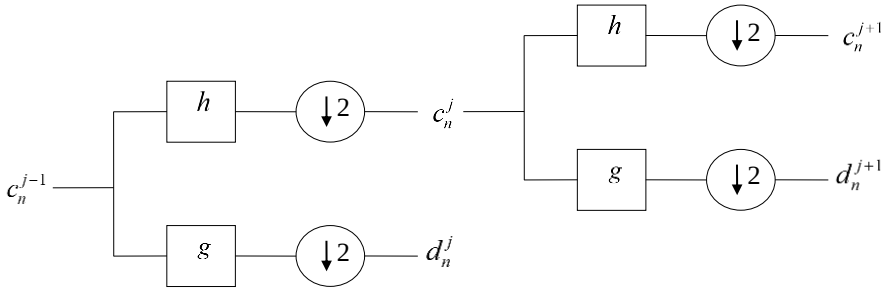
- Réalisez une partition de l'image en blocs de dimension N_B x N_B
- Balayez chaque bloc (*raster-scan*) et réorganisez les valeurs des pixels dans des vecteurs de longueur N_B^2.
- Calculez, sur l'ensemble des vecteurs ainsi déterminés, la matrice de covariance, de taille (N_B^2 x N_B^2)
- Diagonalisez la matrice de covariance et calculez ses vecteurs et valeurs propres.
- Triez les valeurs propres par ordre croissant et construisez la matrice de la transformée FI, ayant comme colonnes les vecteurs propres associés
- Calculez, pour chaque vecteur sa transformée KL, en le multipliant (à gauche) avec Fi^t.
- Réorganisez les vecteurs transformés en blocs carrés de dimensions N_B x N_B pour déterminer l'image transformée.

Pour chaque bloc, mettez à zéro les n_c coefficients de la transformées correspondant aux plus petites valeurs propres et reconstruisez l'image. Déterminez expérimentalement le nombre n_c de coefficients que l'on peut mettre à zéro sans affecter drastiquement la qualité de l'image reconstruite.

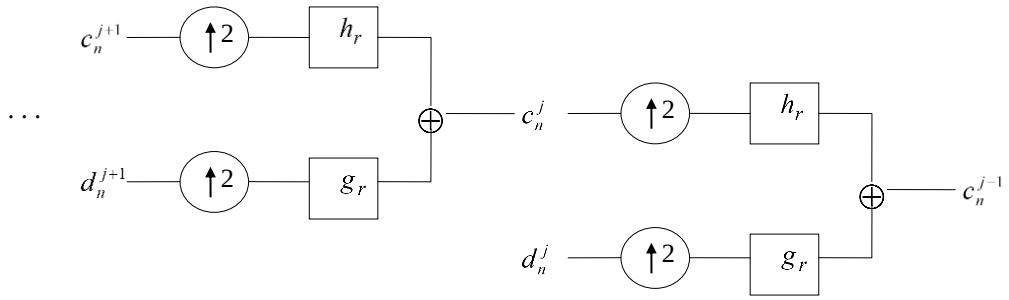
3. Implantez une **transformée en ondelettes,** spécifiée par les filtres d'analyse h et g (de longueurs N_h et N_g respectivement) et les filtres de synthèse (reconstruction) hr, gr (de longueurs N_{hr}, N_{gr}, respectivement):

a. transformée directe
b. transformée inverse

L'algorithme de filtrage récursif (cas 1D) est illustré Figure 1.



1.a. ANALYSE (transformée directe).



1.b. SYNTHESE (transformée inverse).

Figure 1. Schéma bloc de la transformée en ondelettes directe et inverse.

Prototypes:
int ondel2d(double **x; double **xt, int niv_resol, int Width, int Height, double *h, double *g, int N_h, int N_g);
int ondel2d_inv(double **xt, int niv_resol, double **xrec, int Width, int Height, double *hr, double *gr, int N_{hr}, int N_{gr});

// x -l'image initiale
// xt - image transformée
// xrec - l'image reconstruite
// Width, Height : taille de l'image
// niv_resol le nombre de niveaux de résolution

Personnaliser les filtres pour la transformée de Haar :

$$h = (\boxed{\frac{1}{\sqrt{2}}}, \frac{1}{\sqrt{2}}), \qquad g = (\boxed{\frac{1}{\sqrt{2}}}, -\frac{1}{\sqrt{2}})$$

$$h_r = (\boxed{\frac{1}{\sqrt{2}}}, \frac{1}{\sqrt{2}}), \qquad g_r = (\boxed{\frac{1}{\sqrt{2}}}, -\frac{1}{\sqrt{2}})$$

Les valeurs encadrées correspondent à l'échantillon d'index 0 du filtre.

Quantifier uniformément les coefficients de la transformée (en gardant le coefficient DC inchangé) et calculez l'entropie des coefficients quantifiés. Les pas de quantification suivant seront utilisés: delta = 1, 5, 10, 20, 50.

Indications :

- La transformée 2D sera implantée à partir de la transformée 1D, en exploitant la propriété de séparabilité (transformée 1D sur chaque ligne de l'image, suivie d'une transformée 1D sur chaque colonne).

Réaliser une interface (java) permettant :

- d'ouvrir une image,
- de lancer les procédures de calcul des transformées en ondelettes directe et inverse,
- de visualiser les transformées en ondelettes,
- d'entrer le pas de quantification,
- de sélectionner avec la souris un ensemble de sous-bandes et de mettre à zéro leurs coefficients,
- de visualiser l'image reconstruite

4. Implantez une **procédure de quantification vectorielle,** en utilisant l'algorithme LBG (Lindo-Buzo-Gray) pour construire le dictionnaire des prototypes :

Le dictionnaire de prototypes est construit à partir d'un ensemble d'images *d'apprentissage*. Chaque image est décomposée en bloc carrés de dimensions NxN (tailles recomandées : 8x8, 16x16). Chaque bloc est mis sous la forme d'un vecteur de dimesnion N^2, obtenu en balayant *raster-scan* ses pixels.

Les étapes de **l'algorithme LBG** sont les suivantes :

1. Déterminer tout d'abord le dictionnaire composé d'un seul vecteur prototype, noté C_0^0. Le critère étant la minimisation de l'erreur quadratique moyenne, ce vecteur sera le centre de gravité de l'ensemble des blocs de l'image.
2. Partager ce vecteur en deux vecteurs, notés C_0^1 et C_1^1 , donnés par :

$$C_0^1 = C_0^0 + \varepsilon$$
$$C_1^1 = C_0^0 - \varepsilon,$$

où ε est un *vecteur aléatoire de perturbation*, suffisamment petit.

3. On classe tous les vecteurs de la base d'apprentissage relativement à ces deux vecteurs en utilisant le critère de la distance euclidienne minimale. On calcule les centres de gravité des deux classes ainsi obtenues qui remplaceront les anciens prototypes C_0^1 et C_1^1 et on réitère le procédé jusqu'à ce que la décroissance de la distorsion moyenne devienne inférieure à un seuil pré-défini.
4. Partition de chaque nouveau vecteur obtenu en deux, conformément à l'étape 2 et réitération de l'étape 3. L'algorithme s'arrête lorsque le nombre désiré de vecteurs est obtenu.

Remarquons que de l'algorithme LBG résultera un nombre de vecteurs prototype égal à une puissance entière de 2.

Une fois le dictionnaire de prototypes déterminé, la quantification de l'image est finalement réalisée en remplaçant chaque bloc de l'image par son prototype le plus proche (au sens de la distance euclidienne).

Version : application du même algorithme LBG sur des images couleurs et sur des blocs de taille triviale (1 x 1) pour déterminer un ensemble de couleurs dominantes.

Prototypes:
int VQ(double **x; double **xq; int Width, int Height, double *histo, double *seuils, double *rec, double ***dico, int NL);

```
// x -l'image initiale
// xq - l'image quantifiée
// Width, Height : taille de l'image
// BW, BH - dimensions horizontale et resp. verticale des blocs
// NP - le nombre de blocs prototypes
// dico : le dictionnaire (l'ensemble de vecteurs prototype)
```

5. Implantez une **procédure de codage/décodage de séquences d'images**, fondée sur des méthodes de codage par transformée :

a. Prédiction temporelle avec et sans estimation/compensation de mouvement : La première trame est codée en mode "intra" (sans prédiction temporelle), les suivantes en mode "prédictif" (la trame (t) est prédite à partir de la trame (t-1)).
b. Transformée DCT/blocs (8x8) des erreurs de prédiction. Les coefficients de la DCT seront quantifiés selon la règle suivante :

$$\hat{f}_q(u,v) = \frac{\dfrac{\hat{f}(u,v)\cdot 16}{w[u][v]} - k\cdot Q_s}{2Q_s},$$

où

- $k = \begin{cases} 0, & \text{pour les blocs codés en mode intra} \\ signe\{\hat{f}(u,v)\}, & \text{pour les blocs codés en mode prédictif} \end{cases}$,
- $w[u][v]$ représente la matrice de pondération :

8	17	18	19	21	23	25	27
17	18	19	21	23	25	27	28
20	21	22	23	24	26	28	30
21	22	23	24	26	28	30	32
22	23	24	26	28	30	32	35
23	24	26	28	30	32	35	38
25	26	28	30	32	35	38	41
27	28	30	32	35	38	41	45

- Q_s représente la taille (échelle) du quantificateur, paramètre prenant valeurs entières dans l'intervalle [1, 31].

c. Prédiction DPCM ligne par ligne des coefficients DC de chaque transformée DCT.
d. Codage prédictif (DPCM) des vecteurs de mouvement.
e. Estimation des entropies des trois sources à coder (coefficients DCT, erreurs de prédiction des vecteurs de mouvement, erreurs de prédiction des coefficients DC),

Les coefficients de prédiction pour les étapes c et d sont illustré ci-dessous :

0	0.5
0.5	x

Une méthode de *block matching* sera utilisée pour la compensation de mouvement : pour chaque bloc dans la trame t, on détermine un unique vecteur de déplacement (u, v), défini comme :

$$(u,v) = \arg\min_{(i,j)}\left\{\sum_{k=i_0}^{i_0+7}\sum_{l=j_0}^{j_0+7}\left|x^t(k,l) - x^{t-1}(k-i,l-j)\right|\right\}, \qquad i,j = -16,\ldots,16,$$

où (i_0, j_0) sont les coordonnées du coin supérieur gauche du bloc considéré, et x^t est la trame t de la séquence.

La prédiction d'un pixel (i, j) de la trame t, appartenant à un bloc de vecteur de déplacement (u, v) est donnée par la relation suivante :

$$\hat{x}^t(i, j) = x^{t-1}(i - u, j - v)$$

Remarques :
1. Le codage prédictif sans compensation de mouvement est équivalent à un codage avec compensation de mouvement avec $(u, v) = (0, 0)$ pour chaque bloc de l'image.
2. Effectuez toujours la prédiction à partir des images reconstruites, et non à partir des images initiales.

---

6. [**Codage ZTW**] Implantez une méthode de codage ZTW, décrite ci-dessous.

Considérez les relations naturelles de descendance entre les coefficients d'une décomposition multirésolution en ondelettes, illustrées Figure 1 pour une décomposition selon deux niveaux de résolution.

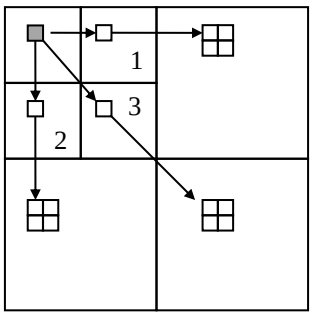


Figure 1. Relations de descendance entre les coefficients de la transformée en ondelettes.

A chaque coefficient de coordonnées (i, j) dans une sous-bande donnée, à l'exception de la sous-bande correspondant aux basses fréquences, on associe donc 4 coefficients enfant dans la sous-bande supérieure de même orientation. Les coefficients de la sous-bande de basses fréquences ont uniquement 3 descendants directs. Ces relations de descendance sont résumées dans le Tableau 1.

		Coordonnées parent	**Coordonnées descendants**
Sous-bande 0		(i, j)	(i+N_0, j), (i, j+N_0), (i+N_0, j+N_0)
Sous-bande 1-3		(i, j)	(2i, 2j), (2i+1, 2j), (2i, 2j+1), (2i+1, 2j+

Tableau 1. Définition des relations de descendance entre les coefficients de la transformée.

1. On détermine tout d'abord une valeur seuil, notée T et donnée par :

$$\mathrm{T} = \frac{1}{2} \max_{(i,j)} \left\{ \left| xt(i,j) \right| \right\}$$

2. On effectue un balayage des coefficients. Dans chaque sous-bande, les coefficients sont balayés selon un *raster-scan*. L'ordre de parcours des sous-bandes est illustré Figure 2.

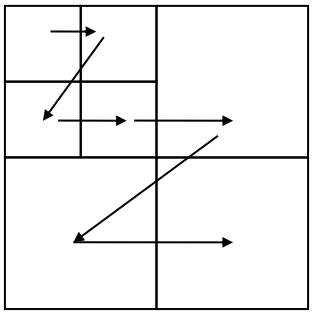


Figure 2. Ordre de parcours des sous-bandes.

Un pixel sera considéré comme *significatif* par rapport à un seuil T si sa valeur absolue est plus grande que T, et comme *non-significatif* dans le cas contraire.

Chaque pixel est classifié en :

- *Zero-Tree Root* (ZTR) : si le pixel est *non-significatif* par rapport à T, **tous** ses descendants (y inclus les descendants de ses descendants) sont *non-significatifs*, et si son parent est significatif. Tous les descendants d'un ZTR seront marqués dans un tableau auxiliaire et ne seront pas balayés.
- Zéro-Isolé (ZI) : si le pixel est *non-significatif* par rapport à T mais il a au moins un descendant significatif.
- Positif (P) : si le pixel est significatif et positif.
- Négatif (N) : si le pixel est significatif et negatif.

Une fois le balayage terminé, les valeurs des coefficients trouvé significatifs sont actualisées selon la règle suivante :

$$xt(i,j) = \begin{cases} xt(i,j) - \dfrac{\mathrm{T}}{2}, & \text{si } xt(i,j) > 0 \\ xt(i,j) + \dfrac{\mathrm{T}}{2}, & \text{sinon} \end{cases},$$

la valeur du seuil est réduite à la moitié (T = T/2), et le balayage des coefficients réitéré.

Les quatre symboles (ZTR, ZI, P et N) ainsi déterminés sont codés par un code à longueurs fixes de 2 bits.
Le codage peut être arrêté à tout instant. Un critère utile d'arrêt est la taille du fichier binaire associé. Cela permet d'avoir le contrôle du taux de compression.

L'algorithme de reconstruction de la transformée en ondelettes est fondé sur le même balayage des coefficients. L'image transformée est tout d'abord initialisée à zéro.
Le décodeur reçoit successivement les quatre symboles et actualise la transformée reconstruite chaque fois qu'un symbole P ou N est détecté :

$$xt(i,j) = \begin{cases} xt(i,j) + T, & \text{si N} \\ xt(i,j) - T, & \text{si P} \end{cases}$$

Une fois le balayage terminé pour le seuil T considéré, le seuil est actualisé (T = T/2), et le balayage des coefficients recommence.

Prototypes :

```
int ztw(double **xt, int Width, int Height, int niv-resol, int size, const char *bitstream_name);
int ztw(double **xtrec, int Width, int Height, int niv-resol, const char *bitstream_name);
        // xt - image transformée
        // xtrec - l'image transformée reconstruite à partir du flux binaire
        // Width, Height : taille de l'image
        // niv_resol le nombre de niveaux de résolution
        // bitstream_name - le nom du fichier de stockage du flux binaire
        // size - la taille imposée du flux binaire (en kbits)
```

7. Implantez en C++ une **procédure de calcul du quantificateur optimal Max-Lloyd :**

Prototypes:
double * calcul_histo(double *x, double &max, double &min);
// procédure de calcul de l'histogramme, prend en entrée une image et retourne un vecteur
// (double*) représentant l'histogramme de l'image et les valeurs minimale et maximale de
// l'image (*min*, resp. *max*)
int Max_Lloyd(double **x; double **xq; int Width, int Height, double *histo, double *seuils, double *rec, int NL);
// x -l'image initiale
// xq - l'image quantifiée
// Width, Height : taille de l'image
// histo - histogramme de l'image
// seuils, rec - les seuils et les niveaux de reconstruction du quantificateur
// NL - le nombre de niveaux de quantification

La solution (les valeurs des seuils *s* et des niveaux de reconstruction *r*) est obtenue en itérant successivement les équations suivantes :

$$t_k = \frac{r_k + r_{k+1}}{2} , \qquad (1)$$

$$r_k = \frac{\sum_{l=t_k}^{t_{k+1}} l\, p(l)}{\sum_{l=t_k}^{t_{k+1}} p(l)} , \qquad (2)$$

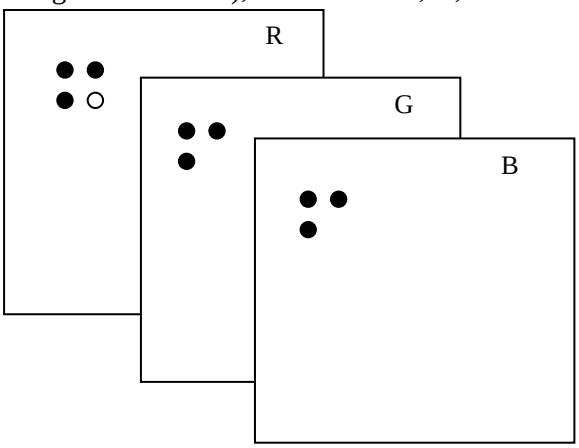
où $p(l)$ est la densité de probabilité de l'image (approchée par l'histogramme de l'image).

L'initialisation correspond aux seuils et niveaux de reconstruction d'un quantificateur uniforme. Les seuils t_0 et t_{NL+1} seront figés aux valeurs minimale et resp. maximale de l'image. Le critère d'arrêt de l'algorithme sera la convergence (pas de changements des seuils et niveaux de reconstruction entre deux itérations successives).

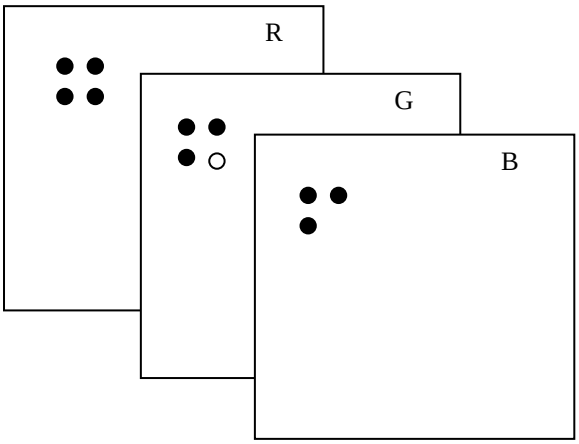
Appliquer le quantificateur Lloyd pour quantifier les erreurs de prédictions issues d'une prédiction DPCM ligne par ligne (utiliser le programme fourni lors du TP). Comparer les images reconstruites, en termes de qualité visuelle et erreur quadratique moyenne, avec celles correspondant à une quantification uniforme des erreurs de prédiction.

8. Implantez une procédure de **codage prédictif des images couleur**, avec boucle de rétroaction.

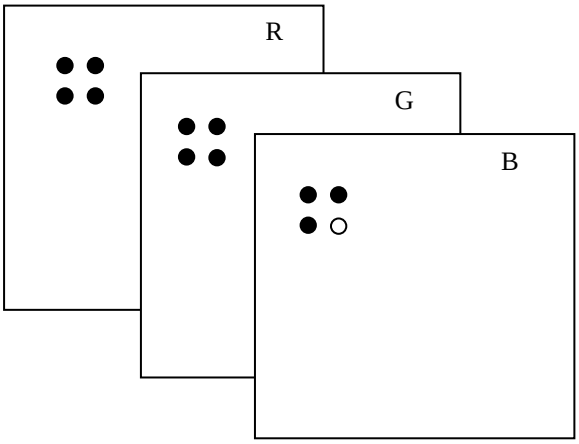
Les fénêtres de prédiction sont illustrées Figure 1. Elles correspondent à un balayage *raster-scan* de l'image (de haut en bas et de gauche à droite), dans l'ordre R, G, B.



1.a. Prédiction d'un pixel du plan de couleur R.



1.a. Prédiction d'un pixel du plan de couleur G.



1.a. Prédiction d'un pixel du plan de couleur B.

Figure 1. Les fenêtres causales de prédiction.

Les prédicteurs sont définis par les équations suivantes :

$$\begin{aligned}\hat{R}(i,j) &= r_1 R(i-1,j) + r_2 R(i,j-1) + r_3 R(i-1,j-1) \\ &+ r_4 G(i-1,j) + r_5 G(i,j-1) + r_6 G(i-1,j-1) \\ &+ r_7 B(i-1,j) + r_8 B(i,j-1) + r_9 B(i-1,j-1)\end{aligned} \quad (1)$$

$$\begin{aligned}\hat{G}(i,j) &= g_1 R(i-1,j) + g_2 R(i,j-1) + g_3 R(i-1,j-1) \\ &+ g_4 G(i-1,j) + g_5 G(i,j-1) + g_6 G(i-1,j-1) \\ &+ g_7 B(i-1,j) + g_8 B(i,j-1) + g_9 B(i-1,j-1) \\ &+ g_{10} R(i,j)\end{aligned} \quad (2)$$

$$\begin{aligned}\hat{B}(i,j) &= b_1 R(i-1,j) + b_2 R(i,j-1) + b_3 R(i-1,j-1) \\ &+ b_4 G(i-1,j) + b_5 G(i,j-1) + b_6 G(i-1,j-1) \\ &+ b_7 B(i-1,j) + b_8 B(i,j-1) + b_9 B(i-1,j-1) \\ &+ b_{10} R(i,j) + b_{11} G(i,j)\end{aligned} \quad (3)$$

Trouver les coefficients de prédiction optimaux (modélisation AR) et implantez les procédures de codage/décodage prédictif d'images couleur.
Deux stratégies de prédiction seront considérées :

* *stratégie globale*: un modèle de prédiction unique pour toute l'image,
* *stratégie locale* : partition de l'image en blocs rectangulaires et optimisation des prédicteurs pour chaque bloc (taille des blocs recommandée : 32x32 pixels).

Comparez les deux approches en terme de réduction d'entropie.
Etudier l'effet, en terme de réduction d'entropie, de l'agrandissement de la taille de la fenêtre de prédiction.

Comparez les résultats obtenus en terme de réduction d'entropie par rapport avec une procédure de prédiction bande-à-bande, faisant intervenir les mêmes pixels que ceux illustrés Figure 1, mais uniquement dans le plan de couleur du pixel courant.

Prototypes :

int predictionRGB(double **r, double **g, double **b, double **err_r, double **err_g, double **err_b, int Width, int Height, double delta, double &R_{med}, double &G_{med}, double &B_{med});
int predictionRGB_inv(double **err_r, double **err_rg, double **err_b, double **r_rec, double **g_rec, double **b_rec int Width, int Height, double R_{med}, double G_{med}, double B_{med});

// r, g, b -les trois plans de couleur de l'image initiale
//err_r, err_g, err_b - les erreurs de prédiction sur chaque plan
// delta - le pas de quantification (uniforme) des erreurs de prédiction
// r_rec, g_rec, b_rec -les trois pans de couleur de l'image reconstruite
// Width, Height : taille de l'image
// R_{med}, G_{med} et B_{med} représentent les valeurs moyennes de l'image sur chaque plan de couleur
// R, G, B. Elles sont calculées pendant la phase de prédiction directe (au codeur) et transmis
// comme arguments au décodeur. La prédiction est réalisé sur des signaux Rp, Gp, Bp de
// moyenne nulle, obtenus en soustrayant les moyennes R_{med}, G_{med} et B_{med} des signaux

// originaux R, G, B. Au décodeur, on ajoute simplement aux valeurs recosntituées les
// moyennes.

Indications :
1. pour déterminer les coefficients de prédiction, on impose la condition de modèle AR. Cela revient à multiplier chacune des équations (1-3) par chacun des échantillons intervenant dans la combinaison linéaire définissant le prédicteur et prendre la moyenne statistique. On obtient alors trois systèmes d'équations linéaires (un pour chaque prédicteur), dont la solution fournit les coefficients de prédiction r_i, g_j, b_k..