Travaux Pratiques 4 - OASIS SI101

Compression des signaux naturels

---

1 Apprendre une base adaptée à l'analyse d'une base de donnée

Dans cette partie nous voyons comment la méthode vue en cours pour apprendre une base orthonormée adaptée à certains signaux peut servir diverses tâches. Nous illustrons ici la compression et le débruitage. Nous utilisons une base de données de visages qui sont donnés sous la forme de 2414 vecteurs de taille 1024 (représentant des images de taille 32 sur 32).

1. Suivez la procédure habituelle pour charger les données du TP4. [1]
2. On commence par les sections "ANALYSE DE VISAGES" que vous exécutez cellule par cellule.
3. Les personnes sont-elles reconnaissables, d'après vous, si on garde 100 composantes par visage ?
4. Les personnes sont-elles reconnaissables après ajout de bruit ?
5. Les personnes sont-elles reconnaissables après ajout de bruit puis compression sur 100 composantes ? (on verra une explication de ce phénomène plus loin)

2 Préliminaires : programmation de la DCT et DCT2D

Comme vu en cours la DCT, (transformée en cosinus discret) est plus adaptée à l'étude des signaux naturels. On rappelle la définition de la DCT d'un signal u de taille N

$$\forall 0 \leqslant k \leqslant N-1,\ \hat{u}_k^D = p_k \sum_{n=0}^{N-1} u_n \cos\left(2\pi(n+\frac{1}{2})\frac{k}{2N}\right)$$

avec $p_0 = \sqrt{\frac{1}{N}}$ et $p_k = \sqrt{\frac{2}{N}}$ pour $k \neq 0$.

Pour calculer la DCT par cette formule il faut effectuer N^2 opérations. Cependant, on peut relier la DCT du signal u à la TFD du signal x de longueur $2N$ (inicé de 0 à $2N-1$) défini comme suit

$$x_n = \begin{cases} u_n \text{ si } 0 \leqslant n < N \\ u_{2N-1-n} \text{ si } N \leqslant n \leqslant 2N-1 \end{cases}$$

par

$$\hat{u}_0^D = \frac{1}{2\sqrt{N}}\hat{x}_0$$

et

$$\hat{u}_k^D = e^{-i\pi\frac{k}{2N}}\frac{1}{\sqrt{2N}}\hat{x}_k \text{ pour } 1 \leqslant k \leqslant N-1 \tag{1}$$

L'avantage est que, si N est une puissance de 2, on peut effectuer la TFD de x par l'algorithme de la FFT en temps $2N\log(2N)$ et ajouter N opérations simples (multiplication) pour calculer la DCT.

1. compléter la fonction `madct`.
2. Justifier le fonctionnement de `madct2` qui effectue la DCT bi-dimensionnelle.
3. Les fonctions `imadct` et `imadct2` sont fournis. Ils effectuent les DCT inverses.
4. Tracer le module de la DCT du signal np.arange(1024) ainsi que le module de sa TFD. Quels commentaires pouvez-vous faire ? (décroissance, symétrie)

1. Pour ce TP il est essentiel de configurer l'affichage de spyder pour que celui-ci se fasse dans des fenêtres externes : Outils-> Console python-> Affichage -> QT5 et relancer le noyau python pour que cela soit pris en compte (roue crantée en haut à droite de la fenêtre interactive)

3 Compression linéaire

Complétz la fonction `compression_lineaire` qui prend en entrée une image et deux entiers N et m et qui renvoie l'image compressée linéairement sur la base de la DCT2D NxN [2] en gardant pour chaque bloc NxN les m coefficients de plus basse fréquence [3].

On donne les étapes à réalisées par cette fonction

(a) Créer un masque de taille NxN qui vaut 1 si la fréquence qui lui correspond dans l'ordre de la DCT2D fait partie des m plus faibles.

(b) Parcourir tous les blocs NxN de l'image (sans chevauchement) et leur appliquer la DCT2D et multiplier par le masque et faire la DCT2D inverse.

3.1 Evaluation

On rappelle que pour charger une image il faut faire `a=charge_image('fichier.png')` et que pour voir une image il faut faire `affiche(a)`.

On chargera l'image "lena.png" dans une variable appelée lena.

1. Comparer le résultat d'une compression linéaire avec la DCT512 et la DCT 16 et 5120 coefficients gardés.
2. Laquelle des deux images est la plus proche en norme de l'originale.
3. Charger l'image "lena.jpg" dans une variable lenajpg. Combien vous faut-il garder de coefficients en compression linéaire DCT 512 pour obtenir un aussi bon résultat en termes de norme ? (on fera une dichotomie sur le nombre de coefficients gardés.) Même question en DCT 8x8 ?

4 Compression adaptative

On rappelle que la compression adaptative sur une base consiste à ne garder que les coefficients de plus fort module. Compléter la fonction `compression_adaptative` qui prend en argument une image et deux entiers N et m et qui renvoie une image dans laquelle n'ont été gardés que les m coefficients les plus forts de la DCT NxN

On donne les étapes de votre fonction :

(a) Parcourir tous les blocs NxN de l'image (sans chevauchement) et leur appliquer la DCT2D.

(b) Dans un tableau (colonne) temporaire recopier les modules de tous les coefficients de la transformée.

(c) Trier le tableau (fonction `np.sort`).

(d) Choisir le seuil en-dessous duquel un coefficient devra être mis à zéro.

(e) Mettre à zéro les coefficients trop petits.

(f) Parcourir tous les blocs de la version transformée (où ont été annulés les faibles coefficients) et leur appliquer une DCT2D inverse.

1. Comparer les résultats de la compression adaptative en base 512, 16 et 8 avec ceux de la compression linéaire pour un même nombre de coefficients gardés.
   À partir de maintenant on n'utilise plus que la compression adaptative.
2. Combien vous faut-il garder de coefficients DCT 8x8 (à 10 près) pour atteindre la qualité de la compression jpg donnée en exemple (lenajpg) ?

2. Vous ne vous préoccupez pas des problèmes de bord et utiliserez toujours des images dont la taille est multiple de N. Dans la pratique N sera toujours 8,16,32,...,512

3. Cela signifie que l'on a gardé, au total, $m(M/N)^2$ si l'image est de taille $M \times M$. Par exemple, si l'image est de taille 512×512 et que l'on garde 5 coefficients dans chaque bloc de DCT 16x16, on a gardé au total : $(512/16)^2 \times 5 = 5120$ coefficients. C'est ce nombre total qui compte dans l'efficacité de la compression

3. Une fonction appelée `compression_adaptative_FFT` effectue une compression en utilisant la TFD2D au lieu de la DCT2D. Effectuez une telle compression 16x16. Que constatez-vous visuellement (toujours avec un nombre de coefficients identique) ? (après cette expérience on n'utilise plus jamais la TFD2D)

5 Le bruit est-il compressible ? (explication du phénomène vu en 1.5)

On crée une image de bruit blanc par la commande `im=randn(512,512)`. Tracer, en fonction de m la quantité d'énergie capturée dans les m plus gros coefficients de la DCT 8x8 d'une telle image. Comparer au même graphique obtenu pour une image naturelle. Vous verrez en TD qu'aucune base (DCT ou autre) ne permet de concentrer l'énergie d'un bruit blanc.

5.1 La compression comme débruitage

On commence par créer une version bruitée de lena par la commande `lenabruit=lena+20*randn(512,512)`.

Pour des des valeurs de m allant de 2000 à 10000 par pas de 500, tracer la norme de la différence entre la compression de "lenabruit" en DCT 16 avec m coefficients et "lena". Comment expliquez-vous que le fait de compresser une image bruitée permette de se rapprocher de l'image originale ? Pourquoi la distance augmente-t-elle quand m devient grand ?

6 Reverse Engineering du format JPEG

On rappelle que le format JPEG (le plus utilisé pour la compression des images) est basé sur une décomposition en DCT 8x8, puis une annulation des coefficients trop faibles. Il y a d'autres composantes telles que le codage prédictif du coefficient de fréquence 0 à partir du coefficient correspondant dans le bloc précédent. Les coefficients non annulés sont ensuite quantifiés et codés (codage entropique, voir CNTI).

En un sens, votre programme fait déjà une bonne partie de la compression JPEG.

1. Construire une version transformée de lenajpg en DCT 8x8.
2. Combien de coefficients sont nuls ?
3. Combien faut-il de coefficients à votre programme pour atteindre la même qualité (norme)(idem question 3.2) ? Et en DCT 16x16 ?
4. Construire une version transformée en DCT 8x8 de l'image originale.
5. Tracer le module de la transformée de lenajpg en fonction de celle de lena (`plt.plot(abs(tr.reshape(-1)),abs(trjpg.reshape(-1)),'+')`) (si tr et trjpg sont les transformées). Cela vous renseigne-t-il sur la manière dont jpg quantifie les coefficients ?
6. Pour les canaux de basse fréquence, tracer ce même graphique. Quelles sont les précision de quantifications pour les fréquence (0,0)...(2,2) (9 canaux au total).
7. Combien de bits sont nécéssaires pour coder chacun de ces canaux ? (il suffit de compter le nombre total de valeurs possibles dans chaque canal)