La transformation en cosinus discrète

Jean-Paul Quélen

8 mars 2025

C'est une transformation appelée, en anglais, Discrete Cosine Transform (DCT) utilisée dans le codage JPEG des images.

Dans le codage JPEG, l'image est d'abord découpée en blocs de 8×8 pixels. La couleur étant séparée de la luminance, on ne s'intéressera ici qu'à la valeur en niveaux de gris pris par chaque pixel du bloc de 64 pixels. On supposera aussi que cette valeur est un entier entre 0 (noir) et 255 (blanc). Cette valeur tient donc sur un octet et le bloc est déterminé par 64 octets correspondants aux 64 niveaux de gris.

Chaque bloc peut être caractérisé par une fonction F de $\{0,1,...7\} \times \{0,1,...7\}$ dans $\mathbf{R}$. À chaque pixel de coordonnées (x,y) on associe un niveau de gris $F(x,y)$.

L'ensemble des fonctions F forme un espace vectoriel de dimension 64 dans lequel on définit le produit scalaire :

$$F \cdot G = \sum_{0 \leqslant x,y \leqslant 7} F(x,y)G(x,y)$$

On démontre que la famille des 64 fonctions $F_{u,v}$, $0 \leqslant u,v \leqslant 7$, définies par :

$$F_{u,v}(x,y) = N(u)N(v)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}$$

avec $N(0) = \dfrac{1}{2\sqrt{2}}$ et $N(u) = \dfrac{1}{2}$ si $1 \leqslant u \leqslant 7$, forme une base orthonormée et donc que, pour toute fonction F, il existe une famille unique de 64 coefficients $c_{u,y}$, $0 \leqslant u,v \leqslant 7$, telle que :

$$F(x,y) = \sum_{0 \leqslant u,v \leqslant 7} c_{u,v}F_{u,v}(x,y)$$

La transformée en cosinus discrète consiste alors à remplacer les 64 niveaux de gris (fonction F) par les 64 coefficients $c_{u,v}$.

Ces coefficients sont des nombres réels qu'on peut simplifier ou même mettre à 0 pour les plus petits. Cela modifie légèrement l'aspect du bloc de 8×8 pixels mais cela permet surtout de caractériser ce bloc avec moins de données. C'est ce qu'on appelle une compression « avec perte ».

Calcul des $F_{u,v}$ et $c_{u,v}$

On calcule d'abord $f_u(x) = N(u)\cos\dfrac{(2x+1)u\pi}{16}$ puis $F_{u,v}(x,y) = f_u(x)f_v(y)$. Les 64 images de base correspondantes aux fonctions $F_{u,v}$ peuvent alors être affichées.

Si F est la fonction correspondant à l'image source alors $c_{u,v}$ se calcule grâce à la formule suivante :

$$c_{u,v} = F \cdot F_{u,v} = \sum_{0 \leqslant x,y \leqslant 7} F(x,y) F_{u,v}(x,y)$$

On démontre alors que :

$$\sum_{0 \leqslant u,v \leqslant 7} c_{u,v} F_{u,v}(x,y) = F(x,y)$$

Pour que les calculs soient compatibles avec les affichages alors on remplacera le niveau de gris $F(x,y)$ (entier entre 0 et 255) par un nombre entre -128 et $+127$ en soustrayant tout simplement 128. De même on ajoutera aux niveaux de gris recalculés à partir des $c_{u,v}$, la valeur 128 et on fera en sorte que le résultat $F(x,y)$ donne un entier entre 0 et 255.

Conclusion

La DCT ne constitue pas en tant que telle une compression avec « avec perte »puisque les coefficients $c_{u,v}$ permettent de récupérer les niveaux de gris de l'image source.

L'idée consiste alors de diviser et d'arrondir les coefficients $c_{u,v}$. On obtient alors des nombres plus petits, voir 0 si le coefficient était déjà petit au départ. La série des 64 nombres obtenus, dont beaucoup peuvent être nuls, peut être codée avec peu de mémoire si on utilise des méthodes de compressions numériques appropriées.

Il faut bien sûr ajouter la table des multiplicateurs de chaque coefficient (table de quantification) car le diviseur peut être différent pour chaque coefficient mais cette table est la même pour tous les blocs de 8×8 pixels donc l'image, surtout si elle est grande, ne prendra pas beaucoup plus de place sur le disque dur.

Utilisation de l'application

Une image de 8×8 pixels est proposée par défaut. On peut sélectionner aucune, toutes ou un certain nombre d'images de base et observer l'aspect de l'image recomposée. On peut aussi ne garder que les images de base correspondant à des coefficients $c_{u,v}$ significatifs.

Dans un second temps, on peut préciser le coefficient multiplicateur. 1 pour un simple arrondi. Essayer une autre valeur, 2 ou 16 par exemple, et observer la modification de l'image reconstituée.

On peut aussi remplacer l'image source en entrant une table de 64 données, entiers compris entre 0 et 255 écrits en décimal ou hexadécimal. On pourra remarquer que, pour une image avec beaucoup moins de détails, les coefficients deviennent vite très faibles.

On peut aussi rentrer une table de multiplicateurs (8×8 valeurs de quantification). Cela détériorera de façon importante l'image si les multiplicateurs sont importants.

La seule chose que l'on ne peut prévoir est la taille définitive du fichier correspondant à l'image. Si le bloc de 8×8 pixels est presque uniforme (morceau de ciel bleu par exemple) la compression sera forte. Si le bloc contient beaucoup de détails, comme l'image donnée par défaut, la compression sera plus ou moins forte suivant les valeurs de quantification et pourra affecter la qualité de l'image.