Compression d'image par série de Fourier

Xavier JUVIGNY

6 Janvier 2021

Contents

1 Présentation

De nombreux algorithmes ont été développés pour la compression et la décompression d'image. Parmi ces algorithmes, on peut trouver la décomposition de l'image en série de Fourier.

Supposons l'image uniquement en niveau de gris. Le principe est alors simple :

- On décompose une image, contenant $n_x \times n_y$ pixels p_{ij}, en $n_x \times n_y$ coefficients complexes décrivant une transformation en série de Fourier dans la direction O_x et dans la direction O_y. Un coefficient c_{jk} de la double série de Fourier sera calculé comme suit :

$$c_{jk} = \sum_{l=1}^{n_x} \sum_{m=1}^{n_y} p_{lm} e^{-2i\pi \frac{j.l}{n_x}} e^{-2i\pi \frac{k.m}{n_y}}$$

- On sélectionne les $p\%$ coefficients c_{jk} possédant le plus gros module (ceux qui apportent le plus d'information pour reconstituer l'image). Ces coefficients sont stockés dans une structure "creuse" de type Sparse Row Compression, c'est à dire qu'on se munit d'une structure possédant :
  - Un tableau `coefficients` possédant les coefficients complexes sélectionnés c_{jk} rangés par ordre croissant des j;
  - Un tableau `ind_cols` possédant les indices k correspondant à ces coefficients complexes (pas obligatoirement rangés par ordre croissant);
  - Un tableau `begin_rows` possédant des indices indiquant le début des coefficients ayant un indice j pour premier indice.

  Par exemple, considérons que nous voulons stocker les coefficients $c_{00}, c_{01}, c_{11}, c_{13}, c_{22}, c_{23}, c_{24}$ et c_{34}. Alors les trois tableaux vaudront :
  - `coefficients` :

c_{00}	c_{01}	c_{11}	c_{13}	c_{22}	c_{23}	c_{24}	c_{34}

  - `ind_cols` :

0	1	1	3	2	3	4	4

  - `beg_rows` :

0	2	4	7	8

  La dernière valeur dans `beg_rows` est égale au nombre de coefficients stockés.

Pour la reconstitution de l'image, il suffit de faire la transformation de Fourier inverse, c'est à dire, pour calculer l'intensité d'un pixel p_{ij}, d'appliquer la formule :

$$p_{jk} = \sum_{l=1}^{n_x} \sum_{m=1}^{n_y} c_{lm} e^{+2i\pi \frac{j.l}{n_x}} e^{+2i\pi \frac{k.m}{n_y}}$$

en ne parcourant que les coefficients sélectionnés durant la compression.

Pour une image en couleur, le principe est le même mais appliqué à chaque canal de couleur.

Remarque : Pour avoir une version plus performante de ce code, il aurait fallu effectuer une FFT (Fast Fourier Transform) (en $N^3 \log(N)$) plutôt qu'une transformée de Fourier discrète (DFT) (en N^4) comme écrite dans le code fourni. Cependant, la parallélisation et la compréhension du code aurait été plus compliquée ! De même, pour éviter une complexité de compression trop grande, on préfère en général compresser une image par bloc afin de réduire le coût de calcul (c'est ce que fait le format jpeg). Mais là encore, le code aurait été beaucoup plus compliqué !

2 Travail demandé

Le programme est conçu pour accepter deux paramètres optionnels lors de son appel :

```
./fourier_compression.exe [image] [taux]
```

où `[image]` est le nom de l'image à traiter (avec son path relatif ou absolu, la valeur par défaut étant `data/small_lena_gray.png`) et `[taux]` qui est le pourcentage de compression voulu c'est à dire le pourcentage par rapport au nombre total de pixels de coefficients qu'on veut conserver (par défaut 10%).

Pour chaque étape de la parallélisation, conserver une version appropriée !

- Mesurez les temps pris par :
  - L'encodage de l'image par DFT
  - La sélection des $p\%$ plus gros coefficients
  - La reconstitution de l'image "compressée";

  pour `tiny_lena_gray.png` et `small_lena_gray.png` (se trouvant dans le répertoire data)
- Parallélisez à l'aide d'OpenMP, si cela est possible, les fonctions prenant le plus de temps. Précisez sur quel type de valeur (pixel, fréquence, etc.) la parallélisation est faite. Mesurez les temps obtenus et calculez et interprétez l'accélération ainsi obtenue.
- Première parallélisation avec MPI : dans un premier temps, faites une partition de l'image, c'est à dire que chaque processus va prendre n_y/nbp lignes et parallélisez la transformation en Fourier de l'image. Faites en sorte que seul le processus 0 fera une reconstitution et une sauvegarde de l'image. De même, mesurez les temps obtenus, calculez et interprétez l'accélération obtenue.
- Seconde parallélisation avec MPI : en repartant de la version séquentielle, on va cette fois partitionner l'espace des fréquences dans la direction O_y, chaque processus ne s'occupant que de n_y/nbp fréquences. Pour la sélection des $p\%$ plus gros coefficients, chaque processus sélectionnera les $N = p\% \times n_x \times n_y$ plus gros coefficients puis sélectionnera parmi ces $nbp \times N$ coefficients les N plus gros coefficients. Mesurez encore les temps de chaque fonction ainsi que l'accélération et interprétez les résultats.