FICHE SYNOPTIQUE 2014

BAHL Gaétan, 7823

Sujet : **Théorie des ondelettes : application à la compression d'images**

Ce TIPE a été réalisé en binôme. Mon camarade s'est chargé de la partie mathématique (théorie et preuves). Je me suis occupé de la partie informatique, qui met en pratique la transformation en ondelettes discrètes en l'appliquant à la compression d'images.

Les langages utilisés sont Python et OpenCL.

Algorithme matriciel de Stéphane Mallat

Nous avons tout d'abord écrit des fonctions de multiplication matricielle en Python, puis mis en place cet algorithme, qui se base sur les ondelettes de Haar (carrées), qui conviennent très bien au traitement de pixels.

L'algorithme sépare l'image en ses trois composantes RGB, puis multiplie les 3 matrices crées avec la matrice représentant la transformation en ondelettes.

Il s'agit ensuite de supprimer les coefficients d'ondelettes négligeables pour compresser l'image. L'algorithme s'occupe de cela grâce à un seuil prédéfini, puis effectue la transformation inverse, pour retrouver l'image de départ, avec certaines zones compressées (les aplats de couleurs sont unifiés, par exemple).

Interface graphique en Tkinter

Nous avons utilisé la librairie d'interfaces graphiques Tkinter incluse dans Python pour créer une petite application permettant : l'affichage d'une image, la compression grâce à l'algorithme présenté ci−dessus, le choix du seuil de compression, la conversion de l'image en niveaux de gris et la sauvegarde de l'image.

Optimisation de l'algorithme de Mallat : fasthaar

Puisque l'algorithme présenté plus haut est matriciel, sa complexité est en O(n^3). Or nous avons remarqué que les matrices de la transformation contiennent beaucoup de zéros, et donc des multiplications inutiles. Nous avons alors créé un nouvel algorithme, qui effectue une transformation similaire, en découpant l'image en carrés de 4 pixels, et qui effectue la compression par ondelettes sur ces carrés. La transformation ne peut plus se faire récursivement, mais sa complexité est alors O(n^2).

Nous avons ensuite codé un programme permettant de chronométrer le nouvel algorithme, en créant des images aléatoires de tailles allant de 1 à 1000 pixels de côté. Nous avons fait tourner cet algorithme pendant 28h, pour un temps total de compression d'environ 5h (le reste du temps a été utilisé pour générer les images aléatoires), et avec un temps de compression médian de 9 secondes. Des graphes ont été dessinés et montrent bien l'évolution en n^2.

Transferts et stockages d'images par le réseau

Nous avons créé un logiciel d'archivage d'images sur serveur distant.

Archivage : Le PC client ouvre l'image et l'envoie au serveur par le réseau. Le serveur effectue la transformation et garde uniquement les coefficients d'ondelettes. Le client garde une image dont les dimensions sont réduites de moitié.

Restauration : Le PC client demande les coefficients d'ondelette au serveur. Le serveur lui envoie, et le client effectue la transformation inverse, et peut ainsi afficher l'image en pleine résolution.

Accélération matérielle de la compression avec OpenCL

OpenCL permet d'effectuer des calculs sur le processeur graphique des ordinateurs (ou GPU). Ces derniers contiennent des milliers de petits processeurs de flux effectuant tous la même tâche. Ce sont des processeurs fortement parallélisés.

Nous avons donc adapté l'algorithme fasthaar pour que chaque coeur graphique puisse exécuter une partie de la compression. Le programme Python charge une petite fonction OpenCL exécutée sur le GPU d'effectuer la transformation.

On découpe à nouveau l'image en blocs de 4 pixels, puis chaque bloc est envoyé à un processeur différent (ceci est géré par OpenCL).

Le programme a été testé avec une carte graphique AMD Radeon HD 7870 sur des images prises par différents appareils photos.
Les performances sont bien meilleures qu'avec un processeur classique :

−0.057s pour une image de 3 Mégapixels
−0.079s pour 5 Mégapixels
−0.122s pour 12 Mégapixels