Parallélisation : Lisseurs pour le Débruitage d'Images

Projet du module CHPS 0801

Ce projet consiste à implanter en mémoire partagée des algorithmes de lissage pour des images.

1 Fondements mathématiques

1.1 Algèbre linéaire

De nombreuses applications nécessitent la résolution de systèmes linéaires d'équations de la forme $Ax = b$ où A est une matrice (le plus souvent carrée), b le vecteur second membre et x le vecteur inconnu.

De nombreux algorithmes ont été développés pour résoudre de tels problèmes linéaires, que l'on peut classer en deux grandes catégories :

— les méthodes directes, qui calculent le résultat x, sans étapes intermédiaires ;

— les méthodes itératives, qui calculent x via une suite de valeurs intermédiaires $x^{(i)}$.

Parmi les méthodes itératives, nous allons nous intéresser plus particulièrement à deux d'entre elles :

— la méthode de Jacobi ;

— la méthode de Gauss-Seidel.

1.1.1 Fonctionnement

Les méthodes de Jacobi et de Gauss-Seidel sont des méthodes de type point fixe où l'on va construire un itéré de la forme : $x^{(k+1)} = F(x^{(k)})$.

Pour cela, on va décomposer la matrice A en $A = M - N$ où M est une matrice inversible et N une matrice.

$$Ax = b \iff Mx = Nx + b \iff x = M^{-1}Nx + M^{-1}b = F(x).$$

L'algorithme de base de ces méthodes se résume donc à la définition de la suite :

$$\begin{cases} x^{(0)} \\ x^{(k+1)} \leftarrow F(x^{(k)}) \end{cases}$$

1.1.2 Méthode de Jacobi

La méthode de Jacobi repose sur cette décomposition de la matrice $A = D + L + U$, où D est la matrice diagonale de A, L la matrice triangulaire inférieure de A (sans la diagonale), et U la matrice triangulaire supérieure de A (sans la diagonale). On pose ensuite $M = D$ et $N = -L - U$ pour appliquer la méthode décrite précédemment.

En développant les calculs, on obtient :

$$x_i^{(k+1)} = -\frac{1}{a_{ii}} \sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(k)} + \frac{b_i}{a_{ii}}$$

.

1.1.3 Méthode de Gauss-Seidel

La méthode de Gauss-Seidel repose sur la même décomposition $A = D - E - F$ que la méthode de Jacobi mais définit les matrices $M = D + L$ et $N = U$.

Cela signifie qu'une itération de l'algorithme de Gauss-Seidel correspond à résoudre le système triangulaire inférieur suivant :

$$(D + L)x^{(k+1)} = b - Ux^{(k)}$$

.

Une telle résolution se fait par une descente, en calculant de proche en proche $x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)}$.

En développant les calculs, on arrive à la formule suivante :

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right)$$

.

1.2 Algorithmes numériques

De nombreux schémas numériques utilisent des algorithmes très similaires aux algorithmes itératifs de résolution de systèmes linéaires présentés précédemment.

Beaucoup de schémas numériques explicites [1] reposent sur la notion de *stencil*.

Dans ces schémas, les inconnues à rechercher sont définies sous forme de tableaux et des itérations font évoluer les valeurs de ces tableaux selon des relations définies localement.

Le *stencil* repose sur une notion de voisinage, sur un maillage, ce dernier correspondant à la discrétisation en espace de l'espace physique.

1. Qui ne reposent pas sur la résolution d'un système linéaire mais qui au contraire détaillent toutes les étapes de la résolution.

Pour les grilles cartésiennes[2], les calculs sont généralement définis par rapport aux valeurs des mailles voisines. L'article wikipedia [`https://en.wikipedia.org/wiki/Iterative_Stencil_Loops`] présente bien les principes généraux.

1.2.1 Itérations à la Jacobi

Dans ces stencils, la nouvelle valeur $x_i^{(k+1)}$ à la maille i est calculée en fonction des valeurs $x_j^{(k)}$ des mailles j dans le voisinage de i.

En 2D, on repose souvent sur un voisinage à 5 points :

— la maille i ;

— les mailles partageant une des quatre faces de i : Nord, Sud, Est, Ouest (ou Haut, Bas, Droite, Gauche).

1.2.2 Itérations amont-aval

Il est également possible de définir des itérations à la Gauss-Seidel en utilisant des informations $x_j^{(k+1)}$ pour les mailles amont, et $x_j^{(k)}$ pour les mailles aval.

1.3 Autres Applications

Ce type de calcul a des propriétés de *lissage*, les calculs élémentaires consistant à effectuer des moyennes pondérées avec les voisins. Pour cela, ils sont notamment utilisés en imagerie pour débruiter les images.

Pour le projet nous nous placerons dans une telle application et le but sera d'implanter des lisseurs efficaces avec une parallélisation en mémoire partagée.

2. Ou les maillages semi-structurés.

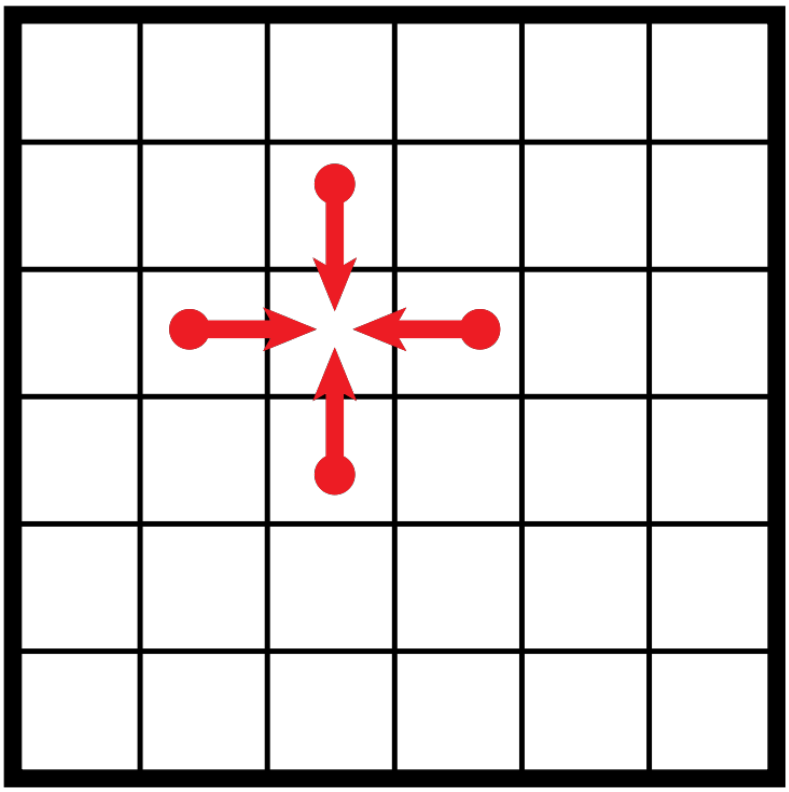

FIGURE 1 – Un stencil à 5 points sur une grille 2D.

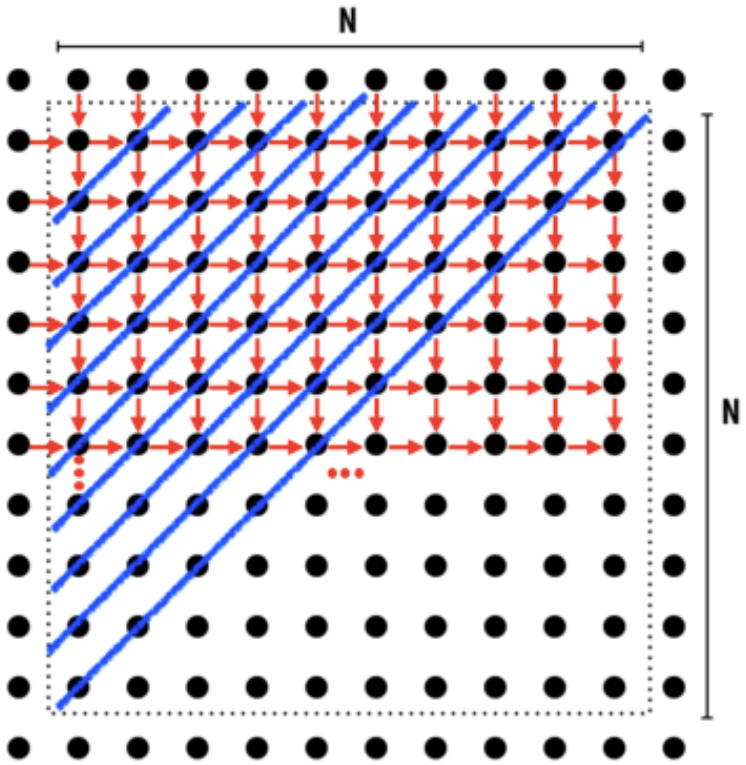

FIGURE 2 – Dépendances amont–aval.

2 Réalisations

L'objectif du projet est d'implanter un programme de lissage parallèle (en mémoire partagée) pour le débruitage d'images. Un exemple de code de lecture/écriture d'images via *opencv* vous sera fourni.

2.1 Méthode Jacobi

Pour le débruitage d'image, on peut exploiter la méthode de Jacobi avec un *stencil* à 5 points en faisant la moyenne des valeurs.

$$x_{i,j}^{(k+1)} \leftarrow \frac{x_{i,j}^{(k)} + x_{i-1,j}^{(k)} + x_{i+1,j}^{(k)} + x_{i,j-1}^{(k)} + x_{i,j+1}^{(k)}}{5}$$

Un des buts du projet sera d'écrire la méthode et la paralléliser, sur CPU et GPU.

2.2 Méthode Gauss-Seidel

On va utiliser la même technique de moyenne que celle de Jacobi mais procéder par avancé de fronts, comme illustré précédemment.

$$x_{i,j}^{(k+1)} \leftarrow \frac{\left(x_{i-1,j}^{(k+1)} + x_{i,j-1}^{(k+1)}\right) + \left(x_{i+1,j}^{(k)} + x_{i,j+1}^{(k)} + x_{i,j}^{(k)}\right)}{5}$$

On remarque une dépendance à des données de l'instant k et de l'instant $k+1$.

Un des buts du projet sera d'écrire la méthode (ou d'une de ses variantes) et la paralléliser, sur CPU et GPU.

3 Projet

3.1 Modalités

Le projet devra être conduit par binôme et devra fournir deux implantations :

1. OpenMP tâches explicites ;
2. Kokkos.

Il faudra aussi :

— réaliser un rapport expliquant les choix algorithmiques et d'implantation, mais aussi présentant des résultats de performances. On pourra notamment commenter les différences entre les 2 versions ;

— présenter le travail effectué et les résultats obtenus (20 minutes de présentation, 10 minutes de questions).

3.2 Aspects pratiques

— Les ressources informatiques utilisées pour la collecte de performances seront celles accessibles sur Romeo.

— Les technologies utilisées seront OpenMP, notamment les concepts de tasks et de targets, ou Kokkos. Les choix d'implantations seront justifiés dans le rapport et lors de la soutenance.

— Le code source devra être fourni (et compilable et exécutable sur Romeo) lors de la remise du rapport.

3.3 Suggestions

3.3.1 Remarques générales

L'obtention de meilleures performances peut passer par une meilleure exploitation du parallèlisme hiérarchique des bibliothèques de parallélisation. Il pourra être nécessaire de créer une décomposition hiérarchique de la boucle de calculs.

3.3.2 Exploitation simultanée

Le débruitage d'une image peut se faire en niveau de gris mais aussi sur les différents canaux de couleurs simultanément.

Dans ce dernier cas, vous pouvez exploiter les capacités de groupement de données de la plateforme de parallélisation que vous utilisez.

3.3.3 Suggestions pour la parallélisation de Gauss-Seidel

L'algorithme de Gauss-Seidel est un peu plus délicat à paralléliser que celui de Jacobi, expliquez pourquoi.

Il existe cependant différentes méthodes, que nous allons juste un peu introduire pour faciliter la réalisation du projet.

1. Approche par fronts

   Dans cette approche il peut être intéressant de modéliser le graphe de tâches de l'algorithme et mettre en place une implantation *task-based*.

2. R-B Gauss-Seidel

   Cette méthode de parallélisation modifie le comportement de l'algorithme mais les propriétés de convergence de la méthode restent bonnes par rapport à Jacobi.

   L'idée de cette méthode est de décomposer les données à calculer en 2 catégories par un coloriage *Red-Black* de la matrice.

   La boucle de l'algorithme est alors scindée en 2 boucles, une par couleur et chacune de ces boucles applique une itération type Jacobi sur les données les plus récentes disponibles.

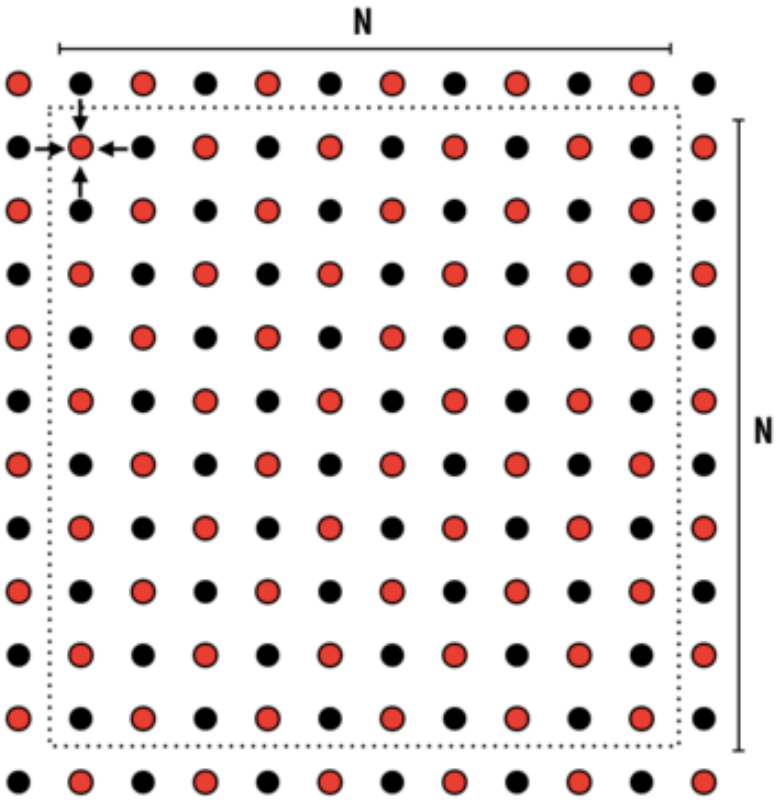

FIGURE 3 – Coloriage *Red-Black* pour l'algorithme de Gauss-Seidel