PAH: Débruitage avec Filtre Bilatéral en CUDA	
Thierry Garcia - Jordan Rey-Jouanchicot	Programmation sur GPU
ING3	Année 2024–2025

1 Description et déroulement du TP

Ce TP consiste à écrire un programme de traitement d'images sur GPU. Le TP devra être rendu le 25 mars avant 18h.

Les livrables attendus sont:

— Le programme source commenté une version C initiale et la version CUDA.

— Un document expliquant les choix effectué pour faire la première version du code sur CPU, puis pour la version CUDA. Ainsi que le facteur d'accélération du code obtenu et les performances mesurées. Inclure les informations nécessaire pour la compilation et l'utilisation du programme.

— Une image sur laquelle le code a été testé.

L'objectif de ce TP est de se familiariser avec la programmation hétérogène, et en particulier *La programmation sur GPU* en implémentant un *filtre bilatéral* en *CUDA*.

L'exercice se déroule en plusieurs étapes :

— Implémentation séquentielle en **C**

— Portage vers **CUDA**

— Evaluation et optimisation des performances

— Mini-rapport

2 Contexte et problématique

Les filtres de lissage conventionnels, tels que le flou gaussien, tendent à **altérer les contours** des images. En revanche, le **filtre bilatéral** permet de **réduire le bruit** tout en **préservant les détails**. Ce filtre est largement utilisé en **vision par ordinateur**, **imagerie médicale** et **traitement HDR**. La Figure 1 montre un exemple de résultat de ce filtre.

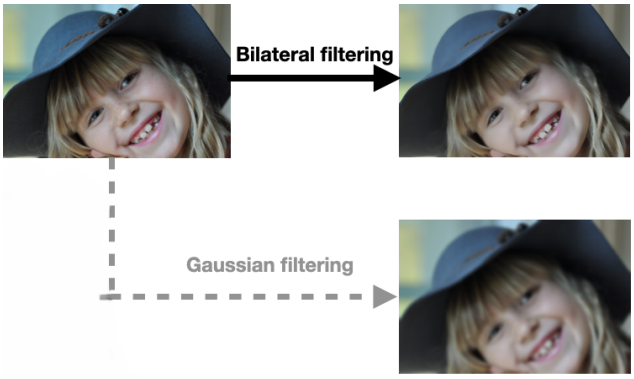

FIGURE 1 – Comparaison entre un filtre Gaussien et un filtre billatéral.

3 Les mathématiques

Le filtre bilatéral peut-être défini comme suit :

$$I'(x) = \frac{1}{W(x)} \sum_{i \in d} I(i) \cdot e^{-\frac{|x-i|^2}{2\sigma_s^2}} \cdot e^{-\frac{|I(x)-I(i)|^2}{2\sigma_r^2}}$$

où :

— $I(x)$: **Intensité initiale** du pixel x.

— $I'(x)$: **Intensité filtrée** du pixel x.

— $W(x)$: **Facteur de normalisation**, défini par :

$$W(x) = \sum_{i \in d} e^{-\frac{|x-i|^2}{2\sigma_s^2}} \cdot e^{-\frac{|I(x)-I(i)|^2}{2\sigma_r^2}}$$

— σ_s: **Paramètre de lissage spatial**, qui contrôle l'influence des pixels en fonction de leur distance à x.

— σ_r: **Paramètre de préservation des contours**, qui contrôle la sensibilité aux variations d'intensité.

— d: **Fenêtre de filtrage**, définissant l'ensemble des pixels i pris en compte autour de x.

Afin de traiter une image en couleur, vous pourrez soit traité les channels indépendamment, soit faire le calcul pour l'image en niveau de gris, puis l'appliqué à chaque channel.

4 Déroulé du TP

Afin de vous aider dans le déroulement du TP, voici une proposition de découpage du sujet.

4.1 Compréhension du Sujet

— Analyse et compréhension du sujet: **filtre bilatéral**.

— Définition des **critères d'évaluation de performance**.

4.2 Implémentation séquentielle en C

— Chargement et manipulation d'une image puis sauvegarde de celle-ci.

— Implémentation d'un **filtre bilatéral** en C, sans utiliser de librairies tels que OpenCV.

— Vérification du bon fonctionnement sur une image bruitée.

4.3 Portage vers CUDA

— Analyse des **difficultés de parallélisation** (fenêtre de voisinage, accès mémoire).

— Portage de l'algorithme en **CUDA**.

— Mesure des performances CPU vs GPU.

4.4 Optimisation et analyse

— Optimisation et amélioration de l'implémentation CUDA.

— Expérimentation avec différentes tailles de blocs CUDA.

— Analyse des performances :

  — Performances CPU vs GPU.

  — Impact de la mémoire partagée sur l'efficacité.

  — Influence des paramètres σ_s, σ_r sur le filtrage.

4.5 Rédaction du rapport

Le rapport doit inclure :

— Présentation des **résultats obtenus** et comparaison CPU vs GPU.

— Discussion sur les **difficultés rencontrées et solutions appliquées**.

5 Critères d'évaluation

Voici les critères de notations du TP.

— Implémentation correcte en **C** (10%).

— Portage **CUDA** fonctionnel et efficace (50%).

— Optimisation et analyse des performances (20%).

— Clarté et structuration du rapport (20%).

Un TP auquel une des parties de l'évaluation n'a pas été réalisé sera noté sur 10.

6 Ressources et outils

— Documentation officielle **CUDA** : `https://developer.nvidia.com/cuda-zone`.

— Cours sur les filtres bilatéraux: `https://people.csail.mit.edu/sparis/bf_course/`.

— Lecture et traitement d'image: `https://docs.opencv.org/4.x/d1/dfb/intro.html`