Programmation GPGPU avec CUDA - TP Projet

Égalisation d'histogramme

maxime.maria@unilim.fr
cyprien.plateauholleville@unilim.fr

En imagerie numérique, l'égalisation d'histogramme est une méthode d'ajustement du contraste d'une image donnée (cf. `http://en.wikipedia.org/wiki/Histogram_equalization`). Pour une image en niveaux de gris, l'idée est de calculer un histogramme comptant l'utilisation de chaque niveau de gris, de calculer la fonction de répartition de cet histogramme, puis d'étaler les niveaux de gris utilisés.

Plus précisément, soit $\{x_i\}$ l'ensemble des pixels d'une image définie sur L niveaux de gris. L'histogramme est un tableau comptant les occurrences de chaque niveau de gris l, pour $l \in [0 \dots L-1]$:

$$h(l) = \sum_{i=0}^{n-1} \delta(x_i - l)$$

où n est le nombre de pixels de l'image, et δ est la fonction de Dirac telle que :

$$\delta(\xi) = \begin{cases} 1 \text{ si } \xi = 0, \\ 0 \text{ sinon.} \end{cases}$$

La fonction de répartition r est définie sur l'intervalle des niveaux de gris comme la somme des nombres d'occurrence des valeurs précédentes :

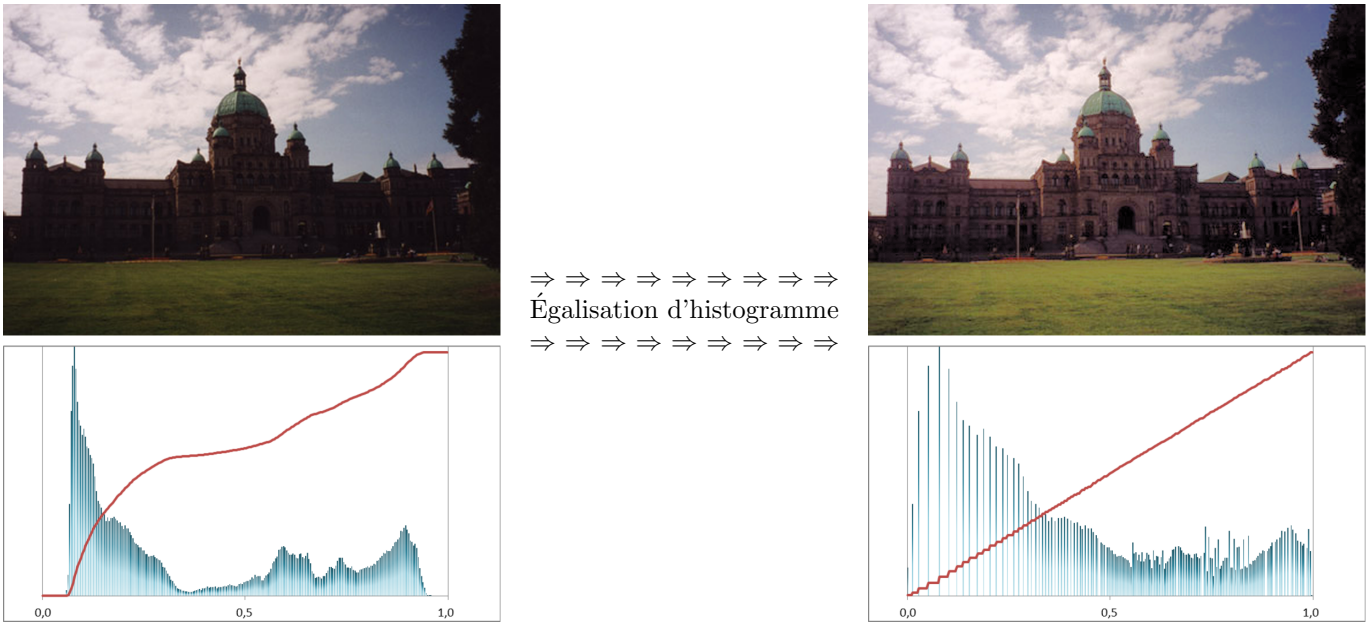
$$r(l) = \sum_{k=0}^{l} h(k).$$

Pour "étaler" l'histogramme, il suffit d'appliquer la transformation suivante :

$$T(x_i) = \frac{L-1}{Ln} r(x_i).$$

Cette méthode est étendue aux images couleurs en appliquant cette transformation sur la composante "intensité" (V) de la couleur exprimée dans le repère HSV : Hue (Teinte), Saturation et Value (*cf.* `http://en.wikipedia.org/wiki/HSL_and_HSV`).

En exemple :



Consignes

L'objectif de ce projet est de vous faire coder l'égalisation d'histogramme sur des images couleurs en CUDA. **Nous attendons que vous appliquiez les notions d'optimisation vues durant la séance précédente.**

S'il y a des optimisations à faire, particulièrement en CUDA, elles doivent avoir du sens et être justifiées un minimum. **Important :** cinq algorithmes ($\times 2$) doivent être codés obligatoirement mais vous avez le droit de laisser les différentes versions écrites dans le code final. Cela permet de voir l'évolution du processus d'optimisation ou de compréhension du problème. Vous pouvez ajoutez des suffixes aux noms des fonctions pour indiquer tout cela.

Ce travail est à réaliser en binôme.

À coder

Vous disposez de la classe `Image` pour pouvoir charger et sauvegarder des images au format `.png`.

1. Implémentez un *kernel* `rgb2hsv` qui, pour chaque pixel de l'image, calcule sa valeur dans l'espace HSV, et répartit le résultat dans trois tableaux différents.
2. Implémentez la transformation inverse (`hsv2rgb`), de HSV vers RGB (donc de trois tableaux vers un seul).
3. Implémentez un *kernel* `histogram` qui, à partir de la composante V de chaque pixel, calcule l'histogramme de l'image.
4. Implémentez un *kernel* `repart` qui, à partir de l'histogramme, applique la fonction de répartition $r(l)$.
5. Implémentez un *kernel* `equalization`, à partir de la répartition précédente, "étaler" l'histogramme.

N.B. Il serait judicieux de commencer par la version séquentielle...

À rendre

Vous devez rendre un dossier nommé `GPGPU_Nom1_Nom2.zip` contenant :

1. **10 points** - Votre code qui doit compiler et fonctionner sur les machines de la fac (I211/I212) ;
2. **10 points** - Un rapport au format PDF dans lequel vous expliquerez votre code et **justifierez vos choix**. Vous discuterez notamment de la répartition des *threads* sur la grille de calcul. Vous comparerez les versions séquentielles et parallèles. Si vous avez plusieurs versions d'un même *kernel*, comparez les performances et analysez-les (*e.g.* utilisation de la mémoire partagée ou non, différentes méthodes de synchronisation...).

Attention : Comme noté ci-dessus, la répartition des points entre le rapport et le code est homogène. Une très bonne implémentation sans analyse pertinente (ou l'inverse) ne permettra donc pas d'obtenir une bonne note. Il vous est fortement conseillé d'utiliser les méthodes d'analyse du TP3-Histogramme (Génération automatique des benchmarks, production de courbes permettant une analyse plus facile, etc.).

Vous devez rendre votre travail avant le **04 janvier 2023, 23h59**.
Tout retard entraînera une pénalité de $2^{nombre\_de\_jours\_de\_retard}$ points.

Attention au plagiat !

Vous le savez, vous avez même signé un engagement concernant le plagiat/la fraude à l'Université... Mais une piqûre de rappel ne fait pas de mal ! Faites très attention, la fraude peut entraîner jusqu'à l'exclusion du système universitaire français !