Travaux Pratique 8

Ce sujet est en lien avec le quatrième chapitre du cours, et concerne la programmation CUDA. Les mêmes commentaires que ceux des derniers TP s'appliquent ici aussi.

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 noté 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 :

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

Eh oui, c'est une somme préfixe, donc un SCAN inclusif ! La transformation suivante permet « d'étaler » l'histogramme :

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

Notez un point important pour T : la division requière un calcul soit en virgule flottante, soit en l'effectuant en toute fin du calcul : d'abord le quotient $(L-1) \times r(x_i)$, puis la division.

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). Avec des images 24 bits, la valeur s'exprime sur 1 octets ; donc L vaut 256 (et $L - 1 = 255$).

Nous allons jouer avec l'implantation de l'histogramme et le scan inclusif.

ATTENTION : pour rappel, tous les travaux non rendus sont transformés en note 0. Si un tiers ou plus des TP ne sont pas rendus (à partir de 4 TP), la note attribuée à l'épreuve sera ABI sans contrôle de rattrapage, conduisant de fait à l'échec au module et à l'année.

> Votre travail est à rendre (il le sera à chaque fois) sous la forme du code (et uniquement la partie « student ») accompagnée d'un rapport au format PDF, le tout dans une archive compressée au format ZIP. Ne respecter pas ces contraintes, et votre note en sera diminuée de quelques points.
>
> **Vous ne devez modifier que ce qui se trouve dans le répertoire « `./student/` » !**
>
> Notez que chaque exercice vient avec un squelette, qui s'occupe de la ligne de commande, du lancement de votre code (défini dans une classe particulière), et d'une vérification sommaire du résultat (lorsque c'est possible). Utilisez l'option « `-h` » ou « `--help` » pour connaître le fonctionnement de la ligne de commande …

Exercice 1

Implémentez la fonction `rgb2hsv` qui, pour chaque pixel de l'image, calcule sa valeur dans l'espace HSV en utilisant la fonction `RGB2HSV`, et **répartit le résultat dans trois tableaux** différents. Notez qu'il s'agit d'une forme de `SCATTER`. Ce type de répartition en trois tableaux vise à optimiser le débit mémoire d'un kernel CUDA (encore la coalescence).

Implémentez la transformation inverse (`hsv2rgb`), de HSV vers RGB, en utilisant la fonction `HSV2RGB`.

Exercice 2

Calculez l'histogramme des valeurs.

Exercice 3

Calculez la fonction de répartition.

Exercice 4

Calculez la transformation finale.

Admirez.

Comme toujours, votre rapport doit discuter les performances en fonctions des nombres de threads et de leur répartition.

Résultats

Image “*Fragornard,_The_Swing.ppm*”

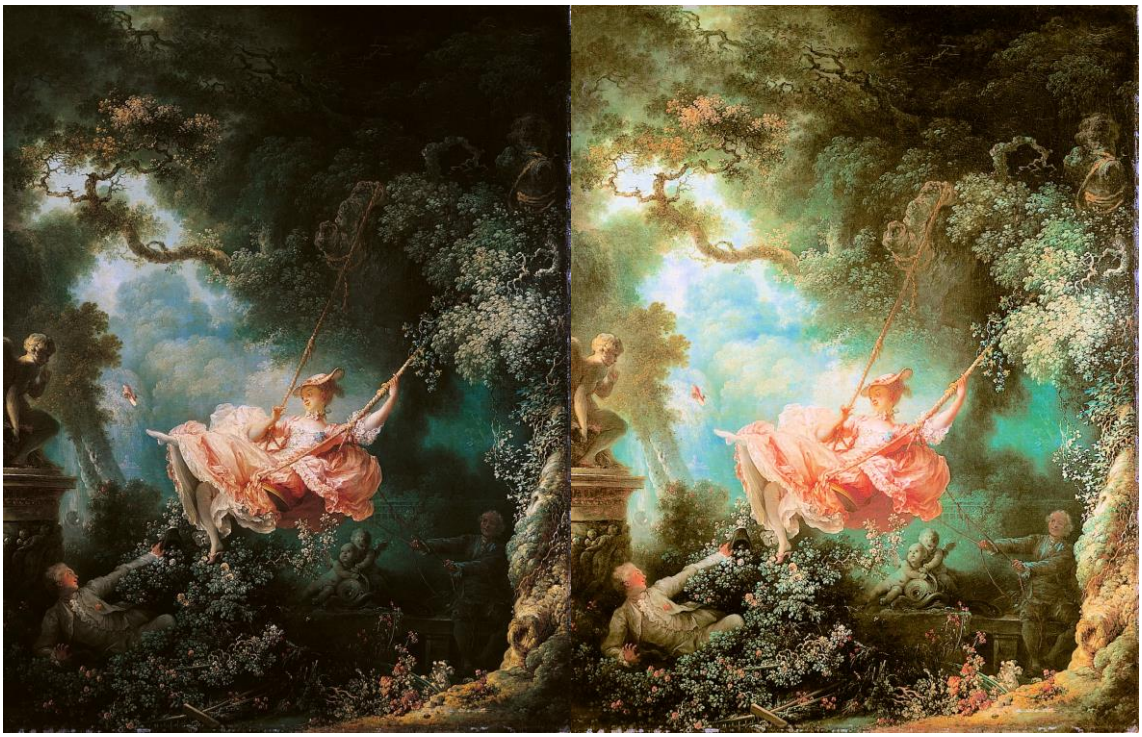

Image « Roy_Lichtenstein_Drowning_Girl.ppm »

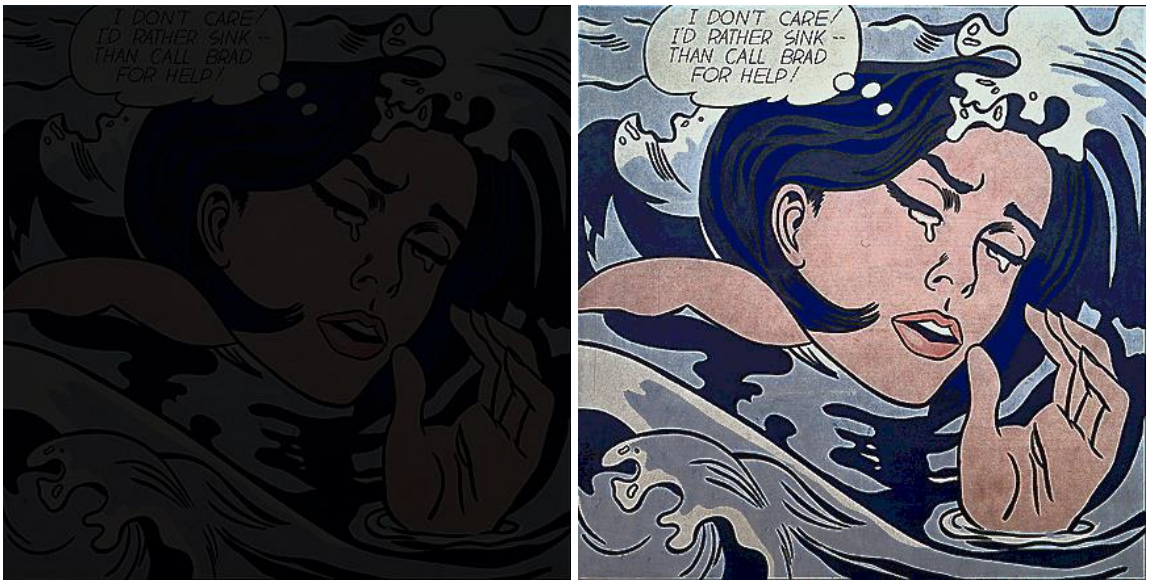

Image “Las_Meninas,_by_Diego_Velasquez,_from_Prado_in_Google_Earth.ppm”

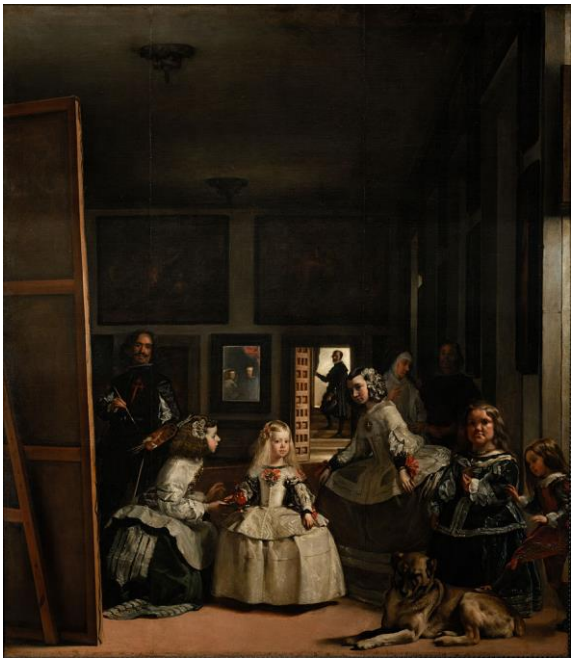


Image “Caravagio_-_La_vocazione_di_San_Matteo.ppm »