Traitement d'image avec Open-CL

Introduction:

Le traitement d'image nécessite d'effectuer un grand nombre de calcul. En multipliant le nombre de pixels par la taille de notre filtre, les calculs s'alourdissent très vite et ralentissent énormément le programme. Pour se faire, on peut utiliser le GPU qui possède un nombre de threads très important et qui va pouvoir parcourir l'image et traiter les calculs en parallèles. Le problème porte sur le filtrage par convolution. L'objectif de ce TP sera d'effectuer ce filtrage en parallèle et de l'optimiser.

Filtrage :

Le premier filtrage est un filtre moyenneur effectué sur une taille N.
On obtient alors l'image suivante :

Le deuxième filtre est un filtre gaussien dont on peut faire varier la valeur de gamma.

L’image filtrée obtenue est alors celle-ci :

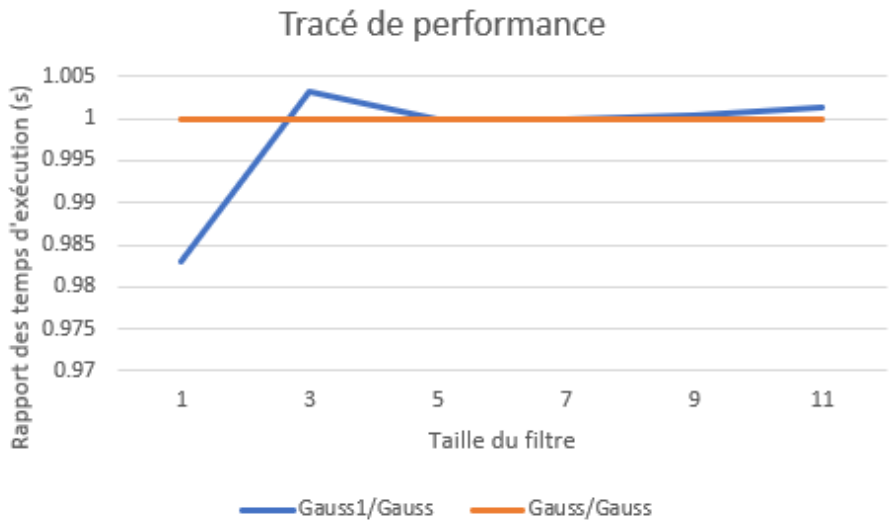
Optimisation :

Bien que les temps de calcul restent faibles. Ceux-ci peuvent augmenter rapidement avec la taille du filtre.

Afin d'accélérer le programme, on peut effectuer des modifications et on trace rapport entre la performance du filtre modifiée et celle du filtre naïf en fonction de la taille du filtre.

Première modification :

Pour réduire le nombre de calcul du coefficient a = 1/racine(2pi). On rentre le coefficient a dans les arguments du kernel. L'amélioration est très faible.



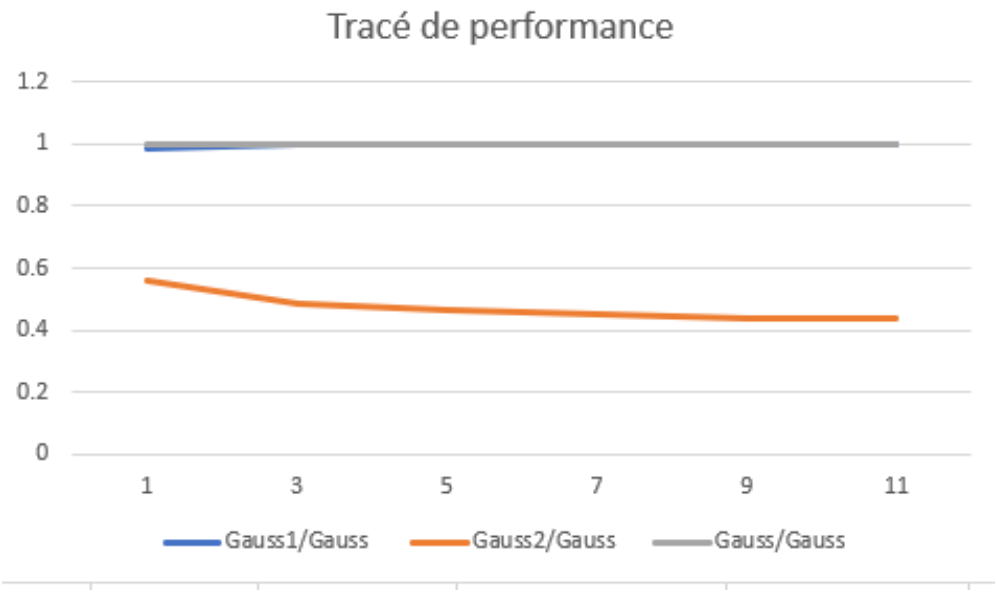
Gauss1 apporte une très faible amélioration pour un N faible (N étant la taille du filtre) et on observe une augmentation du temps lorsque N augmente. Cela peut être dû au fait que le temps de calcul est plus rapide que la somme des temps d'accès à la mémoire.

Deuxième modification :

En plus d'enlever le calcul de a dans le kernel, on peut calculer les exponentiels avant le kernel.

On calcule l'exponentielle sur un tableau de taille 2N+1 qui renvoie exp(-l^2/2*sigma^2)

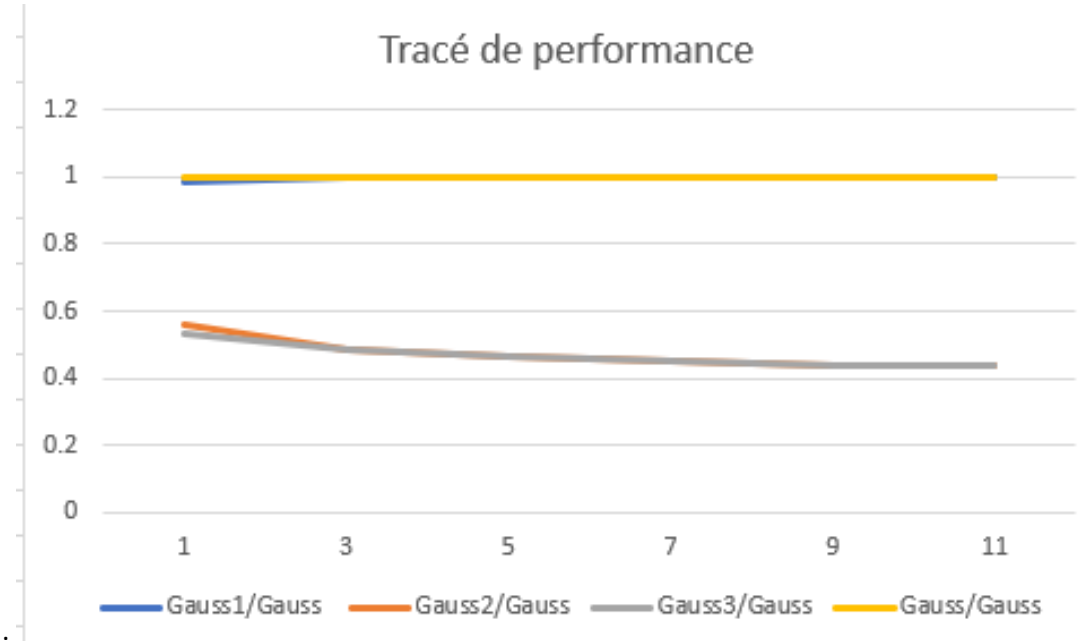
Pour ce faire, on a dû charger un tableau de taille 2N+1 dans le kernel. (On avait initialement chargé un tableau de taille (2N+1)^2 avec toutes les valeurs de l'exponentielle, cette modification a permis d'augmenter les performances).
Cela permet de réduire considérablement le temps de calcul. Il est alors divisé par deux.



On observe une forte amélioration du temps de calcul, cela est du à la réduction du nombre de calculs de e.

<u>Troisième modification</u> :

Elle consiste à mettre e sur la mémoire locale car elle est plus rapide d'accès que la mémoire globale.



.

On remarque que l'amélioration est très faible, on aurait pu mettre en constante imageInput et imageOutput afin d'avoir de meilleurs résultats.

Quatrième modification :

On utilise ici le principe de vectorisation grâce à float4. En effet, cela parallélise notre boucle.
Pour ce dernier cas traité, on a une erreur lors de l'exécution, on ne peut donc effectuer le tracé et vérifier les performances.

Conclusion :

Notre optimisation, notamment sur la valeur de e a permis d'améliorer grandement l'efficacité de notre programme. D'autres optimisations, tels que la vectorisation, le chargement sur la mémoire locale de ImageInput et ImageOutput, sont encore possible.

Pour conclure, OpenCL est un outil permettant de travailler sur le GPU et d'effectuer une parallélisation, il est très utile et très pour le traitement d'image. Mais cet outil nécessite un travail d'optimisation, qui pourront permettre de faire varier considérablement les temps de calculs.