BM3D "Image denoising by sparse 3D transform-domain collaborative filtering" sur GPU

Stéphane Cuenat

January 17, 2016

Abstract

Dans ce papier nous proposons une implementation open-source de l'algorithme BM3D porté sur GPU. Nous discutons du choix de l'ensemble des paramètres et de l'influence de ceux-ci . Chaque phase est étudié séparément afin d'en déterminer les valeurs optimales en ce qui concerne la qualité et la rapidité de l'algorithme. Nous démontrons que BM3D est parallelisable, donc exécutable dans un monde GPU.

Chapter 1

Introduction

Collaborate Filtering est le nom de la méthode the groupage et de filtrage de l'algorithme BM3D. Cette méthode est réalisée en 4 étapes: 1) Trouver a bloc (portion de l'image) similaire à un bloc de référence et ensuite grouper les ensemble pour former un bloc 3D. 2) Appliquer une transformée 3D sur l'ensemble des bloc 3D. 3) Réduire les coefficient du monde spectrale. 4) Appliquer la transformée 3D inverse.

En atténuant le bruit, le *Collaborate Filtering* révèle les plus petits détails partagés par les groupes de blocs. Chaque bloc filtré est alors remit à sa position d'origine. Sachant que deux bloc peuvent se chevaucher, nous pouvons obtenir plusieurs estimations pour un pixel donné. C'est pourquoi, nous devons les combiner. Cette étape finale est l'*Aggrégation*.

Le premier filtre collaboratif est considérablement amélioré par un second filtre *Wiener Filtering*. Cette seconde étape mime la premières étapes avec deux différences. Le *Block-Matching* est appliqué sur l'image filtré et non sur l'image bruité. Nous n'appliquons plus un filtre *Hardthreshold* mais un filtre *Wiener Filtering*. L'étape finale d'*Aggrégation* reste inchangée.

L'algorithme BM3D détaillé ici est directement tiré de l'article originale [1]. Plusieurs analyses préalable de l'algorithme BM3D démontre une diminution de la performance de débrouillage lorsque la déviation standard du bruit dépasse 40. Il a été démontré que pour de grande valeurs de bruit (standard deviation), les paramètres doivent être revu. La seuil de filtrage de la première phase doit être augmenté. Il a aussi été démontré que les transformées 2D appliquées peuvent influencer

l'algorithme. Dans [2], ils montrent que la meilleurs qualité d'image est atteinte en appliquant une transformée *2D bi-orthogonal spline wavelet*.

Dans ce papier nous nous sommes focaliser sur la qualité, mais plus particulièrement sur l'implémentation sur GPU (parallélisation de l'algorithme BM3D). C'est pourquoi nous nous appliquons des DCT-2D. Pour ce qui est de la transformé 1D selon Z, nous avons la transformée d'Hadamar-Welsh [3]. Nous verrons que cette méthode est efficace sur GPU.

Dans le chapitre 2, nous présentons en détails l'algorithme BM3D. Le chapitre 3 est dédié à la réalisation sur GPU. Au chapitre 4, nous parlerons des résultats obtenus en comparaison au travaux mené par [4]. Nous présentons nos conclusions et les prochaines étapes au chapitre 5.

Chapter 2

Algorithme BM3D

2.1 Concepts générals

2.2 Phase 1 - Estimation basic

2.2.1 Block matching

2.2.2 Filtre "Hardthreshold"

2.2.3 Aggregation

2.3 Phase 2 - Estimation finale

2.3.1 Block matching

2.3.2 Wien filter

2.3.3 Aggregation

Chapter 3

Réalisation

3.1 Block matching

3.1.1 Calul de la distance euclidienne

3.1.2 Tri des blocks

3.2 "3D transfom"

3.3 Filtre "Hardthreshold"

3.4 Filtre de "Wien"

3.5 Aggregation

Chapter 4

Résulats

4.1 BM3D vs BM3D-GPU

4.2 Influance de la taille de la fenêtre de recherche

4.3 Influance de la Kaiser-Window

4.4 Influance de la transformée 2D sur les blocks avant "Block-Matching"

4.5 Influance du calcul de distance sur la rapidité de l'algorithme

4.6 Influance du tri des block sur la rapidité de l'algorithme

Chapter 5

Conclusion

5.1 Block matching