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

Stéphane Cuenat

30 janvier 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 parallélisable, donc exécutable dans un monde GPU.

Table des matières

— 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 blocs 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. Le 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). Pour une question de simplicité, nous avons décidé d'appliquer des DCT-2D (même si nous savons que nous pouvons atteindre une meilleure qualité en appliquant d'autres transformées lors de la phase 1 et 2). Nous allons montrer dans les chapitres suivants que nous avons une bonne qualité en comparaison à [4]. Pour ce qui est de la transformé 1D selon Z, nous avons opté pour la transformée d'Hadamar-Welsh [3]. Nous verrons que cette méthode est efficace sur GPU. Dans [1], l'algorithme (écrit avec Matlab) est capable de traiter une image en 4 secondes. Ce temps est relativement long. Le but de ce papier est de montrer que nous pouvons profiter de la puissance du GPU pour réduire ce temps à quelques milisecondes, ce qui revient à montrer que BM3D peut être paraléliser.

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 présentons nos résultats obtenus en comparaison aux travaux menés par [4]. Nous présentons nos conclusions et les prochaines étapes au chapitre 5.

— 2 —

Algorithme BM3D

2.1 Concept général

Dans ce chapitre nous allons présenter les différentes étapes de l'algorithme BM3D sur des images de niveau de gris. Concernant l'application de l'algorithme sur des images couleurs, vous trouverez une explication dans [2]. Dans la suite, nous assumons que le bruit est un bruit Gaussien et sa variance est equal à σ^2 (σ étant la déviation standard du bruit appliqué sur l'image originale).

L'algorithme est divisé en deux étapes:

1. La première étape consiste à estimer le bruit par "hard thresholding" lors du filtrage collaboratif. Les paramètres de cette étapes sont représentés par l'exposant **hard**.
2. La deuxième étage est basée sur l'image bruité et sur l'estimation issue de la première étape. Cette étape utilise un filtre "Wiener Filter" lors du filtrage collaboratif. Les paramètres liés à cette étape sont représentés à l'aide de l'exposant **wiener**.

2.2 Phase 1 - Estimation basic

On dénote par P le patch (bloc) de référence de taille $K^{hard} * K^{hard}$.

Groupement (”Block matching”)

Cette première sous-étape regroupe l'ensemble des patches similaires à chaque patch de référence P. Pour chaque patch de référence P, on recherche les patches Q similaire à P dans une fenêtre centrée sur P de taille $n^{hard} * n^{hard}$. Le groupe de patches est défini par:

$$P(P) = \{Q : d(P,Q) \leq \tau^{hard}\} \tag{2.1}$$

Ou

- τ^{hard}: est la distance limite pour laquelle deux patches sont considérés comme similaire.
- $d(P,Q) = \frac{\|\gamma'(P)-\gamma'(Q)\|_2^2}{(k^{hard})^2}$ est la distance quadratique normalisée entre deux patches[1].
- γ' est le seuil du filtre qui est equal à $\lambda_{2D}^{hard}\sigma$. Pour un $\sigma \leq$40, le seuil est equal à $\lambda_{2D}^{hard}\sigma$ = 0. Pour $\sigma \leq$40, tous les coefficients sont inchangés[2].
- σ^2 est la variance du bruit Gaussien appliquée à l'image.

Le groupe 3D représenté par P(P) est construit en regroupant les patches similaire P(P). Pour augmenter la performance de l'algorithme, seulement les N^{hard} patches de P(P) qui sont les plus proches du patch de référence P sont gardé pour formé le groupe 3D[3]. L'ordre des patches dans le groupe 3D n'a aucune influence sur le résultat. Ceux-ci peuvent être désordonnés.

[1] Nous avons ici calculé la distance Euclidienne carrée.

[2] Nous verrons au chapitre 3 (Réalisation) que nous avons omis cette étape ayant pris en compte des images avec un $\sigma \leq$30 pour nos tests.

[3] Au chapitre 3 nous verrons que N^{hard} doit être une puissance de 2.

Filtre collaboratif

Une fois les groupe 3D construit, le filtre collaboratif est appliqué. On applique une transformée linéaire selon Z (profondeur du groupe 3D). Cette étape est suivie par un seuillage des coefficients du domaine spectrale. Pour finir nous appliquons une transformée 3D inverse afin de connaître l'estimation pour chaque patch du groupe.

$$P(P)^{hard} = \tau_{3D}^{hard^{-1}}(\gamma(\tau_{3D}^{hard}(P(P)))) \tag{2.2}$$

Ou γ est l'opérateur de seuil avec une limite $\lambda_{3D}^{hard}\sigma$:

$$\gamma(x) = \begin{cases} 0 & \text{si } x \leq \lambda_{3D}^{hard}\sigma \\ x & \text{sinon} \end{cases}$$

Pour une raison pratique que nous démontrerons dans les prochains chapitres, la transformée 3D τ_{3D}^{hard} du groupe 3D P(P) est effectuée en deux étapes. Nous appliquons une première transformée 2D dénoté τ_{2D}^{hard}sur chaque patch. Ensuite une transformée 1D dénoté par τ_{1D}^{hard} est appliquée sur la troisième dimension du groupe 3D (profondeur).

Agrégation

La dernière étape est l'agrégation. Nous avons une estimation pour chaque patch et donc une estimation pour chaque pixel. Ces estimations sont sauvegardées dans un buffer ainsi défini:

$$\forall Q \in P(P), \forall x \in Q, \begin{cases} v(x) = v(x) + w_P^{hard} u_{Q,P}^{hard}(x) \\ \delta(x) = \delta(x) + w_P^{hard} \end{cases}$$

Ou:

- $v(resp.\,\delta)$ représente le numérateur (resp. le dénominateur) de l'estimation basic de l'image obtenue à la fin de l'étape de groupage.
- $u_{Q,P}^{hard}$ est l'estimation du pxiel x qui appartient au patch Q obtenu lors de la phase du filtre collaboratif du patch P.

L'intérêt de cet méthode est qu'elle donne aux patches homogènes (ou beaucoup de coefficient ont été éliminés). Un patch avec des effets de bords sera moins pris en compte que des patches homogènes. La figure ci-dessous montre cette effet. L'algorithme donne plus d'importance aux patches vert durant la phase d'agrégation.

Afin de réduire les effets de bords qui peuvent apparaître, nous appliquons une Kaiser window ($k^{hard} * k^{hard}$) après la transformée 3D inverse sur chaque patch. Cela revient à multiplier chaque coefficient de la Kaiser window avec le coefficient du patch [4].

L'estimation basic obtenue après cette première étape est donnée par:

$$u^{basic}(x) = \frac{\sum_{P} w_P^{hard} \sum_{Q \in P(P)} X_Q(x) u_{Q,P}^{hard}(x)}{\sum_{P} w_P^{hard} \sum_{Q \in P(P)} X_Q(x)} \tag{2.3}$$

Avec $X_Q(x)$ définit ainsi:

$$X_Q(x) = \begin{cases} 1 & \text{si et seulement si } x \in Q \\ 0 & \text{sinon} \end{cases}$$

Ce résultat est simplement obtenue en divisant les deux buffers préalablement calculés pour chaque pixel de l'image original:

$$u^{basic}(x) = \frac{v(x)}{\delta(x)} \tag{2.4}$$

[4] Nous montrerons au chapitre 4 (Résultat) que cette étape influence peu le résultat final comme mentionné dans [2].

2.3 Phase 2 - Estimation finale

Groupement (”Block matching”)

Filtre collaboratif

Agrégation

— 3 —

Réalisation

3.1 Block matching

Calul de la distance euclidienne

Tri des blocks

3.2 ”3D transfom”

3.3 Filtre ”Hardthreshold”

3.4 Filtre de ”Wien”

3.5 Aggregation

— 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

— 5 —

Conclusion

5.1 Block matching

Liste des figures

Bibliographie

[1] V. Katkovnik K. Dabov, A. Foi and K. Egiazising. Image denoising by sparse 3d transform-domain collaborative filtering, August 2007.

[2] Marc Lebrun. An analysis and implementation of the bm3d image denoising method, 2012.

[3] TAMPERE UNIVERSITY OF TECHNOLOGY. Image and video denoising by sparse 3d transform-domain collaborative filtering.

[4] Wikipédia. Transformée de hadamard.