

Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries

Cloup Félix - Lesaffre Maxime - Ruiz Yoann
Savio Arthur - Teste Alexis

29 avril 2020

Résumé

Nos différents travaux s'articulent autour de la reconstitution d'images bruitées. Pour cela, nous avons pu utiliser le cas concret du glacier Austre Lovénbreen dont la base de données est remplie de plus de 60 000 images. Le but de la reconstruction des images est de pouvoir étudier l'évolution de ce glacier ainsi que le bilan hydrologique de son bassin versant.

1 Contexte, motivations et objectifs

La base de données d'images que nous avons en notre possession contient de nombreuses images bruitées en raison de différents facteurs (météo, intervention de techniciens, animaux, etc.). Dans ce projet, nous avons souhaité éliminer toutes ces perturbations. Ainsi, nous proposons une nouvelle approche basée sur l'utilisation de dictionnaires appris par l'algorithme K-SVD. Le dictionnaire pour la représentation parcimonieuse est une bonne application pour le débruitage.

Il est intéressant de voir l'efficacité de cette méthode pour cet exemple du glacier. Dans un autre domaine (médical par exemple), l'utilisation de ces outils pourrait avoir une nette importance.

L'objectif principal est donc de restaurer une image dégradée par un bruit par l'application de la méthode K-SVD. D'une part, le but sera d'introduire et de proposer plusieurs améliorations de la méthode et d'en établir des premières conclusions. D'un autre part, il s'agira de comparer les résultats des différentes solutions et d'en tirer des conclusions définitives sur la pertinence et l'efficacité de ces méthodes.

2 Le problème du débruitage

Notre cas n'est pas un cas unique. On retrouve dans beaucoup de domaines des images présentant des dégradations plus communément appelées bruit. L'objectif du débruitage est justement, comme son nom l'indique, une méthode permettant de supprimer ces bruits afin d'obtenir des images les plus nettes possible.

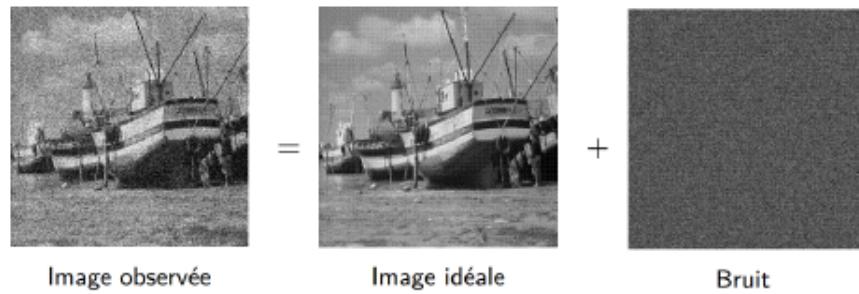


FIGURE 1 – Profil du débruitage

Les images que nous possédons sont des images couleurs RVB. Nous avons voulu tester la méthode en gardant cet aspect couleur. En général, ces travaux sont réalisés sur des images en niveau de gris. Notre base d'images est fortement constituée de neige et de montagne (noir et blanc), ce qui rend cette exploitation compliquée. Ainsi, chaque filtre de couleur constitue la même démarche, nous devons répéter l'algorithme pour chaque niveau de couleur.

C'est de cette idée que nous étions partis pour notre étude. Cependant, après de nombreux tests, nous nous sommes vite rendus compte que cela multiplié par trois le temps de calcul. Nous avons testé différentes méthodes et nous détaillerons pour chaque méthode comment nous avons préparé nos images pour être le plus efficace, performant et optimiser le calcul.

De nombreux algorithmes de débruitage existent, nous nous sommes intéressés donc à la méthode KSVD qui se base sur l'emploi de patchs (sous-images) dont nous détaillerons l'utilisation plus loin dans ce rapport. Il s'agira de former, à l'aide de l'ensemble des patchs (appelé dictionnaire), la meilleure approximation possible, sans bruit, d'une image donnée.



FIGURE 2 – Dataset

3 Méthodes

3.1 Méthode 1

3.1.1 Introduction

La première méthode propose une approche basée sur les représentations redondantes (corrélation entre les différents pixels de l'image) et parcimonieuses sur des dictionnaires appris de patchs se chevauchant en partie, suivies par un algorithme de moyennage. Un dictionnaire est appris par l'algorithme K-SVD sur des patchs d'images d'apprentissage de bonne qualité. L'utilisation de ce types de représentations redondantes et de la parcimonie est de plus en plus utilisée dans le cadre du débruitage.

Ces représentations parcimonieuses consistent à approximer un signal (représenté sous forme de vecteur) par une combinaison linéaire de quelques colonnes seulement (les atomes), constituant une matrice (le dictionnaire). Le signal est alors représenté par un vecteur parcimonieux contenant seulement un maximum de coefficients nuls.

Nous avons donc en notre possession une base d'images redondantes. Cette méthode permet de réduire ce type de redondance puisqu'on transforme l'image en une représentation où les coefficients sont peu corrélés.

3.1.2 Sélection des patchs

L'apprentissage du dictionnaire sur nos données d'entraînement permet de l'adapter à notre type de données. L'apprentissage du dictionnaire est un point clé pour rendre les atomes le plus efficace possible puisqu'un dictionnaire appris a le pouvoir d'offrir une meilleure qualité de reconstruction qu'un dictionnaire prédéfini.

Le nombre d'atomes K dans un dictionnaire n'est pas limité. Dans notre cas, le dictionnaire est sur-complet (le nombre d'atomes K pour le dictionnaire est supérieur à la dimension des atomes n) et le dictionnaire obtenu aura donc certains atomes corrélés. Il est largement possible de faire varier la taille de ce dictionnaire afin d'en voir les conséquences sur nos résultats.

De manière générale, plus le nombre d'atomes K augmente, plus les atomes du dictionnaire sont redondants et cela entraîne que beaucoup d'entre eux peuvent ne pas être utilisés et on obtient des représentations encore plus parcimonieuses.

Cependant, un dictionnaire contenant un grand nombre d'atomes est difficile à stocker et sera difficile à manipuler pour la recherche des coefficients. Il faut aussi tenir compte du coût en calculs pour la recherche de la représentation parcimonieuse des signaux et la mise à jour du dictionnaire qui augmente en fonction du nombre d'atomes K du dictionnaire.

Ici, l'idée principale utilisée est que les atomes du dictionnaire vont capturer la structure de l'image. L'image est traitée comme un ensemble de patchs (blocks d'images ou sous-images) et on apprend un dictionnaire de patchs sur notre ensemble d'images. On effectue alors la décomposition à l'aide du dictionnaire appris.

Ainsi, la première étape réalisée est l'extraction de patchs sur l'ensemble de notre base de données d'images. Pour extraire ces patchs, nous avons réalisé une sélection aléatoire de patchs sur les différentes images de façon à éviter que les patchs se chevauchent entièrement et suffisamment pour avoir des données exploitables. Nous détaillerons dans la partie résultat les différents paramètres utilisés.

Pour faire ce choix, nous avons décidé de réaliser 3 histogrammes (un pour chaque canal RGB) sur les différents patchs des différentes images. En effet, comme une image est composée de pixels possédant une valeur radiométrique pour chacun des canaux qui la composent, on peut observer la fréquence d'apparition des différents niveaux par un histogramme. Comme on s'intéressait à l'information produite totale, nous avons sommé les 3 histogrammes pour n'en obtenir qu'un seul récapitulant les 3 canaux RGB, afin de pouvoir faire notre sélection des patchs. Le but de l'analyse de cet histogramme est de détecter si l'information est peu ou pas visible sur l'image et notre sélection se base donc sur les patchs qui ont le plus d'informations différentes. Voici l'aperçu d'un histogramme pour chaque canal. Plus la fréquence est haute, plus l'information est importante dans le canal en question.

Une fois l'extraction des patchs réalisée, nous les avons centrés et réduits. Cette normalisation a pour but de compenser les variations de couleur liées aux conditions

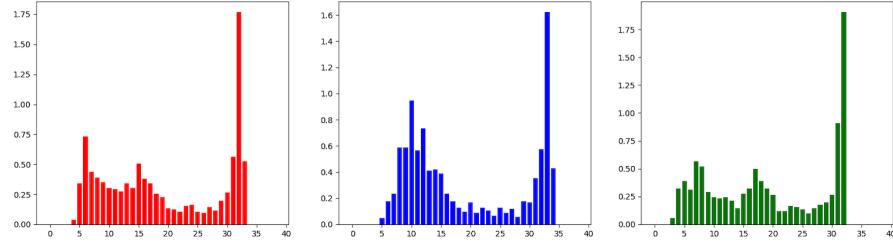


FIGURE 3 – Histogramme

d'éclairage et à l'appareil utilisé par exemple. En effet, les valeurs des couleurs d'une image dépendent de l'éclairage de celle-ci. La normalisation va ainsi reproduire le principe de constance des couleurs que l'on trouve dans le système de perception humaine : les objets paraissent sensiblement de la même couleur même si l'éclairage diffère. Nous l'avons donc appliquée patch par patch pour éviter qu'il n'y ait de trop grandes différences avec les couleurs d'origine. Voici un aperçu des patches que nous avons obtenus :



FIGURE 4 – Patchs sélectionnés

3.1.3 Apprentissage du dictionnaire sur-complet sur l'ensemble des images propres

Avant d'effectuer l'apprentissage du dictionnaire, nous avons dû vectoriser les différents patchs de blocs, $x_i = V(Xi)$, $i = 1 \dots B^2$ chacun.

L'algorithme K-SVD permet d'apprendre un dictionnaire sur-complet. Le principe est d'itérer entre l'étape de codage parcimonieux des vecteurs x_i (1) sur le dictionnaire courant (2) et l'étape de mise à jour du dictionnaire permettant d'améliorer la représentation des données. On réalise la mise à jour atome par atome en considérant les autres atomes fixes par une décomposition en valeurs singulières. Le but est de mettre à jour chaque colonne du dictionnaire afin de réduire au maximum l'erreur de reconstruction. Le but est d'itérer L fois le processus. Nous avons fixé L à 10.

(1) Nous avons choisi de réaliser l'étape du codage parcimonieux par l'algorithme OMP (Orthogonal Matching Pursuit) qui se base sur la sélection de l'atome qui contribue le plus à chaque itération. Avec cette méthode, un atome ne peut être sélectionné plusieurs fois et l'algorithme converge assez rapidement. Cela nous permet également d'obtenir la meilleure approximation possible.

(2) Le dictionnaire D0 est initialisé de façon aléatoire. Nous avions la possibilité d'initialiser notre dictionnaire par DCT (Transformation en cosinus discrète) ou aléatoirement. Nous avons donc opté pour cette dernière.

Une fois le processus terminé, nous avions donc un dictionnaire sur-complet appris par K-SVD sur notre base d'images.

3.1.4 Débruitage et reconstruction d'une image bruitée

Maintenant que notre dictionnaire a été appris par l'ensemble des images, il nous faut désormais débruiter et reconstruire une image bruitée.

Débruiter une image nécessite plusieurs étapes. Tout d'abord, il est nécessaire de charger le dictionnaire D surcomplet. Ensuite, nous décomposons l'image bruitée que l'on souhaite débruiter en patchs de façon à avoir une couverture totale de l'image grâce à ces patchs. De ces patchs, il faut en calculer la moyenne et l'écart type afin de les retirer pour l'encodage de l'image, afin d'obtenir des nouveaux patchs que l'on nommera \hat{x}_i .

C'est donc l'étape du codage parcimonieux, une nouvelle fois par l'algorithme OMP, de ces patchs bruités \hat{x}_i afin d'obtenir la décomposition offrant le niveau de parcimonie maximal. Pour cela, nous cherchons à déterminer la projection \hat{x}_i optimale, de manière à minimiser la valeur absolue de cette projection.

L'image étant bruitée, nous ne voulons pas une reconstruction à l'identique nos patchs \hat{x}_i . Finalement, nous décodons chaque patch en inversant la normalisation effectuée à l'étape précédent le codage parcimonieux. Nous obtenons alors notre image débruitée X_i .

3.2 Méthode 2

3.2.1 Introduction

Dans le second cas, l'apprentissage du dictionnaire est effectué directement sur l'image bruitée. L'algorithme d'apprentissage du dictionnaire est alors imbriqué dans l'algorithme de débruitage. Le problème du débruitage est repris en ajoutant le dictionnaire D dans les inconnues à déterminer. Les coefficients et le dictionnaire sont alors mis à jour alternativement et de façon itérative en utilisant les principes de l'algorithme K-SVD.

3.2.2 Apprentissage d'un nouveau dictionnaire à partir d'une image bruitée

Nous nous intéressons finalement à la possibilité d'apprendre directement le dictionnaire et la décomposition parcimonieuse sur l'image bruitée. Le principe reste le même que la méthode précédente, nous mettons en place un outil permettant d'effectuer l'apprentissage de nos patchs directement extraits de l'image bruitée à partir de l'algorithme K-SVD. L'intérêt ici est de pouvoir comparer les deux méthodes et en tirer des conclusions.

3.2.3 Débruitage

Le dictionnaire appris sur l'image bruitée elle-même donne globalement de meilleurs résultats que le dictionnaire appris sur la collection de patchs d'images d'apprentissage et que le dictionnaire sur-complet.

Cependant, le débruitage offre normalement de meilleurs résultats avec cette méthode uniquement lorsque les bruits sont petits. En effet, nous devrions nous apercevoir que la méthode présente un avantage en dessous d'un certain niveau de bruit, puis se détériore plus rapidement.

C'est ce que nous avons voulu vérifier avec nos propres images.

4 Résultats des méthodes

4.1 Premier essai

Notre premier essai s'est basé sur la première méthode. Nous avons voulu essayé de tester la méthode avec sélection de patchs à partir d'un histogramme RGB. Nous

avons ainsi sélectionné $l = 1024$ patchs de blocs de taille B^2 , avec le choix de $B = 32$, que l'on appellera $X_1 \dots X_l$.

Le principe de la méthode était de partir d'une base d'image non bruitée, ce que nous ne disposons pas. Nous sommes donc voulu tester sur l'ensemble de nos images en sélectionnant des patchs se chevauchant au maximum à 50% sur l'ensemble de nos images. Malheureusement, nous arrivions à des erreurs dans notre programme.

Nous avons donc opté par choisir une seule image de notre base, celle que nous considérons la plus propre. Cette fois-ci, notre algorithme a été très performant et permettait d'avoir un résultat très rapidement. Cependant, le résultat trouvé n'était absolument pas à la hauteur de nos erreurs. Nous avons alors pensé que nos travaux étaient embêtants avec les trois niveaux de couleur. C'est de là que nous sommes partis sur un second essai.

4.2 Second essai

Dans notre second essai, nous avons donc opté pour une utilisation de nos images en noir et blanc ou plus précisément en nuance de gris. Nous sommes donc repartis de notre image sélectionnée et nous avons testé de débruiter une image considérée comme en mauvais état. Pour des calculs plus rapides, nous avons réduit considérablement la taille des images et nous les avons transmuté en image carré afin d'avoir des matrices carrées et un temps de calcul rapide.



FIGURE 5 – Images sélectionnées pour le second essai

Pour calculer l'écart entre l'image bruitée et l'image reconstruite, nous utiliserons le PSNR (Peak Signal to Noise Ratio) pour avoir une indication de l'écart entre les deux images. Attention, ce chiffre ne veut pas dire que notre image reconstruite est bien reconstruite. Il nous sert uniquement à mesurer l'écart entre les deux images.

Cette fois-ci, nous avons eu des résultats cohérents. Voici un aperçu avec nos images de taille 196, 196. Nous avons fait varier nos paramètres B , taille des patchs et L itération du K-SVD.

B	L	Dictionnaire	PSNR	Image
4	5	(16,300)	36	Image assez nette mais beaucoup de bruit
8	5	(64,300)	30	Image plus floue mais moins de bruit
16	5	(256,300)	26	Image très floue mais moins de bruit apparent
4	10	(16,300)	36	Image meilleure mais toujours beaucoup de bruit
4	20	(16,300)	37	Peu d'influence en plus

Il faut trouver un juste milieu entre une image nette et une image sans bruit, difficile de trouver de bons résultats avec cette méthode. Mais voici visuellement les améliorations que nous avons pu voir en ajustant les différents paramètres. A gauche l'image bruitée d'origine, à droite l'image reconstruite à partir du dictionnaire d'une image dite propre.

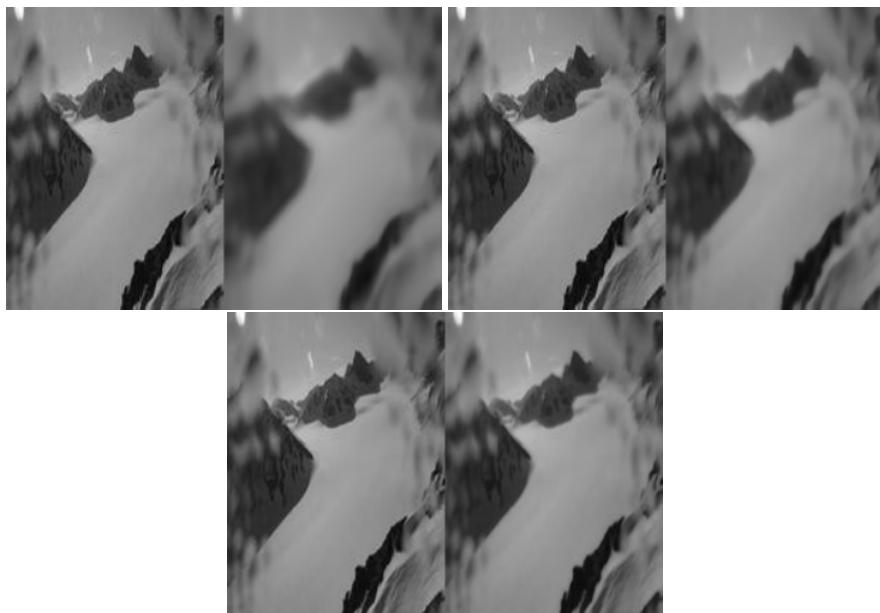


FIGURE 6 – Ajustement des paramètres

4.3 Troisième essai

Ce troisième essai a constitué de premier essai pour la seconde méthode. Nous sommes donc partis cette fois-ci de l'image bruitée pour construire notre dictionnaire. Pas besoin d'une image carré cette fois-ci, mais nous avons toutefois réduit la taille de l'image pour accélérer le processus. Voici l'image pour laquelle nous avons réalisé la méthode.



FIGURE 7 – Image sélectionnée pour le troisième essai

Pour calculer l'écart entre l'image bruitée et l'image reconstruite, nous utiliserons une nouvelle fois le PSNR. Voici un aperçu avec cette image. Nous avons fait varier nos paramètres B , taille des patchs et L itération du K-SVD, X la dimension du dictionnaire est variable en fonction des patchs sélectionnés.

B	L	Dictionnaire	PSNR	Image
4	5	(16,X)	33.04	Image assez nette avec légèrement de bruit
8	5	(64,X)	32.13	Image pas nette et du bruit
8	10	(64,X)	32.46	Image meilleure mais toujours énormément de bruit
16	10	(256,X)	30	Image nette mais le même bruit

Voici notre meilleure résultat visuellement, avec à gauche l'image bruitée et à droite l'image reconstruite :



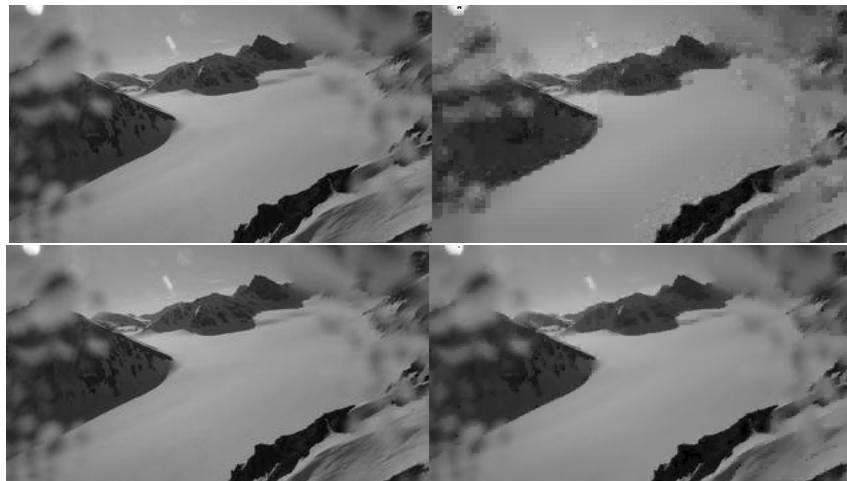
FIGURE 8 – Résultat pour le troisième essai

Au final, on ne voit que très peu de différence. Cela s'explique par le faible bruit de l'image. Nous avons décidé de tester les paramètres les plus performants sur une image plus bruitée.



FIGURE 9 – Test pour le troisième essai

Comme on peut le voir ci-dessous, en voulant ajuster la netteté de l'image et le bruit, nous n'arrivons jamais à un résultat très concluant sur une image aussi bruitée que celle-ci.



4.4 Quatrième essai

Ici, un dernier exemple qui est en fait un mixte des méthodes précédentes. En effet, le dictionnaire est ajusté sur la moitié gauche de l'image, et utilisé par la suite pour reconstruire la moitié droite où on accentue le bruit. On part du principe que le côté gauche n'est pas bruité. Le but est d'utiliser les mêmes méthodes que précédemment avec un codage parcimonieux avec OMP et un apprentissage de dictionnaire par K-SVD.

Le principe ici est d'ajuster le nombre de coefficients non nuls que non souhaitons dans notre résultat final. Nous avons donc pris une des photos que nous avons réduit légèrement afin d'accélérer une nouvelle fois les calculs.



FIGURE 10 – Image sélectionnée pour le quatrième essai

Après apprentissage du dictionnaire sur l'image et accentuation du bruit sur la droite de l'image, on a les objets suivants :

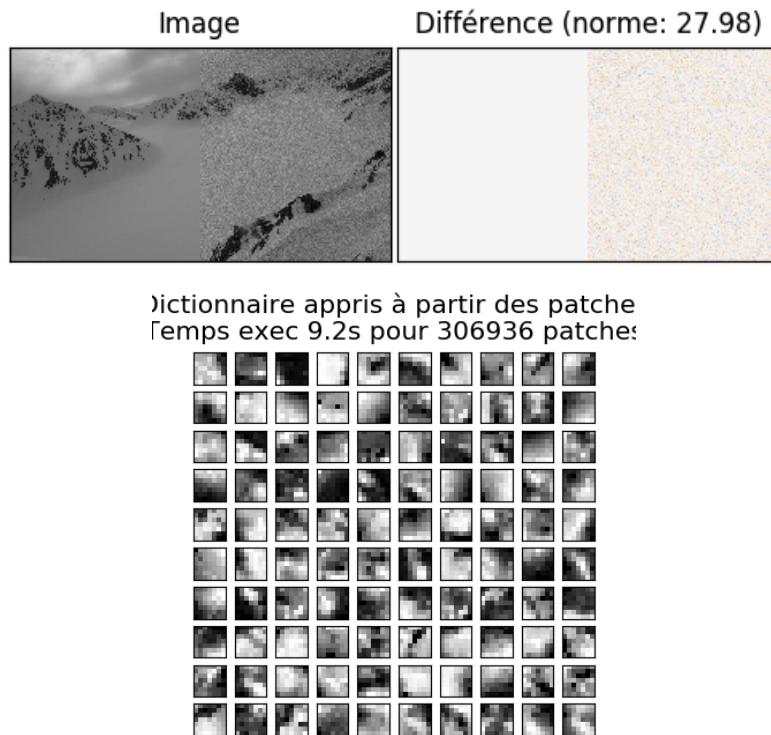
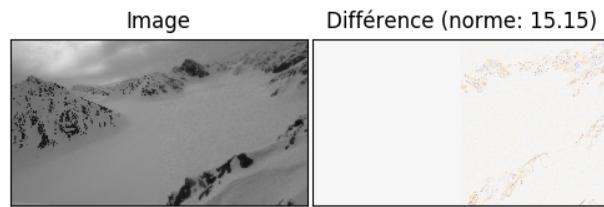


FIGURE 11 – Patchs et aperçu de la nouvelle image

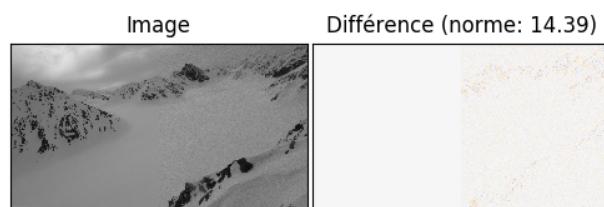
Nous avons donc fait évoluer le nombre de coefficients (atomes) non nuls dans la

méthode OMP. Le débruitage devient intéressant plus le nombre d'atomes non nuls augmente. Cependant, l'inverse se produit lorsqu'on a un nombre d'atomes trop grand.

Orthogonal Matching Pursuit
1 atome (Temps: 18.5s)



Orthogonal Matching Pursuit
5 atomes (Temps: 87.6s)



Orthogonal Matching Pursuit
10 atomes (Temps: 144.7s)

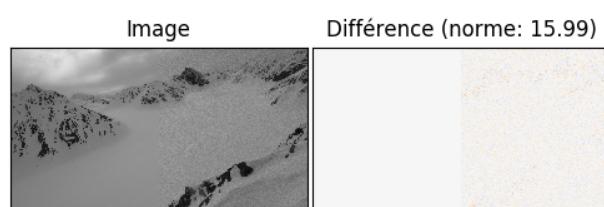


FIGURE 12 – Résultats pour le quatrième essai

Influence des paramètres

Nous avons donc pu voir l'influence de deux paramètres notamment :

- L, le nombre d'itérations dans notre algorithme K-SVD
- B, la taille des patchs

Pour L, il est clair qu'il a une influence directe sur le temps d'exécution du programme. On peut clairement voir qu'augmenter le nombre d'itérations améliore toutefois la qualité du débruitage puisque l'apprentissage du dictionnaire est amélioré à chaque itération. Par contre, si on choisit un nombre d'itérations trop grand, comme l'algorithme est susceptible de choisir un plus faible nombre d'atomes à chaque itération, on peut observer des grosses irrégularités (image floue, présence de blocs dans l'image, etc.). Dans notre cas, à partir de 10 ou 15 itérations semblent suffisant. On optera plutôt pour 10 pour éviter d'allonger la durée du programme et pour gagner en performance sur l'ordinateur.

Pour B, il y a également une incidence sur le temps d'exécution du programme. En effet, plus la taille est patchs est petite, plus le temps sera long. Il faut donc faire attention à ce paramètre par rapport au temps mais par rapport aussi à la taille de l'image en elle-même. Prendre de trop grands patchs sur une image de petite taille n'aura pas grand intérêt. Ici, nous avons fait varier B entre 4 et 16. Nous avons de meilleur résultat lorsque B est petit puisque notre image est petite et que plus de détails nous avons, plus le résultat reconstruit est bon. Cependant, sur une image en taille réelle, il faudrait un B proportionnel à la taille de l'image, donc un B nettement plus grand que 4, 8 ou 16 pour avoir de bons résultats. Il faudrait au moins envisager 32.

Un dernier paramètre intéressant est le nombre de patch utilisé, il serait intéressant de faire varier ce nombre. On sait qu'il aura une incidence, il faudra trouver un juste milieu entre un nombre de patchs trop petits pour reconstruire une image cohérente et un nombre de patchs trop grands qui permettrait de reconstruire une image trop à l'identique de l'image bruitée, par exemple avec la seconde méthode.

Notre resize des images et le fait que la résolution est donc plus faible pour pouvoir faire fonctionner nos programmes a donc un peu biaisé nos analyses et l'influence réelle des paramètres, il faudrait voir si nous tirons les mêmes conclusions sur les images en taille réelle.

Comparaison des résultats et méthodes

On obtient sensiblement les mêmes résultats sur les images les moins bruitées. Cependant, avec la première méthode, on est susceptible d'obtenir de meilleures résultats (surtout avec une image propre comme référence pour le dictionnaire) qu'avec la seconde méthode avec nos images les plus bruitées.

Au niveau de nos PSNR, nous trouvons sensiblement les mêmes degrés. Cependant, cela ne nous apprend juste la différence entre l'image sélectionnée et l'image reconstruite. C'est simplement un indicateur de différence et l'on voit que l'on évolue très peu en général.

5 Perspectives

Nous avons plusieurs axes d'amélioration dans le cadre de ce projet. Les premiers consistent à améliorer nos méthodes actuelles. Il faut que nous obtenions de meilleurs résultats globaux. Il faut pour cela passer des images en taille réelles et potentiellement des images en couleur. Il faut donc trouver un processus qui à la fois utilise les canaux RBG et également n'importe quel format d'image tout en restant performant en temps de calcul. Actuellement, nous utilisons des images en nuance de gris et pour la plupart du temps des images carrées et réduites.

Ensuite, il y a plusieurs autres possibilités. Il est possible de tester d'autres méthodes de débruitage pour avoir un aspect global et pouvoir comparer la méthode K-SVD avec d'autres méthodes afin de déterminer l'apport et l'intérêt de cette nouvelle méthode.

Nous pouvons également essayer de modifier la méthode de codage parcimonieux en utilisant une variante de l'OMP (Cholesky par exemple) ou complètement une autre méthode.

Il serait aussi intéressant d'utiliser une base d'images plus exploitables et classifiées par le degré de bruit pour pouvoir faire des analyses pertinentes. Nos fonctions marchent par exemple très bien sur des images de type 'lena' mais réellement moins bien sur les images de ce projet du fait notamment de la réduction et donc de la perte d'information sur les images.

Enfin, il peut être intéressant de se renseigner sur d'autres approches dans le domaine. Par exemple, l'inpainting qui a pour objectif de reconstruire des régions d'images où l'information est manquante ou fortement perturbée, toujours par la méthode de reconstruction avec K-SVD mais utilisée différemment.

6 Conclusion

Pour conclure, nous avons mis au point deux méthodes et des essais afin de débruiter une image initialement bruitée par divers éléments.
La première méthode était basée sur l'apprentissage d'un dictionnaire sur des images considérées propres, avec une sélection au préalable de patchs se chevauchant en partie, par l'algorithme K-SVD. Tandis que pour la seconde méthode, le dictionnaire est

directement appris directement sur l'image bruitée elle-même. Dans les deux, la reconstruction se fait par un méthode de débruitage utilisant un moyennage par codage parcimonieux OMP.

Dans la globalité, ces essais sont plutôt intéressants sur des images quelconques (léna par exemple) mais très décevants sur notre base d'images au vu des images reconstruites obtenues. Nous ne considérons pas cela comme un échec mais plutôt mais il reste un gros travail d'adaptation de nos algorithmes à tout type de format d'image.

Grâce à nos différents essais, nous avons pu faire le tour des quelques méthodes existantes pour le K-SVD et nous avons pu analyser l'influence de certains paramètres de la méthode dans nos résultats.

Bien que nos deux méthodes fonctionnent, nous sommes conscients des améliorations qu'il serait susceptible d'apporter afin d'obtenir de meilleures reconstructions.

Pour finir, nous pourrions avoir une approche différente du problème en utilisant des méthodes tierces de codage parcimonieux ou de débruitage général.

Références

- [1] Michael Elad and Michal Aharon. *Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries*, IEEE transactions on image processing, Vol. 15, No. 12, December 2006
- [2] Université Pierre et Marie Curie. *Application de l'apprentissage de dictionnaires parcimonieux en traitements d'images*, Master 2 Informatique - Spécialité IMA
- [3] Université Pierre et Marie Curie. *Apprentissage de dictionnaires image*, Master 2 Informatique - Spécialité IMA
- [4] R. Rubinstein, M. Zibulevsky, and M. Elad.. *Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit*. Technion, Computer Science Department - Technical Report, 2008.