Rapport de Projet

February 24, 2021

Contents

1 Introduction **1**

2 L'algorithme NL mean **1**
2.1 Le principe et les limites 1
2.2 Les solutions 2

3 Ma tentative **3**
3.1 Idée 3
3.2 Construire les arbres 3
3.2.1 Tentative PCA 3
3.2.2 Tentative "hybride" 6
3.2.3 Tentative aléatoire 8
3.2.4 Tentative du sandwitch au jambon 9
3.3 Résoudre les mauvaises ségmentations 10

4 Résultats **10**

5 Bibliographie **13**

1 Introduction

Les images sont souvent parasités par du bruit, qui est défini par n'ayant pas de signification dans le contexte d'intérprétation du signal. L'un de ces bruits les plus courants, le bruit gaussien, est présent sur les photographies avec une intensité dépendante des conditions de capture de l'image et de la qualité de l'appareil. Il est parfois intéressant de diminuer le bruit d'une image, afin de mieux percevoir l'information utile de l'image. Une approche classique pour débruiter une image est de supprimer les hautes fréquences en moyennant

les pixels entre eux, comme par exemple avec une convolution avec un noyau gaussien, ou encore de retirer directement ces hautes fréquences via une analyse spectrale de l'image, par FFT ou ondelettes. Cette approche retire le bruit de haute fréquence, mais aussi les détails abrupts de l'image, comme les bordures qui se retrouvent floutés car un changement net d'intensité est une variation de haute fréquence. D'autres approches sont alors explorées, afin de nullifier le bruit de haute fréquence sans pour autant perdre les fortes variations d'intensité de l'image.

2 L'algorithme NL mean

2.1 Le principe et les limites

Le principe est toujours de moyenner des pixels entre eux, mais l'idée est de moyenner les pixels venants de zones de l'images qui se ressemblent.

On découpe l'image en patches (par défaut carrés de rayon fixe) centrés sur chaque pixel de l'image.

Pour chaque pixel, on fait la somme avec tous les autres pixels de l'image pondérés par une gaussienne de la distance entre les patches. C'est à dire, pour une image I de n pixels découpés en un ensemble de patches P, on a l'image I':

$$I'(i) = \frac{\sum_j I(j) e^{-\frac{\|P(i)-P(j)\|^2}{2\sigma^2}}}{\sum_j e^{-\frac{\|P(i)-P(j)\|^2}{2\sigma^2}}}$$

On va donc moyenner les pixels de l'image dont les voisinages sont similaires, ce qui en pratique permet de conserver les fortes variations d'intensités de l'image sans pour autant retirer trop d'efficacité au débruitage.

Une chose saute aux yeux: la complexité en $O(n^2)$ de l'algorithme. L'enjeu va donc être de restreindre le nombre de sommes sans pour autant perdre l'efficacité de l'algorithme.

2.2 Les solutions

Une première solution et de rendre cet algorithme de nouveau local, en limitant les patches contribuant à un pixel à un voisinage fixe autour de lui. On perds alors la puissance de la non-localité de l'algorithme de base, car plus on a d'occurences du même motif, meilleur sera le résultat. Néanmoins, les images naturelles ne répartis pas uniformément les motifs, et donc la plupart du temps, l'image se traîte assez bien avec un voisinage fixe de patches. On

pourrait aussi trouver des moyens plus intelligent de choisir les pixels à sommer, ou alors encore utiliser une Image intégrale et une FFT afin de calculer plus rapidement les distances [1]

Une autre chose faite en complément est de projeter les patches de tailles $(2r+1)^2$ sur un sous-espace de dimension moindre, afin de baisser les coûts de calcul. En TP et dans la littérature, cela se fait par analyse spectrale de la matrice de covariance des patches, qui permet d'obtenir une base dans la participation de chaque direction au signal est connue, et on peut donc choisir de garder uniquement un nombre fixe de plus importantes dirrections, ou encore d'en garder autant qu'il le fait pour récupérer une fraction de l'intensité totale (ce que je fais pour éviter de m'adapter aux images).

3 Ma tentative

3.1 Idée

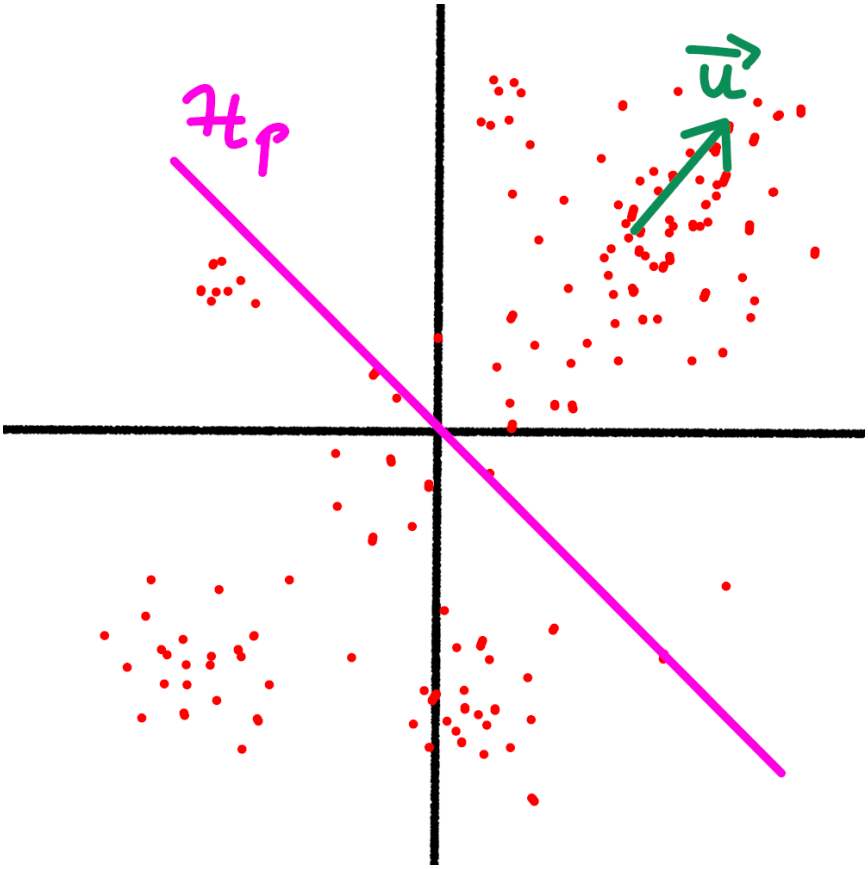
Une piste pour améliorer cet algorithme serait de trouver un meilleur moyen de choisir les pixels à moyenner. Un voisinage géométrique fixe est souvent suffisant, mais on observe souvent qu'une partie non négligeable des pixels du voisinage ne contribuent pas à la somme car leurs voisinages sont trop différents. L'idée est de ne plus choisir les patches à comparer avec un voisinage fixe, mais de segmenter l'espace des patches de façon à rassembler les patches proches et permettre d'avoir un assortiment plus pertinnent de patches pour chaque pixel. En effet, les patches ne sont pas répartis uniformément dans leur espace, et séparer les patches en pôles de densité forte semble être assez intéressant. On peut espérer avoir un meilleur résultat ou une baisse de la complexité de calcul. Enfin franchement, étant donné le temps accordé à ce projet, un résultat presque aussi bien que l'algorithme de base seraît déjà un bon début.

Évidement, je ne travaille pas sur les patches entiers, mais sur leur projections sur un sous-espace comme détaillé plus haut. Je détaille par la suite diverses approches de segmentation en arbre binaire de l'espace.

3.2 Construire les arbres

3.2.1 Tentative PCA

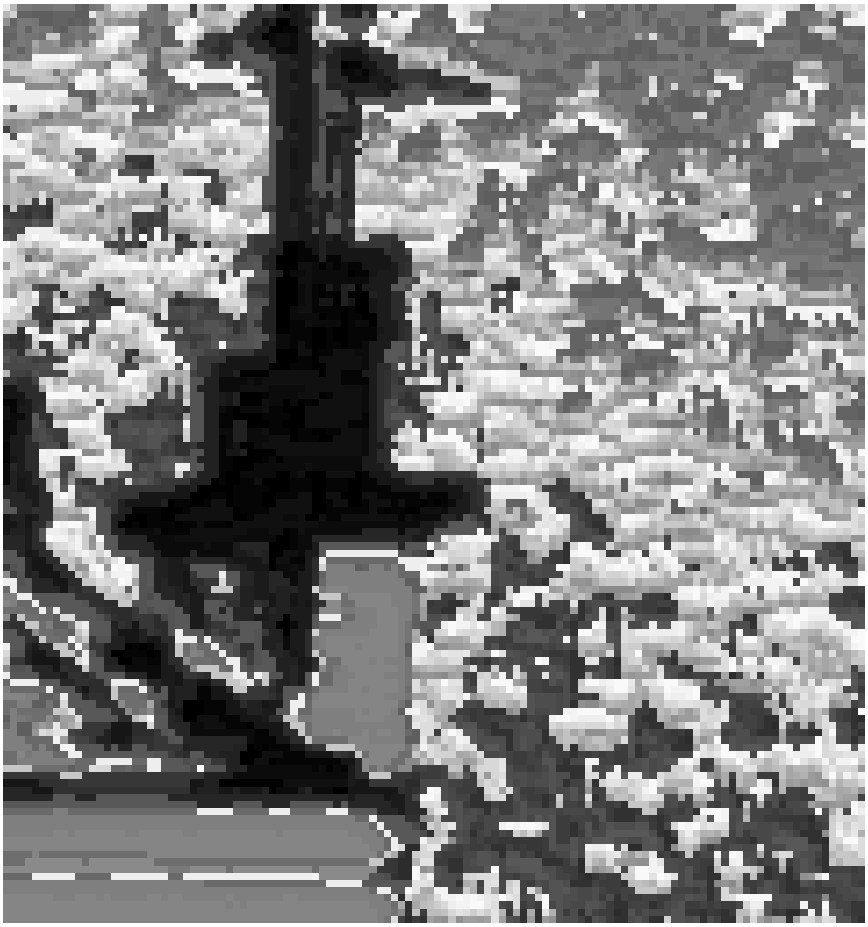
Parce que j'utilise une PCA pour réduire la dimension des patches, j'ai émis l'hypothèse qu'une séparation de points par un hyperplan normal au vecteur le plus important de la répartition des patches dans leur espace et passant par le barycentre puisse être suffisant. Voici l'idée illustrée en 2D:



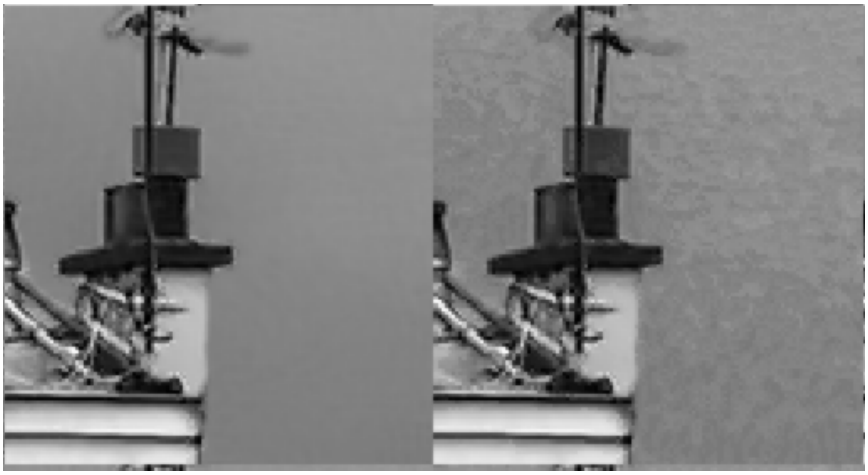
On sépare les patches en 2 sous-ensemble, puis on les resépare récursivement jusqu'à atteindre un certain nombre de points.

En pratique, les images comportent beaucoup de patches proches du barycentre (notamment ceux des zones d'applats), et des séparations abusives arrivent assez rapidement.

Voici à quoi ressemblent les feuilles de l'arbre obtenues, représentées en niveau de gris:

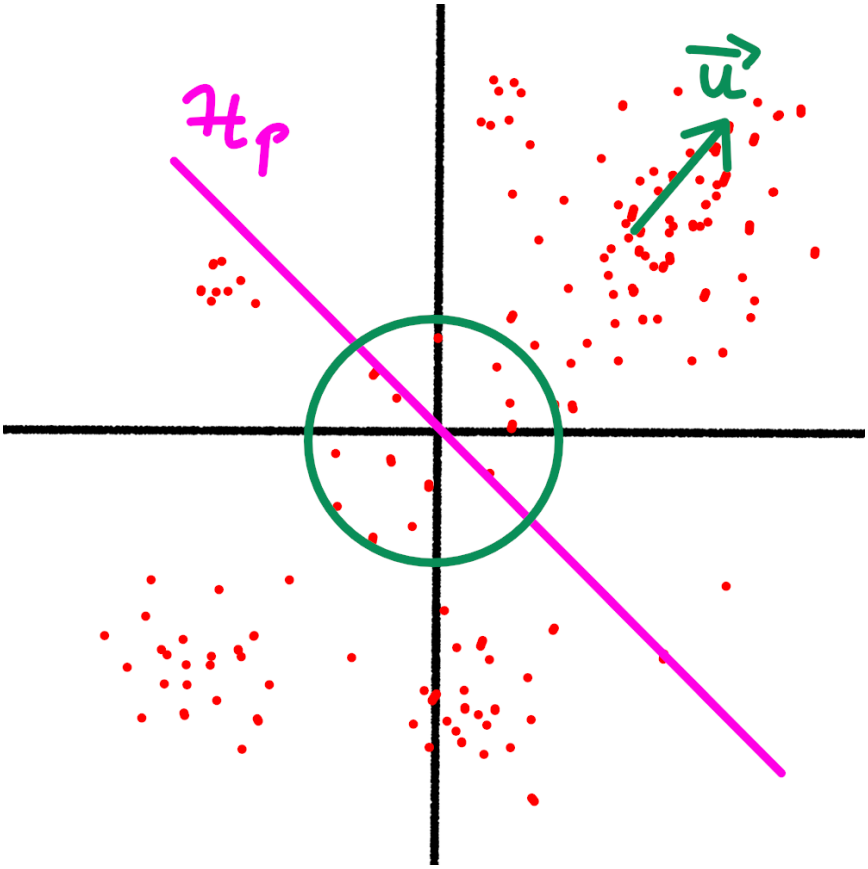


On observe comme prévus beaucoup de séparations abusive dans les zones de faible variation: (à gauche, l'image traitée par l'algorithme de base, à droite, par l'algorithme pca)

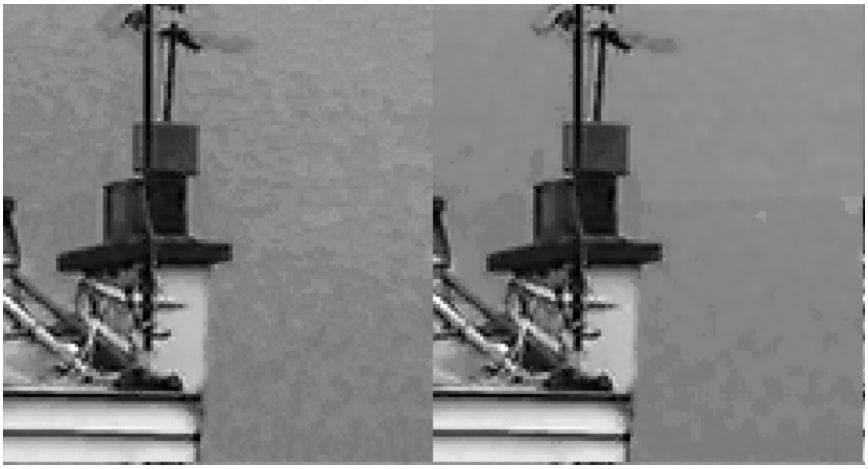


3.2.2 Tentative "hybride"

Les points centraux se faisant trop séparer à mon goût, j'ai tenté une modification: à chaque récursion, j'extrait d'abord les points centraux avec une balle centrée au barycentre, puis je scinde le reste avec une coupe pca. Pour bien choisir le rayon, je fait un histogramme de la norme des points recentrés, et je cherche le premier minimum local (signe de baisse de densité à la frontière et donc qu'on intersecte probablement peu de pôles de forte densité).



Comparé à une approche pca pour les mêmes paramètres, on obtient une nette amélioration:

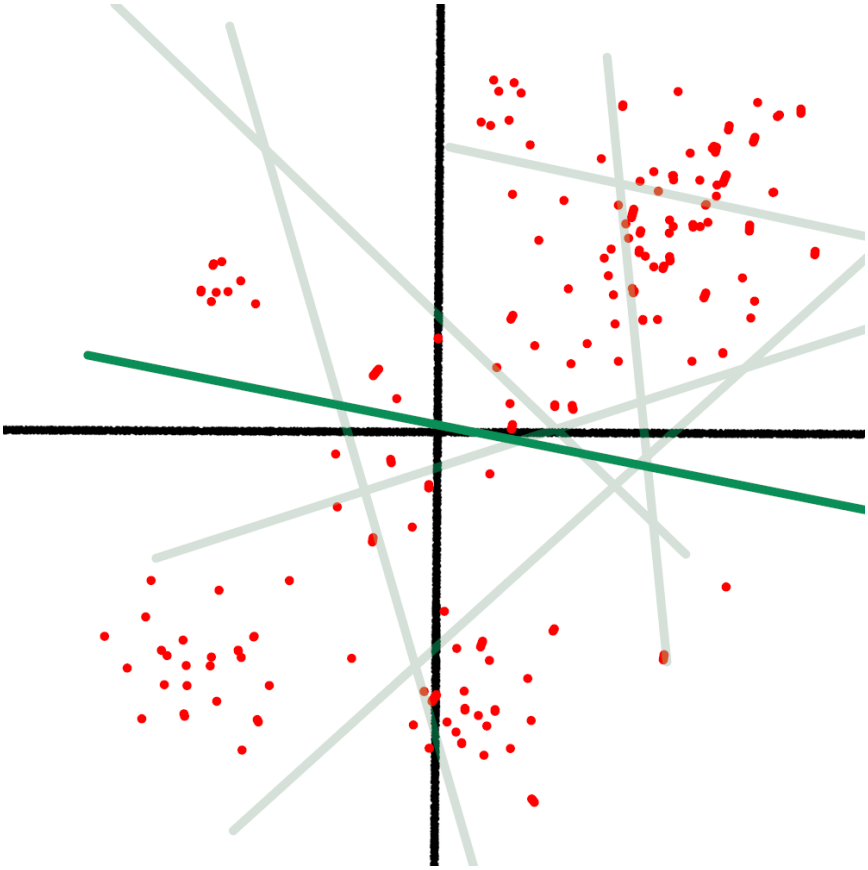


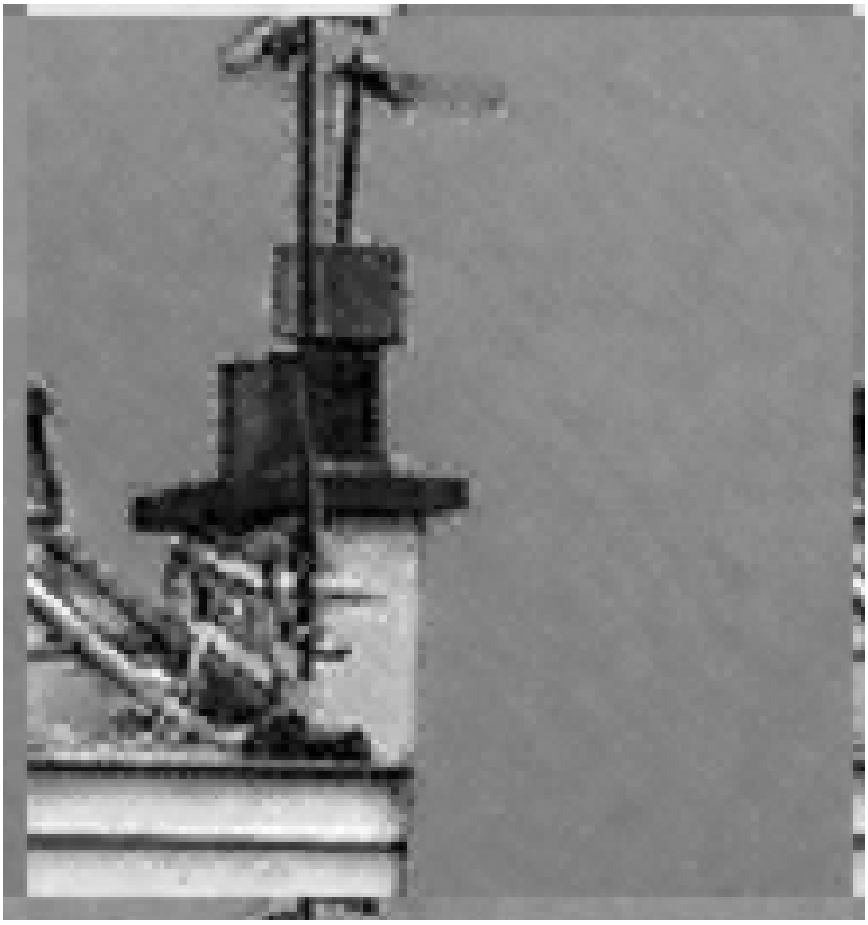
3.2.3 Tentative aléatoire

Une autre méthode que j'ai essayé est de générer des hyperplans aléatoirement à partir des points de l'ensemble à subdiviser, puis de garder le meilleur, c'est à dire celui qui minimise

$$|n_{right} - n_{left}|$$

.





3.2.4 Tentative du sandwitch au jambon

J'ai décidé d'appliquer une méthode que j'ai déjà vue dans le contexte de diminution de complexité d'un ensemble de points avec conservation de l'information de densité qui utilisait le théorème du sandwich au Jambon. Ce théorème stipule que pour tout espace de dimension n, tout ensemble de points colorié en n couleurs différentes peut être scindé équitablement couleur par couleur par un unique hyperplan. Cette propritété est utilisée par Matheny, Michael & Phillips et Jeff. pour construire un arbre de partitionnement de l'espace à moindre coût (en fesant une première séparation triviale, puis en reséparant les deux moitiées avec un seul hyperplan), afin d'obtenir un arbre qui ressemble à ce que je cherche. En effet, ils ont pour objectif d'obtenir un arbre dont chaque feuille pourrait être représentée par

un seul de leur points, ce qui devraît être adapté à mon approche.

Le résultat est identique à la tentative aléatoire, mais je pense que c'est dû à de mauvais réglages de ma part, je n'ai pas eu le temps de me pencher plus avant sur la question.

3.3 Résoudre les mauvaises ségmentations

Le point commun qu'ont toutes ces méthodes est la sursegmentation de certains ensemble de patches, menant à des discontunitées là où il ne devrait pas y en avoir.

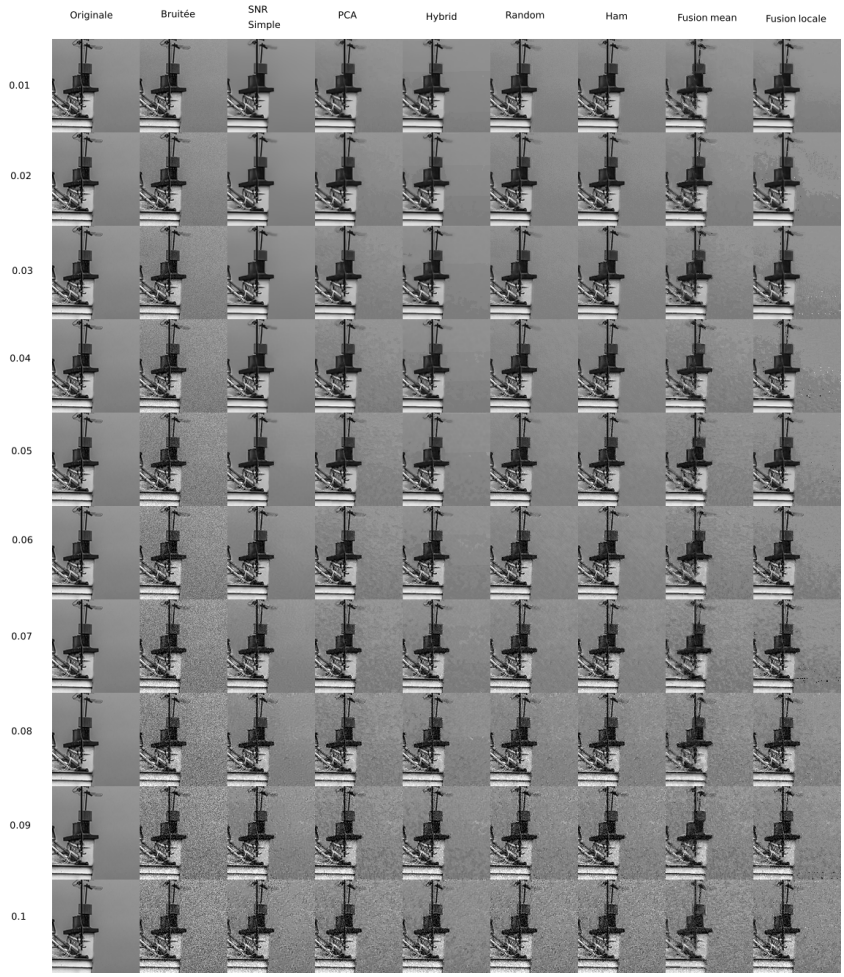
J'ai donc pensé à fusionner à postériori les ensembles qui n'auraient pas dû être segmentés. J'ai tenté deux approches:

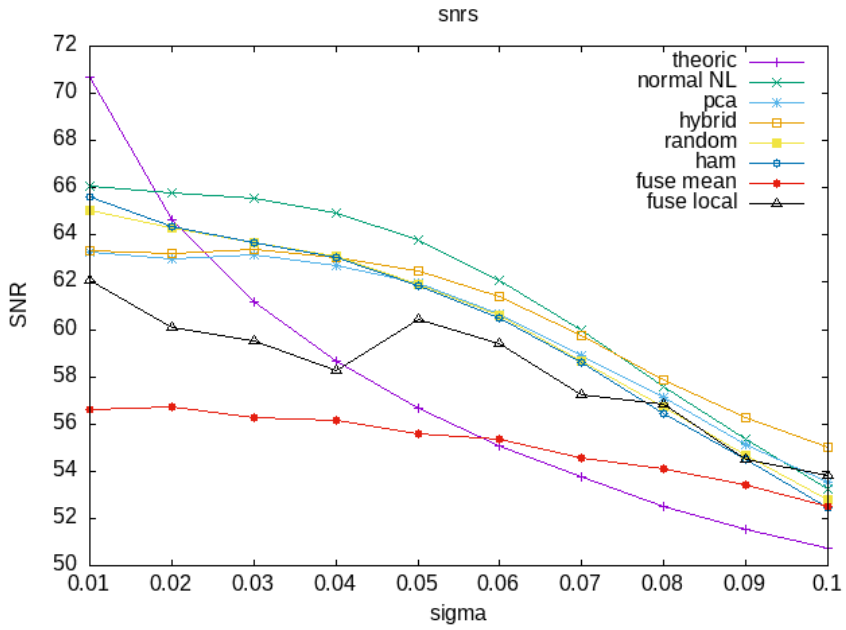
- comparer des moyennes approximatives des différents ensemble de points
- ne plus travailler au seins d'un sous-ensemble de points, mais aussi inclure ceux qui sont voisins géographiquement et ensuite faire une fusion comme dans l'approche précédente

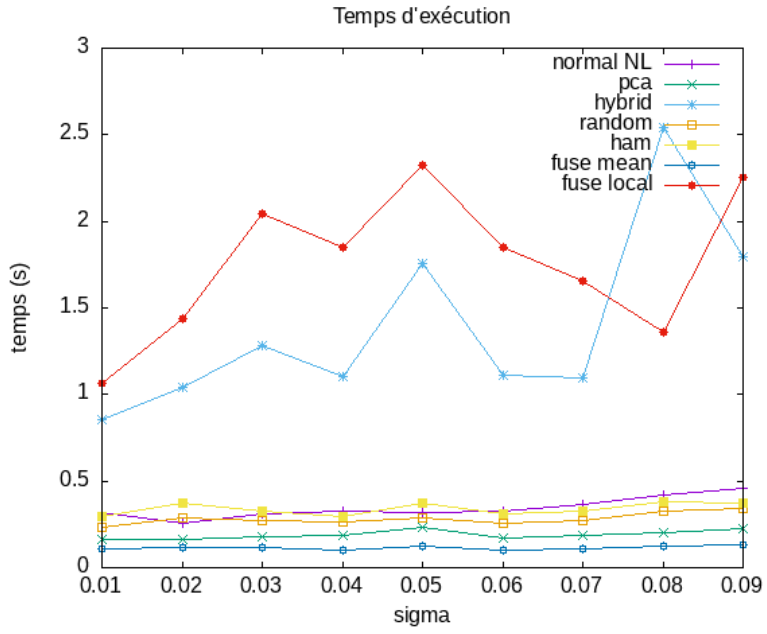
Je n'ai pas eu le temps de beaucoup explorer cette voie, et je ne l'es ai testés que pour l'approche PCA.

4 Résultats

Je n'ai pas eu énormément le temps de jouer sur les paramètres, mais voici une comparaison de toutes les approches citées avec un bruit gaussien d'intensité croissante.


Originale
Bruitée
SNR Simple
PCA
Hybrid
Random
Ham
Fusion mean
Fusion locale
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1





J'ai exclus le sigma=0.01 du tracé du temps d'exécution car la méthode hybride y explose, ce qui rend le reste du graphique peu intéressant.

5 Bibliographie

1. J. Wang, Y. Guo, Y. Ying, Y. Liu and Q. Peng, "Fast Non-Local Algorithm for Image Denoising," 2006 International Conference on Image Processing, Atlanta, GA, USA, 2006, pp. 1429-1432, doi: 10.1109/ICIP.2006.312698.

2. Matheny, Michael & Phillips, Jeff. (2018). Practical Low-Dimensional Halfspace Range Space Sampling.

3. `https://www.college-de-france.fr/media/pierre-louis-lions/UPL67973_Jean_Michel_MorelTransparents.pdf`