TP 3: Restauration d'images

Im4, Restauration

2017-2018 Semestre 2

Remarque: les en-têtes de fonctions sont indicatives. Libres à vous de les modifier, d'utiliser d'autres types pour la représentation des images etc.

Exercice 1

Ecrire une fonction qui génère différents types de bruit sur une image en entrée `Im`:
`void Bruit(Image *Im, Image *Imb, type_bruit t)`
La variable `t` codera pour:
a) un bruit gaussien de variance σ,
b) un bruit uniforme d'amplitude a
c) un bruit "poivre et sel" où un pourcentage p de pixels est affecté par un défaut systématique qui rend ces pixels noirs ou blancs.
On fera attention de bien recadrer les niveaux de gris de l'image à afficher quand c'est nécessaire.

Exercice 2

a) Ecrire une fonction qui calcule le SNR et le PSNR entre une image de référence I et une image bruitée I_b de même taille $M \times N$:
`void SignalBruit(Image *Im,Image * Im2,double *snr)`
On redonne les formules:

$$SNR = 10 \log_{10} \left(\frac{\sum_{i,j} I(i,j)^2}{\sum_{i,j} (I(i,j) - I_b(i,j))^2} \right)$$

$$PSNR = 10 \log_{10} \left(\frac{MN \max_{i,j} I(i,j)^2}{\sum_{i,j} (I(i,j) - I_b(i,j))^2} \right)$$

b) Utilisez les méthodes et filtres que vous avez déjà implémenté pour débruiter des images: équation de la chaleur, filtre gaussien, diffusion anisotrope, filtres médian et de Nagao. Vous utiliserez les mesures précédentes pour comparer l'image débruitée avec l'image originale et conclure quant à la meilleure méthode.

Exercice 3

Ecrire une fonction
`void Inpainting(Image *Im1, Image *masque, Image *Im2, double dt, int n)`
qui réalise l'inpainting d'une image `Im1` en comblant les zones du `masque` comme vu en cours.
`n` correspond au nombre d'itérations, `dt` est le pas de temps pour la discrétisation temporelle.