Débruitage de Tikhonov

```
typedef struct
    {
    int width, height, size;
    double *data;
      } ImageD;
```

Fonction tikhonov

Entrée :

- *deux images de type « Image » nommées dep et res.*
- *deux réels de type « double » nommés dt et lam.*

Sortie :

- *l'image res*

Déclarez deux entiers i et j.

Déclarez deux entiers maxi et mini.

Déclarez un réel dist, initialisez cette variable à 100 (par exemple).

Créez trois images U1 U2 et lap de type «ImageD» et de mêmes dimensions que l'image dep.

Créer une image temp de type « Image ».

// MINIMISATION DE TIKHONOV

Recopiez l'image dep dans l'image U1 (ne pas oubliez les conversions short vers double !)

Tant que (dist > 0.05) :

    laplacien(U1,lap)

    pour 0 <= i < largeur :

        pour 0 <= j < hauteur :

            U2[i,j] = U1[i,j] + dt * [(double) dep[i,j] – U1[i,j] + lam * lap[i,j])

        fin pour

    fin pour

    dist = **distance**(U1,U2)

    recopier l'image U2 dans l'image U1

fin tant que

// MINIMISATION DE TIKHONOV (FIN)

Recopiez l'image U2 dans l'image temp (ne pas oubliez les conversions double vers short).

Repérez les valeurs maximum et minimum de l'image dep, et mémorisez-les dans les variables maxi et mini.

Réalisez une expansion dynamique de l'image temp, de sorte que ses valeurs maximum et minimum soit égales à maxi et mini. Stockez le résultat dans l'image res.

Libérez la mémoire...

Fonction laplacien

entrée : une image im1 de type « ImageD »

sortie : une image im2 de type « ImageD »

// ValmirrorD réalise la même tâche que la fonction ValMirror habituelle, mais prend en entrée une image de type « ImageD ».

Déclarez deux entiers i et j.

Pour 0 <= i < largeur :

pour 0 <= j < hauteur :

im2[i,j] = ValMirrorD(im1, i-1, j)+ValMirrorD(im1, i+1, j)+ValMirrorD(im1, i, j-1)+ValMirrorD(im1, i, j+1) - 4.*ValMirrorD(im1, i, j)

fin pour

fin pour

Fonction distance

entrée : deux images im1 et im2 de type « ImageD »

sortie : un réel dist de type « double »

// Une manière comme une autre de mesurer la ressemblance entre deux images : somme des différences des pixels au carré

Déclarez deux entiers i et j.

Déclarez un réel somme, initialisez cette variable à zéro.

Pour 0 <= i < largeur :

pour 0 <= j < hauteur :

somme += $(im1[i,j] - im2[i,j])^2$

fin pour

dist = $\sqrt{somme}$