Traitement d'images par approches variationnelles et équations aux dérivées partielles

Jean-François Aujol

IMB, Université Bordeaux
351 Cours de la libération, 33405 Talence Cedex, FRANCE

Email : jaujol@math.u-bordeaux.fr
http ://www.math.u-bordeaux.fr/∼jaujol/

Le corps du document contient de nombreuses autres expériences en relation avec le cours. Vous êtes invités à les tester aussi pour votre culture scientifique.

Table des matières

1. Introduction

Dans tout ce cours, nous ne considèrerons que des images en niveau de gris. La partie pratique sera effectuée sous Matlab.

1.1 Du continu au discret

Soit Ω un ouvert borné de $\mathbb{R}^2$ (typiquement, nous considèrerons le cas où Ω est un rectangle). On considère une image comme une fonction $u : \Omega \to \mathbb{R}$ ($u : \Omega \to \mathbb{R}^3$ dans le cas d'une image couleur).

La fonction u est définie sur Ω. Mais en traitement d'images, nous n'avons accès qu'à des valeurs discrètes de cette fonction u.

1.2 Matlab et les images

Pour Matlab, une image de taille (m, n) est une matrice I de taille (m, n), et la valeur de $I(i, j)$ correspond à la valeur du niveau de gris de l'image au pixel (i, j).

Matlab est capable de lire a peu près tous les formats standards d'images.

On trouve des images au format *Matlab* dans le répertoire :

```
\Matlab\toolbox\matlab\demos
```

et des images au format *tiff* dans :

```
\Matlab\toolbox\images\imdemos
```

1.2.1 Exemples : visualisation d'une image

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Chargement d'une image en Matlab:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load gatlin2;
% -> L'image est chargee dans la variable X
%Autres images:
%load clown; load gatlin; load mandrill;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Visualisation:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
imagesc(X);
colormap gray;
%pour voir l'image en niveaux de gris
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Pour ouvrir une deuxieme figure:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(2);
```

FIGURE 1 – Image gatlin2

```
colormap gray;
XX=imread('cameraman.tif');
imshow(XX);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

1.2.2 Remarques :

Matlab est un logiciel facile à utiliser. Il ne faut surtout pas oublier de regarder l'aide (commandes : *help* et *helpwin*). Par-contre, les fonctions Matlab peuvent très rapidement demander du temps et de la place mémoire.

Matlab a été développé au départ pour résoudre des problèmes matricielles. Il est fortement conseillé d'écrire les algorithmes sous formes matricielles plutôt qu'en utilisant des boucles *for*.

Pour la lisibilité des programmes, il est recommandé d'utiliser des noms de variables explicites. De plus, même si ce n'est pas indispensable, il est recommandé d'initialiser les variables (fonctions *zeros*, *ones*).

Enfin, on rappelle que pour insérer des commentaires dans un fichier Matlab, il suffit d'utiliser l'instruction %. Il est beaucoup plus facile de corriger un programme bien commenté et avec des noms de variables explicites.

1.2.3 Un premier exemple de fonction Matlab : bruitage d'une image

```
function out = bruitage_gaussien(I,s)
%bruitage gaussien (d'ecart-type s) d'une image I
% bruitage_gaussien(I,s)
```

FIGURE 2 – Image gatlin2 bruitée par un bruit additif gaussien ($\sigma = 5$)

```
[m,n]=size(I);
J=zeros(m,n);

%creation du bruit gaussien
J=s*randn(m,n);

%bruitage
out=I+J;
```

La Figure 2 donne un exemple de bruitage de la Figure 1 par un bruit blanc gaussien d'écart-type $\sigma = 5$.

1.3 Conditions de bord

Comme nous travaillons sur un ouvert Ω borné, nous avons besoin de condition de bord. Les conditions qui interviennent naturellement en traitement d'images sont les conditions de Neumann :

$$\frac{\partial u}{\partial N} = 0 \text{ sur } \partial\Omega \tag{1.1}$$

Pour imposer une telle condition, il suffit de prolonger l'image discrète I par réflexion par rapport à ses bords.

1.3.1 Exemple :

La procédure suivante prolonge une image par réflexion.

```
function B=bord(A,d)

%Extension d'une image A par reflexion (pour d pixels)
%utile pour les conditions de Neumann
%syntaxe: bord(A,d)
[m,n]=size(A);
% On cree la matrice B de la bonne taille
M=m+2*d;
N=n+2*d;
B=zeros(M,N);
B(d+1:M-d,d+1:N-d)=A;

%On complete par reflexion
for i=1:m
    for j=1:d
        B(i+d,j)=A(i,d-j+1);
    end;
    for j=N-d+1:N
        B(i+d,j)=A(i,n+N-j-d);
    end;
end;


for j=1:N
    for i=1:d
        B(i,j)=B(2*d-i+1,j);
    end;
    for i=M-d+1:M
        B(i,j)=B(2*M-i-2*d,j);
    end;
end;
```

La Figure 3 donne un exemple d'extension de la Figure 1 par réflexions.

1.4 Consistence, stabilité, convergence

Nous reprenons ici la présentation de [5]. En traitement d'images, la méthode la plus couramment employée pour discrétiser les EDP est celle des différences finis (cela provient de la structure des images digitales qui sont formées par un ensemble de pixels uniformément distribués).

Nous présentons quelques notions d'analyse numérique dans le cas 1-D (par souci de clarté). Considérons le problème :

$$\begin{cases} \mathcal{L}v = F & t > 0,\ x \in \mathbb{R} \\ v(0,x) = f(x) & x \in \mathbb{R} \end{cases} \tag{1.2}$$

où $\mathcal{L}$ est un opérateur linéaire.

FIGURE 3 – Image gatlin2 étendue par réflexions

On veut résoudre (1.2) numériquement. On commence par discrétiser l'espace spatial en plaçant une grille sur cette espace. Par souci de simplicité, on utilisera une grille uniforme de pas h. On discrétise de la même manière le temps, avec un pas δt.

Résoudre numériquement le problème signifie trouver une fonction discrète u définie aux points $(n\delta t, nh)$ (on note u_i^n la valeur de u en ces points). La fonction u est obtenue comme la solution d'une équation discrète (qui est une approximation de (1.2)) :

$$\begin{cases} L_i^n u_i^n = G_i^n & i = -\infty, \dots, +\infty \\ u_i^0 = f(ih) \end{cases} \tag{1.3}$$

Il est important de voir que l'équation discrétisée remplace l'équation originale par une nouvelle équation, et qu'une solution exacte du problème discrétisé donne une solution approchée de l'EDP originale, puisque nous avons introduit une erreur dûe à la discrétisation.

Une première notion importante est celle de *convergence*. On note $u^n = (\dots, u_{-1}^n, u_0^n, u_1^n, \dots)$.

Définition 1.1. Le schéma (1.3) approchant l'EDP (1.2) est un schéma convergent au temps t si lorsque $(n+1)\delta t \to t$, on a :

$$|u^{n+1} - v^{n+1}|_* \to 0 \tag{1.4}$$

quand $h \to 0$ et $\delta t \to 0$, et où $|.|_*$ est une norme à préciser

Une autre notion importante est celle d'ordre de convergence :

Définition 1.2. Le schéma (1.3) approchant l'EDP (1.2) est un schéma convergent d'ordre (p, q) si et seulement si pour tout t, lorsque $(n+1)\delta t \to t$, on a :

$$|u^{n+1} - v^{n+1}|_* = O((h)^p) + O((\delta t)^q) \tag{1.5}$$

quand $h \to 0$ et $\delta t \to 0$.

Théorème 1.1. *[Lax] Un shéma numérique aux différences finies à deux niveau consistent (pour un problème linéaire, à valeurs initiales, bien posé) est convergent si et seulement si il est stable.*

Remarques :

1. Un shéma numérique à deux niveau est un schéma dans lequel seuls deux niveaux de temps différents sont présents dans l'équation (typiquement $H(u^{n+1}, u^n) = 0$)
2. L'erreur de consistence concerne l'erreur introduite par la discrétisation de l'équation (elle doit tendre vers 0 lorsque δt et h tendent vers 0). Ecrivons le schéma numérique à deux niveau (1.3) sous la forme :

$$u^{n+1} = Qu^n + \delta t\, G^n \tag{1.6}$$

Définition 1.3. Le schéma (1.3) est consistent avec l'EDP (1.2) pour la norme $|.|_*$ si la solution v de l'EDP vérifie :

$$v^{n+1} = Qv^n + \delta t\, G^n + \delta \tau^n \tag{1.7}$$

avec $\tau^n \to 0$ lorsque h, $\delta t \to 0$.

On construit des schémas numériques consistent en utilisant les formules de Taylor.

3. L'idée intuitive de la stabilité est qu'une petite erreur sur la donnée initiale doit entrainer une petite erreur sur la solution (c'est la notion analogue à celle de *bien posé* pour les EDP).

Définition 1.4. Le schéma à deux niveau

$$\begin{cases} u^{n+1} = Qu^n + \delta t\, G^n \ , \ n \geq 0 \\ u^0 \text{ donnée} \end{cases} \tag{1.8}$$

est dit stable pour la norme $|.|_*$ s'il existe des constantes strictement positives h_0 et δt_0, ainsi que des constantes positives K et β telles que :

$$\left|u^{n+1}\right|_* \leq K e^{\beta t} \left|u^0\right|_* \tag{1.9}$$

pour $0 \leq t = (n+1)\delta t$, $0 < h \leq h_0$, et $0 < \delta t \leq \delta t_0$.

1.5 Différences finies

On note u l'image considérée, et $u_{i,j}$ sa valeur au pixel (i, j). Le pas d'espace h est fixé égal à 1.

Il existe la fonction *gradient* dans Matlab. Cependant, il sera utile de coder plusieurs types d'approximations du gradient. L'approche par différences finies est la plus simple. Elle se révèle bien adaptée pour les équations paraboliques.

1.5.1 Différences finies centrées :

$$\delta_x u_{i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2} \tag{1.10}$$

$$\delta_y u_{i,j} = \frac{u_{i,j+1} - u_{i,j-1}}{2} \tag{1.11}$$

1.5.2 Différences finies décentrées à droite :

$$\delta_x^+ u_{i,j} = u_{i+1,j} - u_{i,j} \tag{1.12}$$

$$\delta_y^+ u_{i,j} = u_{i,j+1} - u_{i,j} \tag{1.13}$$

1.5.3 Différences finies décentrées à gauche :

$$\delta_x^- u_{i,j} = u_{i,j} - u_{i-1,j} \tag{1.14}$$

$$\delta_y^- u_{i,j} = u_{i,j} - u_{i,j-1} \tag{1.15}$$

1.6 Schéma ENO

Les shémas ENO (essentially non-oscillatory) sont bien adaptés pour la discrétisation des équations hyperboliques. On donne ici deux approximations possibles de $|\nabla u|$ au pixel (i, j).

$$\nabla^+ u_{i,j} = \sqrt{\max(\delta_x^- u_{i,j}, 0)^2 + \min(\delta_x^+ u_{i,j}, 0)^2 + \max(\delta_y^- u_{i,j}, 0)^2 + \min(\delta_y^+ u_{i,j}, 0)^2} \tag{1.16}$$

et

$$\nabla^- u_{i,j} = \sqrt{\max(\delta_x^+ u_{i,j}, 0)^2 + \min(\delta_x^- u_{i,j}, 0)^2 + \max(\delta_y^+ u_{i,j}, 0)^2 + \min(\delta_y^- u_{i,j}, 0)^2} \tag{1.17}$$

La Figure 4 montre le gadient de l'image de la Figure 1.

2. Equation de la chaleur : filtrage linéaire et non linéaire

Pour toute cette section, nous renvoyons le lecteur à [5, 32, 26] ainsi qu'aux références données par ces ouvrages.

On pourra se servir des discrétisations données a la section 4.2.

2.1 Equation de la chaleur

La première EDP à avoir été utilisée en traitement d'images est certainement l'équation de la chaleur. Il s'agit d'une EDP parabolique.

$$\begin{cases} \frac{\partial u}{\partial t} - \Delta u(t,x) = 0 \text{ , si } t \geq 0 \text{ et } x \in \mathbb{R}^2 \\ u(0,x) = u_0(x) S \end{cases} \tag{2.1}$$

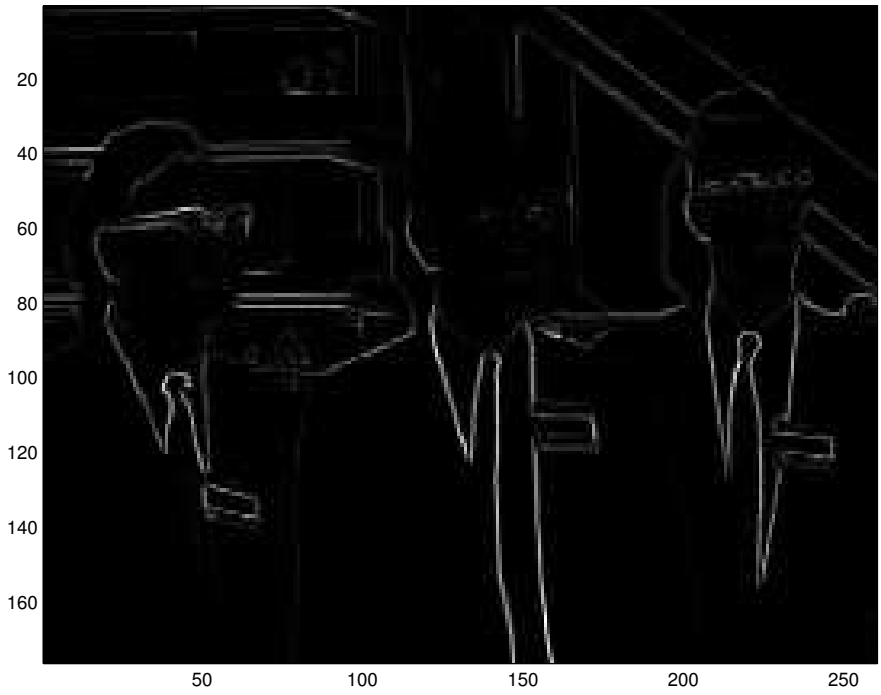

FIGURE 4 – Gradient de gatlin2

L'introduction de cette équation provient de la remarque suivante : si u_0 la donnée initiale est suffisamment régulière, la solution explicite de (2.1) est donnée par :

$$u(t,x) = \int_{\mathbb{R}^2} G_{\sqrt{2}t}(x-y)u_0(y)\,dy = \left(G_{\sqrt{2}t} * u_0\right)(x) \tag{2.2}$$

où G_σ désigne le noyau Gaussien en dimension 2 :

$$G_\sigma(x) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{|x|^2}{2\sigma^2}\right) \tag{2.3}$$

L'opération de convolution par un noyau positif est une opération de base en traitement d'image. Cela correspond à un filtrage passe-bas.

2.1.1 Schéma numérique

Notation : On note δt le pas de discrétisation en temps, et $h = 1$ le pas de discrétisation en espace (on prend le même pas selon les deux axes). $u^n_{i,j}$ désigne la valeur de $u(n\delta t, hi, hj)$. On va utiliser un schéma d'Euler explicite à l'ordre 1 en temps : on approxime donc le terme $\frac{\partial u}{\partial t}(n\delta t, i, j)$ par $\frac{u^{n+1}_{i,j} - u^n_{i,j}}{\delta t}$. Pour obtenir une discrétisation du Laplacien, on discrétise les différentes dérivées par différences finies décentrées à gauche et à droite. Vérifier qu'on obtient l'expression suivante :

$$\Delta u^n_{i,j} = u^n_{i+1,j} + u^n_{i-1,j} + u^n_{i,j+1} + u^n_{i,j-1} - 4u^n_{i,j} \tag{2.4}$$

Remarque : Toute ces discrétisation ne tiennent pas compte de la nature 2D et des propriétés des opérateurs. Par exemple, le Laplacien est un opérateur invariant par rotation. Le

FIGURE 5 – Image gatlin2 (filtrée par l'EDP de la chaleur)

schéma discret utilisé ici ne l'est clairement pas. Pour résoudre ce problème, on peut modifier la discrétisation du Laplacien en utilisant tous les points dans un voisinage 3*3 du pixel considéré. On obtient ainsi des shémas numériques plus robustes. Cependant, dans ce cours, nous ne nous intéresserons pas à ces discrétisations. Pour plus de détails, voir [5].

2.2 Exemples :

Suivant le temps d'évolution, on obtient une version plus ou moins lissée de l'image de départ (cf Figures 1 et 5).

EDP de la chaleur : Programmer l'équation de la chaleur. La tester sur des images non bruitées puis bruitées. Vérifier que les zone homogènes sont bien restaurées, mais par-contre que les bords sont érodés. La Figure 8 montre un exemple de restauration de l'image Gatlin2 bruitée (cf Figure 2) par l'EDP de la chaleur.

Convolution avec une Gaussienne : Pour vérifier (2.2), effectuer la convolution d'une image par un noyau Gaussien (fonction *filter2*)

Application : Sur une image lissée, on peut plus facilement essayer de détecter les bords d'une image. On va utiliser ici le détecteur de Hildrett-Marr : on cherche les zéros du Laplacien d'une image u. Si en un point x, Δu change de signe et si ∇u est non nul, alors on considère qu'en x l'image u possède un bord. Programmer un tel détecteur de bord. Comparer avec la fonction *edge* de Matlab.

Avec cet algorithme, calculer les bords d'une image lissée par l'EDP de la chaleur (comparer les résultats obtenus suivant le temps d'évolution) (cf Figures 6 et 7).

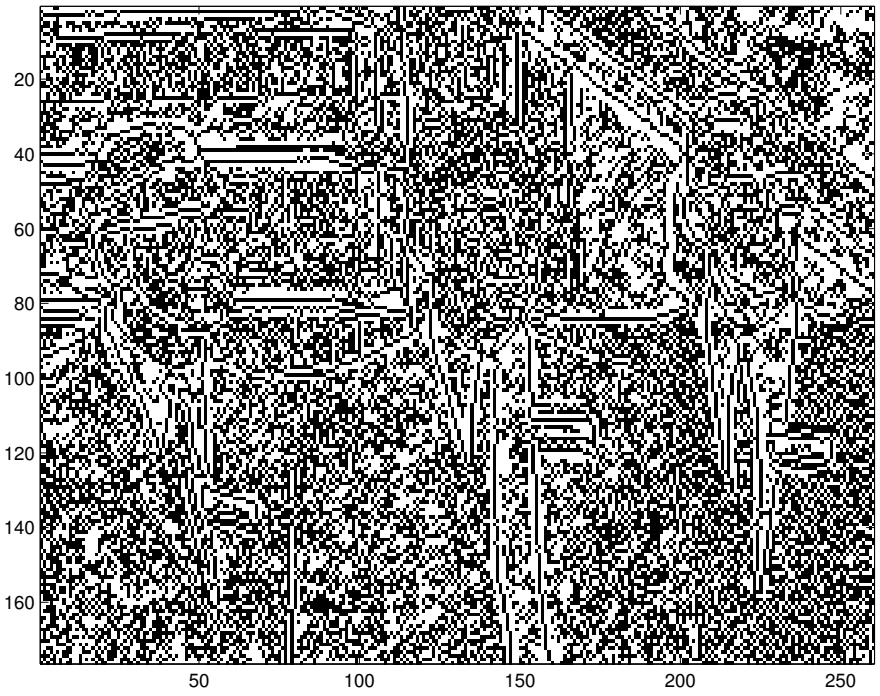

FIGURE 6 – Bords de l'image Gatlin2 donnés par le détecteur de Hildrett-Marr

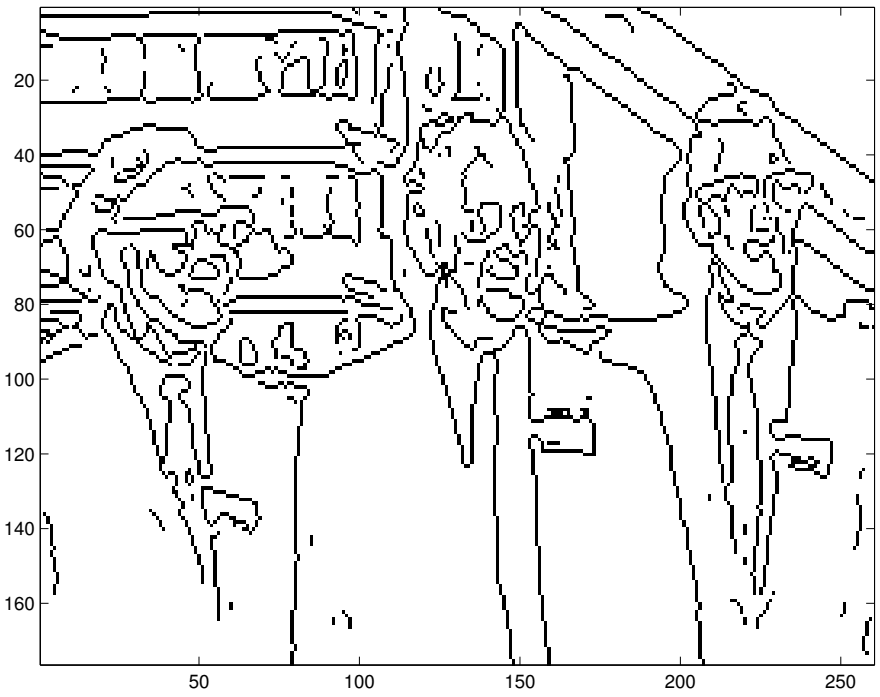

FIGURE 7 – Bords de l'image Gatlin2 donnés par le détecteur de Hildrett-Marr (après filtrage par l'EDP de la chaleur)

FIGURE 8 – Restauration de gatlin2 (EDP chaleur)

2.3 Perona-Malik

Pour améliorer les résultats obtenus par l'EDP de la chaleur, Perona et Malik ont proposé de modifier l'équation en y intégrant le processus de détection des bords :

$$\begin{cases} \frac{\partial u}{\partial t} = \text{div } (c(|\nabla u|)\nabla u) \text{ dans } \Omega \times (0,T) \\ \frac{\partial u}{\partial N} = 0 \text{ sur } \partial\Omega \times (0,T) \\ u(0,x) = u_0(x) \text{ dans } \Omega \end{cases} \tag{2.5}$$

où c est une fonction décroissante de $\mathbb{R}_+$ dans $\mathbb{R}_+$.

Remarque : Si $c = 1$, on retrouve l'équation de la chaleur.

On impose souvent que $c(0) = 1$ et $\lim_{t\to+\infty} c(t) = 0$. Ansi, dans les régions de faible gradient, l'équation agit essentiellement comme l'EDP de la chaleur, et dans les régions de fort gradient, la régularisation est stoppée (ce qui permet de préserver les bords).

2.3.1 Schéma numérique

Sur le même principe que l'EDP de la chaleur, écrire le schéma numérique pour (2.5). Tester le nouvel algorithme avec les fonctions $c(t) = \frac{1}{\sqrt{1+(t/\alpha)^2}}$ et $c(t) = \frac{1}{1+(t/\alpha)^2}$. Vérifier les propriétés annoncées (comparer avec l'équation de la chaleur).

On rappelle que si l'opérateur gradient ∇ est discrétisé par différence finie à droite, alors une discrétisation possible de la divergence est donnée par :

$$(\text{div } (p))_{i,j} = p^1_{i,j} - p^1_{i-1,j} + p^2_{i,j} - p^2_{i,j-1} \tag{2.6}$$

avec $p = (p^1, p^2)$.

FIGURE 9 – Image gatlin2 filtrée par Perona-Malik

FIGURE 10 – Restauration de gatlin2 (version bruitée) par Perona-Malik

La Figure 9 montre un exemple de filtrage de l'image Gatlin2 par Perona-Malik, et la Figure 10 donne un exemple de restauration de l'image Gatlin2 bruitée (cf Figure 2) par Perona-Malik.

2.4 Amélioration : convolution du gradient par une gaussienne

L'un des principales défauts du modèle précédent concerne la restauration des images bruitées. En effet, le bruitage d'une image entraine de fortes oscillations de son gradient. Et le modèle va donc garder le bruit (en le considérant comme des bords). Une approche simple pour limiter ce problème consiste à utiliser le modèle suivant :

$$\begin{cases} \frac{\partial u}{\partial t} = \text{div } (c(|G_\sigma| * \nabla u|)\nabla u) \text{ dans } \Omega \times (0,T) \\ \frac{\partial u}{\partial N} = 0 \text{ sur } \partial\Omega \times (0,T) \\ u(0,x) = u_0(x) \text{ dans } \Omega \end{cases} \tag{2.7}$$

Tester cette nouvelle approche, et la comparer avec le modèle précédent sur des images bruitées.

Cependant, ce nouveau modèle possède quand même une grande part des défauts du modèle précédent.

3. La théorie Scale-Space d'Alvarez-Guichard-Morel-Lions

Pour toute cette section, nous renvoyons le lecteur à [5, 32, 26, 13], ainsi qu'aux références données par ces ouvrages.

3.1 Théorie Scale-Space

3.1.1 Rappels sur les solutions de viscosité

La notion de solution de viscosité s'applique aux équations définies sur un ouvert Ω de $\mathbb{R}^N$de la forme suivante :

$$F(x,u,Du,D^2u) = 0 \tag{3.1}$$

et elle s'étend très facilement aux équations d'évolution :

$$\frac{\partial u}{\partial t} + F(x,u,Du,D^2u) = 0 \tag{3.2}$$

Dans le cas de (3.2), les notations Du et D^2u correspondent respectivement aux dérivées premières et secondes en espace.

La définition des solutions de viscosité s'introduit naturellement à partir du principe du maximum. Elle permet de reporter la régularité de la solution sur une fonction test (on retrouve la même idée dans la théorie des distributions). Pour de plus amples détails sur cette théorie, nous renvoyons le lecteur à [12, 20, 21].

Soit f une fonction localement bornée. On notera respectivement f_* et f^* les enveloppes semi-continue inférieurement (sci) et semi-continues supérieurement (scs) de f :

$$f_*(x) = \liminf_{y\to x} f(y) \tag{3.3}$$

$$f^*(x) = \limsup_{y\to x} f(y) \tag{3.4}$$

Remarque : Nous ne donnons les définitions des solutions de viscosité que dans le cas des équations stationnaires (du type (3.1)), mais la généralisation aux équations d'évolution (du type (3.2)) est immédiate.

Définition 3.1. Le hamiltonien F est supposé défini en tout point et localement borné.

(i) Une fonction u localement bornée, scs sur Ω, est une sous-solution de viscosité de (3.1) si et seulement si :
$\forall \Phi \in C^2(\Omega)$, si $x_0 \in \Omega$ est un point de maximum local de $u - \Phi$ on a :

$$F_*(x_0, u(x_0), D\Phi(x_0), D^2\Phi(x_0)) \leq 0 \tag{3.5}$$

(ii) Une fonction u localement bornée, sci sur Ω, est une sur-solution de viscosité de (3.1) si et seulement si :
$\forall \Phi \in C^2(\Omega)$, si $x_0 \in \Omega$ est un point de minimum local de $u - \Phi$ on a :

$$F^*(x_0, u(x_0), D\Phi(x_0), D^2\Phi(x_0)) \geq 0 \tag{3.6}$$

(iii) Nous appellerons solution de viscosité de (3.1) toute fonction continue satisfaisant (3.5) et (3.6).

3.1.2 Scale-Space

On définit une analyse multi-échelle comme étant une famille d'opérateurs $\{T_t\}_{t\geq 0}$, qui appliquée à l'image originale $u_0(x)$ conduit à la suite d'images : $u(t,x) = (T_t u_0)(x)$. On donne une liste d'axiomes qu'on souhaiterait être vérifiée par la famille $\{T_t\}_{t\geq 0}$.

(H**1**) *Récursivité :* $T_0(u) = u$ et $T_s o T_t(u) = T_{s+t}(u)$

(H**2**) *Régularité :* $|T_t(u + hv) - (T_t(u) + hv)|_{L^\infty} \leq cht$

(H**3**) *Localité :* $(T_t(u) - T_t(v))(x) = o(t)$ quand $t \to 0^+$.

(H**4**) *Principe de comparaison :* $T_t(u) \leq T_t(v)$ si $u \leq v$.

(H**5**) *Invariance par translation de niveau de gris :* $T_t(0) = 0$, et $T_t(u+c) = T_t(u) + c$ pour toute constante c.

(H**6**) *Invariance par translation :* $T_t(\tau_h.u) = \tau_h(T_t u)$ où $(\tau_h.u)(x) = u(x+h)$.

Théorème 3.1. *On suppose que les hypothèses précédentes sont vérifiées. Si $u_0 \in C_b(\mathbb{R}^2)$, alors il existe une fonction continue F telle que $u(t,x) = (T_t u_0)(x)$ soit l'unique solution de viscosité de :*

$$\begin{cases} \frac{\partial u}{\partial t} = F(\nabla u, \nabla^2 u) \\ u(0,x) = u_0(x) \end{cases} \tag{3.7}$$

Théorème 3.2. *Si on suppose que T_t satisfait (**H** 1)–(**H** 6), ainsi que l'hypothèse :*

(H **7**) *Invariance par isométrie :* $T_t(R.u)(x) = R.(T_t u)(x)$ pour toute transformation orthogonale R, où $(R.u)(x) = u(Rx)$.

Si on suppose de plus que $u \mapsto T_t u$ est linéaire, alors $u(t,x) = (T_t u_0)(x)$ est la solution de l'équation de la chaleur

$$\frac{\partial u}{\partial t} = c\Delta u \tag{3.8}$$

où c est une constante strictement positive.

Théorème 3.3. *Si on suppose que T_t satisfait (**H** 1)–(**H** 6), ainsi que les hypothèses :*

*(H***8)** *Invariance du niveau de gris : $T_t(\phi(u)) = \phi(T_t(u))$ pour toute fonction croissante ϕ.*

(H **9)** *Invariance par changement d'échelle : $\forall \lambda,\ t > 0$, il existe $t'(t,\lambda) > 0$ tel que $H_\lambda.(T_{t'}u) = T_t(H_\lambda.u)$, où $(H_\lambda.u)(x) = u(\lambda x)$.*

(H **10)** *Invariance par projections :* Pour tout $A : \mathbb{R}^2 \to \mathbb{R}^2$ linéaire, et pour tout $t > 0$, il existe $T'(t,A) > 0$ tel que $A.(T_{t'}u) = T_t(A.u)$.

Alors $u(t,x) = (T_t u_0)(x)$ est la solution de :

$$\begin{cases} \frac{\partial u}{\partial t} = |\nabla u|(t\, curv(u))^{1/3} \\ u(0,x) = u_0(x) \end{cases} \tag{3.9}$$

où

$$curv(u) = \frac{u_{xx}u_y^2 + u_{yy}u_x^2 - 2u_{xy}u_x u_y}{|\nabla u|^3} = \text{div}\left(\frac{\nabla u}{|\nabla u|}\right) \tag{3.10}$$

Remarques :

1. Si dans le Théorème 3.3, on remplace l'hypothèse (**H** 10) par (**H** 7), on obtient l'EDP :

$$\begin{cases} \frac{\partial u}{\partial t} = |\nabla u|\beta(t\, curv(u)) \\ u(0,x) = u_0(x) \end{cases} \tag{3.11}$$

   où β est une fonction continue croissante.

2. Il existe des liens très étroits entre les EDP présentées ci-dessus et les opérateurs morphologiques (i.e. monotone, et invariant par translation ou changement de contraste). En itérant de tels opérateurs, on retrouve ces EDP.

3.2 Restauration d'image par mouvement par courbure moyenne

Il s'agit de l'équation :

$$\begin{cases} \frac{\partial u}{\partial t} = |\nabla u|\text{div}\left(\frac{\nabla u}{|\nabla u|}\right) \\ u(0,x) = u_0(x) \end{cases} \tag{3.12}$$

On peut généraliser la méthode en considérant comme dans l'approche de Perona-Malik l'équation :

$$\begin{cases} \frac{\partial u}{\partial t} = c\left(|\nabla G_\sigma * u|\right)|\nabla u|\text{div}\left(\frac{\nabla u}{|\nabla u|}\right) \\ u(0,x) = u_0(x) \end{cases} \tag{3.13}$$

L'interprétation à la main de cette équation est la suivante :

1. Le terme $|\nabla u|\text{div}\left(\frac{\nabla u}{|\nabla u|}\right)$ est un terme de diffusion dégénéré : il diffuse u dans la direction orthogonale à ∇u (mais pas dans la direction de ∇u). Le but de ce terme est de lisser u de part et d'autre des bords, tout en préservant justement les bords (un bord est défini comme une ligne le long de laquelle le gradient est "grand").
2. Le terme $c\left(|\nabla G_\sigma * u|\right)|$ est un terme d'accentuation des bords. Si $|\nabla u|$ est petit au voisinage d'un point x, alors ce point est considéré comme appartenant à une zone homogène, et la diffusion est donc forte. Si $|\nabla u|$ est grand au voisinage d'un point x, alors ce point est considéré comme appartenant à un bord, et la diffusion est faible.

Comme pour le modèle de Perona Malik, cette approche repose sur le choix de deux paramètres :

1. La fonction de contraste c, qui décide si un détail est suffisamment important pour être gardé.
2. Un paramètre d'échelle, donné par la variance σ du noyau gaussien, qui fixe la taille minimale des détails conservés.

3.2.1 Schéma numérique

L'équation (3.12) est une équation parabolique et présente des effets de diffusion (tout comme l'équation de la chaleur). L'utilisation des différences finies classiques est donc appropriée. On utilise un schéma d'Euler explicite à l'ordre 1 en temps.

$$u_{i,j}^{n+1} = u_{i,j}^n + \delta t\sqrt{(\delta_x u_{i,j}^n)^2 + (\delta_y u_{i,j}^n)^2}K_{i,j}^n \tag{3.14}$$

où $\delta_x u_{i,j}^n = (u_{i+1,j}^n - u_{i-1,j}^n)/2$, et $K_{i,j}^n$ est l'approximation par différence finie centrée de la courbure : $K = \text{div}\left(\frac{\nabla u}{|\nabla u|}\right)$. On peut utiliser une approximation plus précise de la courbure :

$$K_{i,j} = \left(\frac{u_x}{|\nabla u|}\right)_{i+1/2,j} - \left(\frac{u_x}{|\nabla u|}\right)_{i-1/2,j} + \left(\frac{u_x}{|\nabla u|}\right)_{i,j+1/2} - \left(\frac{u_x}{|\nabla u|}\right)_{i,j-1/2} \tag{3.15}$$

où :

$$\begin{aligned}
\left(\frac{u_x}{|\nabla u|}\right)_{i+1/2,j} &= \frac{u_{i+1,j} - u_{i,j}}{\sqrt{(u_{i+1,j} - u_{i,j})^2 + 0.25\left[0.5\left(u_{i,j+1} - u_{i,j-1}\right) + 0.5\left(u_{i+1,j+1} - u_{i+1,j-1}\right)\right]^2}} \\
\left(\frac{u_x}{|\nabla u|}\right)_{i-1/2,j} &= \frac{u_{i,j} - u_{i-1,j}}{\sqrt{(u_{i,j} - u_{i-1,j})^2 + 0.25\left[0.5\left(u_{i-1,j+1} - u_{i-1,j-1}\right) + 0.5\left(u_{i,j+1} - u_{i,j-1}\right)\right]^2}} \\
\left(\frac{u_x}{|\nabla u|}\right)_{i,j+1/2} &= \frac{u_{i,j+1} - u_{i,j}}{\sqrt{(u_{i,j+1} - u_{i,j})^2 + 0.25\left[0.5\left(u_{i+1,j+1} - u_{i-1,j+1}\right) + 0.5\left(u_{i+1,j} - u_{i-1,j}\right)\right]^2}} \\
\left(\frac{u_x}{|\nabla u|}\right)_{i,j-1/2} &= \frac{u_{i,j} - u_{i,j-1}}{\sqrt{(u_{i,j} - u_{i,j-1})^2 + 0.25\left[0.5\left(u_{i+1,j} - u_{i-1,j}\right) + 0.5\left(u_{i+1,j-1} - u_{i-1,j-1}\right)\right]^2}}
\end{aligned}$$

Programmer la restauration d'image par mouvement par courbure moyenne. Attention : numériquement, dans l'expression de $K_{i,j}^n$, il peut apparaître des divisions par 0. Il convient de traiter ces cas à part dans le shéma numérique. Programmer ensuite l'équation (3.13). Les Figures 11 et 12 montrent des exemples obtenus avec cette équation.

FIGURE 11 – Image gatlin2 filtrée par mouvement par courbure moyenne

FIGURE 12 – Image gatlin2 bruitée, puis filtrée par mouvement par courbure moyenne

FIGURE 13 – Image gatlin2 filtrée par mouvement par AMSS

3.2.2 Expériences

Regarder les résultats obtenus sur différentes images (bruitées ou non). Comparer avec les résultats obtenus au chapitre précédent. Commentaires ?

3.3 Affine Morphological Scale-Space (AMSS)

Programmer l'équation (3.9).
Attention : pour Matlab, $(-27)^{1/3}$ n'est pas égal à -3.
Les Figures 13 et 14 montrent des exemples obtenus avec cette équation.

Remarque : Le schéma numérique proposé ici est trop simple pour vérifier toutes les hypothèses du Théorème 3.3. Des schémas numériques respectant toutes ces propriétés d'invariance ont été proposé par Cao [13] (mais nous ne les aborderons pas dans ce cours).

4. Restauration d'images

4.1 Introduction

Etant donnée une image originale u, on suppose qu'elle a été dégradée par un bruit additif v, et éventuellement par un opérateur R. Un tel opérateur est souvent modélisé par un produit de convolution, et n'est pas nécessairement inversible (et même lorsqu'il est inversible, son inverse est souvent numériquement difficile à calculer). A partir de l'image observée $f = Ru + v$ (qui est donc une version dégradée de l'image originale u), on cherche à reconstruire u. Si on suppose que le bruit additif v est Gaussien, la méthode du Maximum de vraisemblance nous conduit à

FIGURE 14 – Image gatlin2 bruitée, puis filtrée par mouvement par AMSS

chercher u comme solution du problème de minimisation

$$\inf_u \|f - Ru\|_2^2 \tag{4.1}$$

où $\|.\|_2$ désigne la norme dans L^2. Il s'agit d'un problème inverse mal posé. Pour le résoudre numériquement, on est amené à introduire un terme de régularisation, et à considérer le problème :

$$\inf_u \quad \underbrace{\|f - Ru\|_2^2}_{\text{attache aux données}} \quad + \quad \underbrace{L(u)}_{\text{régularisation}} \tag{4.2}$$

Dans toute la suite, nous ne considérerons que le cas où R est l'opérateur identité ($Ru = u$). Le terme de régularisation classique $L(u) = \|\nabla u\|_2^2$ (régularisation de Tychonov) n'est pas adapté au problème de restauration d'images : l'image restaurée u est alors beaucoup trop lissée (en particulier, les bords sont érodés). Une approche beaucoup plus efficace consiste à considérer la variation totale, c'est à dire à prendre $L(u) = \int |Du|$, et donc à regarder le problème [39] :

$$\inf_u \left(\|f - u\|_2^2 + \int |Du| \right) \tag{4.3}$$

4.2 Discrétisation

Une image numérique, ou discrète, est un vecteur à deux dimensions de taille $N \times N$. On note X l'espace euclidien $\mathbb{R}^{N \times N}$, et $Y = X \times X$. On munit l'espace X du produit scalaire :

$$(u, v)_X = \sum_{1 \leq i,j \leq N} u_{i,j} v_{i,j} \tag{4.4}$$

et de la norme :

$$\|u\|_X = \sqrt{(u,u)_X} \tag{4.5}$$

Pour définir une variation totale discrète, on introduit d'abord une version discrète de l'opérateur gradient. Si $u \in X$, le gradient ∇u est un vecteur de Y donné par :

$$(\nabla u)_{i,j} = ((\nabla u)^1_{i,j}, (\nabla u)^2_{i,j}) \tag{4.6}$$

avec

$$(\nabla u)^1_{i,j} = \begin{cases} u_{i+1,j} - u_{i,j} & \text{si } i < N \\ 0 & \text{si } i = N \end{cases} \tag{4.7}$$

et

$$(\nabla u)^2_{i,j} = \begin{cases} u_{i,j+1} - u_{i,j} & \text{si } j < N \\ 0 & \text{si } j = N \end{cases} \tag{4.8}$$

La variation totale discrète de u est alors donnée par :

$$J(u) = \sum_{1 \le i,j \le N} |(\nabla u)_{i,j}| \tag{4.9}$$

On introduit également une version discrète de l'opérateur divergence. On le définit par analogie avec le cadre continu en posant :

$$\text{div} = -\nabla^* \tag{4.10}$$

où ∇^* est l'opérateur adjoint de ∇ : i.e., pour tout $p \in Y$ et $u \in X$, $(-\text{div}\, p, u)_X = (p, \nabla u)_Y$. Il est aisé de vérifier que :

$$(\text{div}\,(p))_{i,j} = \begin{cases} p^1_{i,j} - p^1_{i-1,j} & \text{si } 1 < i < N \\ p^1_{i,j} & \text{si i=1} \\ -p^1_{i-1,j} & \text{si i=N} \end{cases} + \begin{cases} p^2_{i,j} - p^2_{i,j-1} & \text{si } 1 < j < N \\ p^2_{i,j} & \text{si j=1} \\ -p^2_{i,j-1} & \text{si j=N} \end{cases} \tag{4.11}$$

Nous utiliserons aussi une version discrète de l'opérateur Laplacien définie par :

$$\Delta u = \text{div}\, \nabla u \tag{4.12}$$

4.3 Régularisation de Tychonov

On considère le problème :

$$\inf_u \|f - u\|^2_X + 2\lambda \|\nabla u\|^2_X \tag{4.13}$$

4.3.1 Résolution par EDP

L'équation d'Euler associée est :

$$u - f - \lambda \Delta u = 0 \tag{4.14}$$

Et l'on est ainsi amené à résoudre l'équation :

$$\frac{\partial u}{\partial t} = \lambda \Delta u - u + f \tag{4.15}$$

La Figure 15 montre un exemple de restauration obtenue.

FIGURE 15 – Restauration (Tychonov) par EDP

4.3.2 Résolution en utilisant la transformée de Fourier

Une méthode plus rapide consiste à utiliser la transformée de Fourier discrète (TFD).

On rappelle que la TFD d'une image discrète $(f(m,n))$ $(0 \leq m \leq N-1$ et $0 \leq n \leq N-1)$ est donnée par $(0 \leq p \leq N-1$ et $0 \leq q \leq N-1)$:

$$\mathcal{F}(f)(p,q) = F(p,q) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} f(m,n)e^{-j(2\pi/N)pm}e^{-j(2\pi/N)qn} \tag{4.16}$$

et la transformée inverse est :

$$f(m,n) = \frac{1}{N^2}\sum_{p=0}^{N-1}\sum_{q=0}^{N-1} F(p,q)e^{j(2\pi/N)pm}e^{j(2\pi/N)qn} \tag{4.17}$$

On a de plus $\|\mathcal{F}(f)\|_X^2 = N^2\|f\|_X^2$ et $(\|\mathcal{F}(f), \|\mathcal{F}(g))_X = N^2(f,g)_X$.

Montrer que :

$$\|\mathcal{F}(\nabla f)\|^2 = \sum_{p,q} |\mathcal{F}(\nabla f)(p,q)|^2 = \sum_{p,q} 4\,|\mathcal{F}(f)(p,q)|^2 \left(\sin^2\frac{\pi p}{N} + \sin^2\frac{\pi q}{N}\right) \tag{4.18}$$

En utilisant l'identité de Parseval, en déduire que la solution u de (4.13) vérifie :

$$\mathcal{F}(u)(p,q) = \frac{\mathcal{F}(f)(p,q)}{1 + 8\lambda\left(\sin^2\frac{\pi p}{N} + \sin^2\frac{\pi q}{N}\right)} \tag{4.19}$$

Attention, avant d'effectuer la transformée de Fourier d'une image, il vaut mieux la prolonger par symmétrie.

La Figure 16 montre un exemple de restauration obtenue.

FIGURE 16 – Restauration (Tychonov) par TFD

4.4 Modèle de Rudin-Osher-Fatemi, et algorithme de projection de Chambolle

4.4.1 Descente de gradient

Dans [39], les auteurs décomposent une image f en une composante u appartenant à $BV(\Omega)$ et une composante v dans $L^2(\Omega)$. Dans ce modèle, v est supposée être le bruit. Dans une telle approche, l'énergie minimisée est la suivante :

$$\inf_{(u,v)/f=u+v} \left(J(u) + \frac{1}{2\lambda}\|v\|_X^2 \right) \tag{4.20}$$

où $J(u)$ désigne la variation totale de u. En pratique dans [39], la solution numérique est calculée à partir de l'équation d'Euler-Lagrange associée à (4.20) (après avoir remplacé le terme $\int |Du|$ par $\int |\nabla u|$). L'étude mathématique de (4.20) a été faite dans [16]. Pour améliorer les résultats de restauration d'images par variation totale, il est possible de rajouter d'autres contraintes dans le processus de minimisation [19].

Formellement, l'équation d'Euler-Lagrange associée à (4.20) est la suivante :

$$-\text{div}\left(\frac{\nabla u}{|\nabla u|}\right) + \frac{1}{\lambda}(u - f) = 0 \tag{4.21}$$

On résoud cette équation par descente de gradient :

$$\frac{\partial u}{\partial t} = \text{div}\left(\frac{\nabla u}{|\nabla u|}\right) - \frac{1}{\lambda}(u - f) \tag{4.22}$$

Programmer cette algorithme (attention aux divisions par zéro).

4.4.2 Algorithme de point fixe

Récemment, Antonin Chambolle a proposé dans [15] un algorithme de projection pour résoudre le problème de Rudin-Osher-Fatemi [39]. Plus précisément, le problème étudié dans [15] est le suivant :

$$\inf_{u\in X}\left(J(u)+\frac{1}{2\lambda}\|f-u\|_X^2\right) \tag{4.23}$$

Comme J est homogène de degré un (i.e. $J(\lambda u)=\lambda J(u)\ \forall u$ et $\lambda>0$), il est standard (cf [23]) que J^* la transformée de Legendre-Fenchel de J,

$$J^*(v)=\sup\left((u,v)_X-J(u)\right) \tag{4.24}$$

est la fonction indicatrice d'un ensemble convexe fermé K. Un calcul rapide montre que K s'identifie à l'ensemble :

$$K=\{\mathrm{div}\,(g)/g\in Y,|g_{i,j}|\leq 1\ \forall i,j\} \tag{4.25}$$

et

$$J^*(v)=\chi_K(v)=\begin{cases}0 & \text{si } v\in K\\ +\infty & \text{sinon}\end{cases} \tag{4.26}$$

Le résultat suivant est montré dans [15] :

Proposition 4.1. *La solution de (4.23) est donnée par :*

$$u=f-\lambda P_{\lambda K}(f) \tag{4.27}$$

où P est le projecteur orthogonal sur λK.

Preuve : Donnons ici quelques éléments pour montrer (4.27) (nous suivons la démarche de [15]).

On note ∂J le sous-différentiel de J (voir [37, 27]) défini par :

$$w\in\partial J(u)\Longleftrightarrow J(v)\geq J(u)+(w,v-u)_X\ \forall v\in X \tag{4.28}$$

Si u est un minimiseur de (4.23), alors nécessairement :

$$0\in\frac{u-f}{\lambda}+\partial J(u) \tag{4.29}$$

(4.29) est équivalent à :

$$\frac{f-u}{\lambda}\in\partial J(u) \tag{4.30}$$

On passe ensuite à une formulation duale en utilisant une identité classique en analyse convexe (cf corollaire 5.2 dans [23]), qui affirme que (4.30) est équivalente à $u\in\partial J^*\left(\frac{f-u}{\lambda}\right)$ soit encore

$$0\in\frac{f-u}{\lambda}-\frac{f}{\lambda}+\frac{1}{\lambda}\partial J^*\left(\frac{f-u}{\lambda}\right) \tag{4.31}$$

On en déduit que $w=\frac{f-u}{\lambda}$ est un minimiseur de $\frac{\left\|w-\frac{f}{\lambda}\right\|^2}{2}+\frac{1}{\lambda}J^*(w)$. Comme J^* est donnée par (4.26)), cela implique que $w=\frac{f-u}{\lambda}$ est la projection orthogonale de $\frac{f}{\lambda}$ sur le convexe K.

On remarque de plus que $P_K\left(\frac{f}{\lambda}\right)=\frac{1}{\lambda}P_{\lambda K}(f)$, et on en déduit (4.27).

■

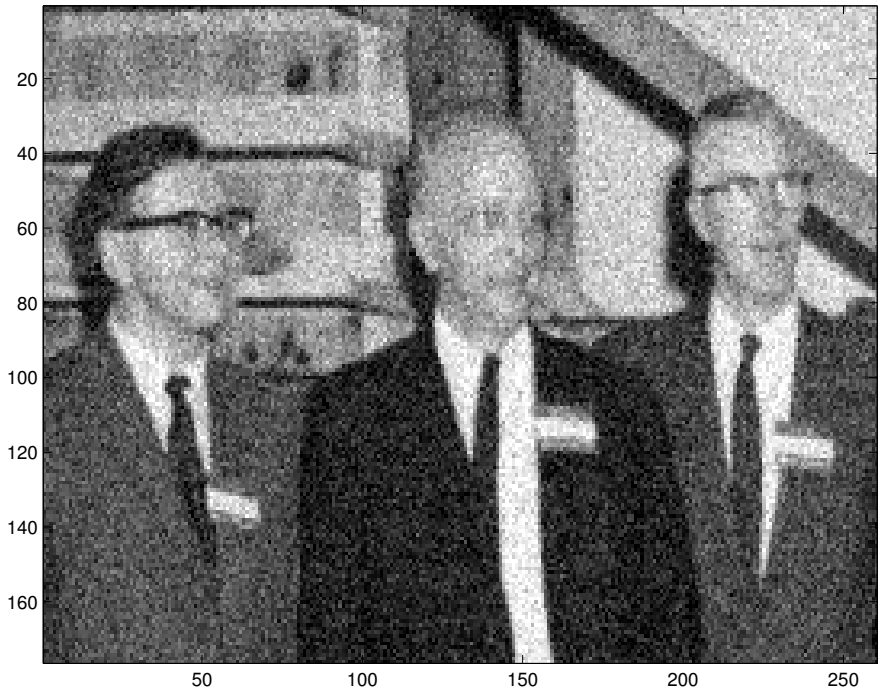

FIGURE 17 – Image bruitée à restaurer

Algorithme : [15] donne un algorithme pour calculer $P_{\lambda K}(f)$ qui peut s'écrire :

$$\min\left\{\|\lambda \mathrm{div}\,(p) - f\|_X^2 : p \ / \ |p_{i,j}| \leq 1 \ \forall i,j = 1,\ldots,N\right\} \tag{4.32}$$

(4.32) peut se résoudre par une méthode de point fixe.

$$p^0 = 0 \tag{4.33}$$

et

$$p_{i,j}^{n+1} = \frac{p_{i,j}^n + \tau(\nabla(\mathrm{div}\,(p^n) - f/\lambda))_{i,j}}{1 + \tau|(\nabla(\mathrm{div}\,(p^n) - f/\lambda))_{i,j}|} \tag{4.34}$$

Et [15] donne une condition suffisante pour que l'algorithme converge :

Théorème 4.1. *Supposons que le paramètre τ dans (4.34) vérifie $\tau \leq 1/8$. Alors $\lambda div(p^n)$ converge vers $P_{\lambda K}(f)$ quand $n \to +\infty$.*

La solution du problème (4.23) est donc donnée par :

$$u = f - \lambda \mathrm{div}\,(p^\infty) \tag{4.35}$$

où $p^\infty = \lim_{n\to+\infty} p^n$

Implémenter cet algorithme.

La Figure 18 est l'image restaurée obtenue avec cet algorithme à partir de l'image dégradée de la Figure 17.

FIGURE 18 – Image restaurée (ROF)

4.4.3 Gradient projeté

On peut aussi calculer la projection en utilisant un algorithme de gradient projeté.

$$\begin{cases} v^m = \frac{f}{\lambda} + \operatorname{div} p^m \\ p_{i,j}^{m+1} = \frac{p_{i,j}^m + \tau(\nabla v^m)_{i,j}}{\max\left\{1, |p_{i,j}^n + \tau(\nabla v^m)_{i,j}|\right\}} \end{cases} \tag{4.36}$$

et si $\tau < 1/4$, on montre que λv^m converge vers la solution de (4.20).

Programmer cette approche, et la comparer à l'algorithme de point fixe.

4.4.4 Acclération : algorithme de Nesterov

Une nouvelle classe d'algorithme du premier ordre particulièrement efficace est apparue récemment. L'algorithme suivant permet de calculer la solution de (4.20).

1. On fixe $k = 0$, $v_0 = 0$, $x_0 = 0$, $L = \lambda\|\operatorname{div}\|^2 = 8\lambda$.
2. On pose $k = k + 1$, et on calcule $\eta_k = \nabla\left(f - \lambda \operatorname{div}(x_k)\right)$.
3. On pose $y_k = P_K(x_k - \eta_k/L)$, avec $K = \{x \in L^2 \times L^2 \ / \ \|x\| \leq 1\}$.
4. Soit $v_k = v_{k-1} + \frac{k+1}{2}\eta_k$.
5. Soit $z_k = P_K(-v_k/L)$.
6. Soit $x_{k+1} = \frac{2}{k+3}z_k + \frac{k+1}{k+2}y_k$.
7. La sortie de l'algorithme est : $u = f - \lambda \operatorname{div}(y_{\lim})$.

Programmer cette méthode. Comparer sa vitesse de convergence avec les autres méthodes étudiés dans ce TP.

4.5 Déconvolution

Fonctionnelle approchée On considère le problème :

$$\inf_u \frac{1}{2\lambda}\|f - Au\|^2 + \int \sqrt{\epsilon^2 + |\nabla u|^2} \tag{4.37}$$

L'équation d'Euler associée est :

$$\frac{1}{\lambda}A^*(Au - f) - \text{div}\left(\frac{\nabla u}{\epsilon^2 + |\nabla u|^2}\right) = 0 \tag{4.38}$$

Un algorithme de descente de gradient est alors :

$$u_{n+1} = u_n - \delta t\left(\frac{1}{\lambda}A^*(Au - f) - \text{div}\left(\frac{\nabla u}{\epsilon^2 + |\nabla u|^2}\right)\right) \tag{4.39}$$

Tester cette approche (pour le applications numériques, on prendra pour A un noyau gaussien).

Forward-Backward Splitting

$$\frac{1}{2\mu}\|Au - f\|^2 + \int_\Omega |Du| \tag{4.40}$$

où A est un opérateur de flou (numériquement, on pourra prendre pour A un noyau gaussien).

On peut montrer que le schéma suivant permet de calculer la solution u avec $\nu\|A^*A\| \leq 1$:

$$\begin{cases} v_n = u_n + \nu A^*(f - Au_n) \\ u_{n+1} = \text{argmin}_u \left(\frac{1}{2\mu\nu}\|v_n - u\|^2 + \int |Du|\right) \end{cases} \tag{4.41}$$

Coder cette méthode. Comparer sa vitesse avec les méthodes pour la fonctionnelle approchée.

4.6 Seuillage en ondelettes

4.6.1 Espaces de Besov

On note $\{\psi_{j,k}\}$ une base d'ondelettes. Une fonction f dans $L^2(\mathbb{R}^2)$ s'écrit :

$$f = \sum_{j,k} c_{j,k}\psi_{j,k} \tag{4.42}$$

où les $c_{j,k}$ sont les coefficients en ondelettes de f, et on a : $\|f\|_{L^2(\mathbb{R}^2)} = \sum_{j,k} c_{j,k}^2$.

Une famille d'espaces adaptée aux ondelettes est celle des espaces de Besov $B^s_{p,q}$ (pour $0 < s < \infty$, $0 < p \leq \infty$ et $0 < q \leq \infty$) [31, 30, 18, 16]. $B^s_{p,q}$ correspond schématiquement aux fonctions possédant s "dérivées" dans $L^p(\mathbb{R}^2)$, le troisième paramètre q permettant d'ajuster avec précision la régularité.

Remarque : si $p = q = 2$, alors $B^s_{2,2}$ est l'espace de Sobolev $W^{s,2}$, et lorsque $s < 1$, $1 \leq p \leq \infty$, et $q = \infty$, alors $B^s_{p,\infty}$ est l'espace de Lipschitz $Lip(s, L^p(\mathbb{R}^2))$.

On peut donner une définition intrinsèque des espaces de Besov $B^s_{p,q}$ et de leur norme $\|.\|_{B^s_{p,q}}$ à partir des modules de régularité de f [18, 16]. Si on suppose que l'ondelette choisie ψ possède au moins $s+1$ moments nuls et est de régularité au moins C^{s+1}, alors si $f \in B^s_{p,q}$, la norme $\|f\|_{B^s_{p,q}}$ est équivalente à :

$$\left(\sum_k \left(\sum_j 2^{skp} 2^{k(p-2)} |c_{j,k}|^p \right)^{\frac{p}{q}} \right)^{\frac{1}{q}} \tag{4.43}$$

(les constantes d'équivalence dépendent de l'ondelette choisie).

Dans la suite, nous utiliserons toujours la norme équivalente (4.43) pour $\|f\|_{B^s_{p,q}}$.

Dans notre travail, nous nous intéresserons essentiellement aux versions homogènes des espaces de Besov, définies par :

$$\dot{B^s_{p,q}} = B^s_{p,q} / \left\{ u \in B^s_{p,q} \ / \ \nabla u = 0 \right\} \tag{4.44}$$

Définition 4.1. $\dot{B}^1_{1,1}$ est l'espace de Besov homogène usuel (cf [31]). Soit $\psi_{j,k}$ une base orthonormale composée d'ondelettes régulières à supports compacts. $\dot{B}^1_{1,1}$ est un sous-espace de $L^2(\mathbb{R}^2)$ et une fonction f appartient à $\dot{B}^1_{1,1}$ si et seulement si :

$$\sum_{j \in \mathbb{Z}} \sum_{k \in \mathbb{Z}^2} |c_{j,k}| 2^{j/2} < +\infty \tag{4.45}$$

Dans [31], l'auteur suggère l'utilisation de l'espace dual de $\dot{B}^1_{1,1}$, l'espace $E = \dot{B}^\infty_{-1,\infty}$ pour modéliser les phénomènes oscillants.

Définition 4.2. L'espace dual de $\dot{B}^1_{1,1}$ est l'espace de Banach $E = \dot{B}^\infty_{-1,\infty}$. Il est caractérisé par le fait que les coefficients en ondelettes d'une fonction généralisée dans $E = \dot{B}^\infty_{-1,\infty}$ sont dans $l^\infty(\mathbb{Z} \times \mathbb{Z}^2)$.

4.6.2 Algorithme de seuillage

Une application particulièrement intéressante des ondelettes concerne le débruitage d'image. Si on suppose qu'une image originale u a été dégradée par un bruit blanc gaussien, une méthode efficace pour restaurer l'image dégradée f consiste à seuiller ses coefficients en ondelettes.

Définissons un opérateur de seuillage :

$$\theta_\tau(t) = \begin{cases} t - \tau \text{ si } t \geq \tau \\ 0 \text{ si } t \leq \tau \\ t + \tau \text{ si } t \leq -\tau \end{cases} \tag{4.46}$$

L'intérêt d'utiliser une base d'ondelettes orthonormales pour restaurer l'image dégradée f provient du fait que ses coefficients en ondelettes $c_{j,k}(f)$ sont des variables aléatoires gaussiennes centrées de variance σ (σ correspondant au niveau du bruit blanc gaussien).

Le seuillage en ondelette doux de f de paramètre τ, noté $WST(f, \tau)$ dans ce cours (Wavelet Soft Thresholding), est la fonction dont les coefficients en ondelette s'écrivent $\theta_\tau(c_{j,k}(f))$ La valeur théorique proposée par Donoho est $\tau = \sigma\sqrt{2\log(N^2)}$, où N^2 désigne la taille de l'image (en pratique, cette valeur du seuil est souvent trop forte).

Pour plus de détails, nous renvoyons le lecteur à [22, 28, 31, 30].

FIGURE 19 – Restauration par seuillage en ondelettes (Haar)

4.6.3 Interprétation variationnelle

Considérons la fonctionnelle :

$$\inf_u \|f-u\|^2 + 2\tau \|u\|_{\dot{B}^1_{1,1}} \tag{4.47}$$

La solution du problème (4.47) est donnée par :

$$u = WST(f,\tau) \tag{4.48}$$

Tester l'opérateur de seuillage doux avec différents type d'ondelettes (Haar, dbN, symN, ...) (instruction *helpwin wfilters* pour connaître les différents filtres disponibles). Utiliser d'abord la transformée en ondelettes non invariante par translation (intruction *wavedec2*). Pour le seuillage, on pourra utiliser l'instruction *wthresh*. Comparer ensuite avec la transformée en ondelettes invariante par translation (instruction *swt2*). Calculer les différents SNR. On rappelle que le SNR est défini comme $10 \log \frac{\|u\|^2}{\|f-u\|^2}$ avec f l'image observée et u l'image à reconstruire. Commentaires ? Comparer avec l'approche par minimisation de la variation totale.

Les Figure 19 et 20 montrent des exemples de restauration.

4.7 Modèle d'Osher-Sole-Vese

Nous présentons ici un modèle proposé par Osher-Solé-Vese dans [34]. Nous proposons un algorithme original pour le résoudre, algorithme inspiré de celui d'A. Chambolle (pour plus de détails, voir [11]).

Le problème considéré est :

$$\inf_u \left(J(u) + \frac{1}{2\lambda} \|f-u\|^2_{-1,2} \right) \tag{4.49}$$

FIGURE 20 – Restauration par seuillage en ondelettes (Daubechies 12)

où $\|.\|_{-1,2}$ est une version discrète de la norme de $W^{-1,2}$, l'espace dual de $W_0^{1,2}$. La méthode proposée dans [34] consiste à résoudre une EDP du quatrième ordre. On va utiliser ici un algorithme de projection.

On a le résultat suivant :

Proposition 4.2. *Si $\hat{u}$ est solution du problème (4.49), alors $\hat{v} = \Delta^{-1}\left(\frac{\hat{u}-f}{\lambda}\right)$ est solution du problème :*

$$\inf_v \left(\frac{\lambda}{2}\|\nabla v\|_Y^2 - (f,v)_X + J^*(v) \right) \tag{4.50}$$

Algorithme : D'après la Proposition 4.2, on a

$$\hat{u} = f + \lambda \Delta \hat{v} \tag{4.51}$$

Donc pour calculer $\hat{u}$, il suffit de calculer $\hat{v}$ solution de (4.50). Comme J^* est donnée par (4.26), $\hat{v}$ est en fait solution du problème :

$$\min \left\{ \frac{\lambda}{2}\|\nabla \mathrm{div}\,(p)\|_Y^2 + (\nabla f, p)_Y : p \ / \ |p_{i,j}| \leq 1 \ \forall i,j = 1,\ldots,N \right\} \tag{4.52}$$

En reprenant la démarche de [15], on voit qu'on peut résoudre ce problème par une méthode de point fixe.

$$p^0 = 0 \tag{4.53}$$

et

$$p_{i,j}^{n+1} = \frac{p_{i,j}^n - \tau(\nabla(\Delta \mathrm{div}\,(p^n) + f/\lambda))_{i,j}}{1 + \tau|(\nabla(\Delta \mathrm{div}\,(p^n) + f/\lambda))_{i,j}|} \tag{4.54}$$

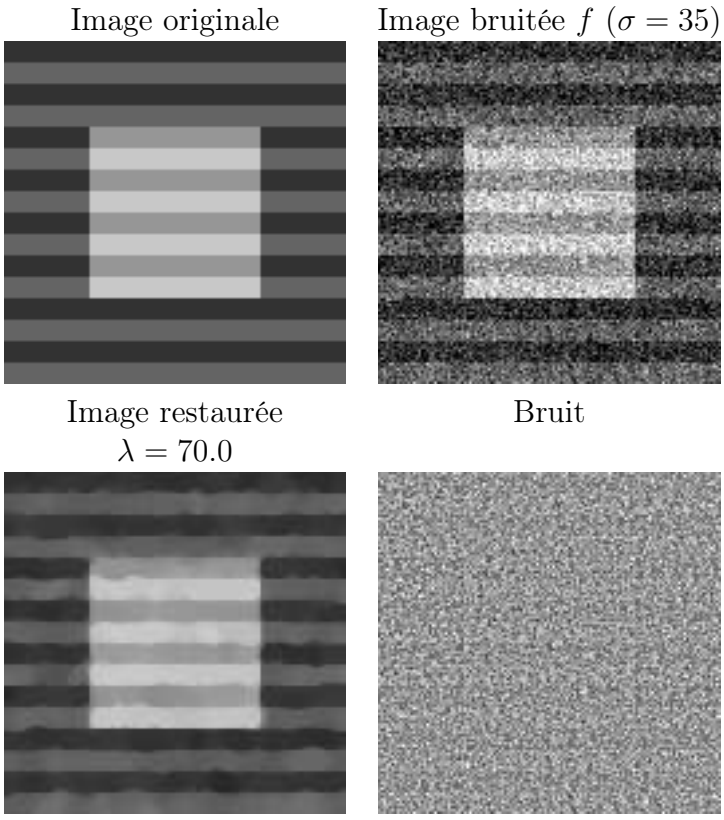

FIGURE 21 – Débruitage de rayures avec le modèle d'Osher-Sole-Vese

Et comme dans [15], on montre que pour τ suffisamment petit, on a $f + \lambda\Delta\mathrm{div}\, p^n \to \hat{u}$ quand $n \to \infty$.

Un exemple de débruitage avec cet algorithme est présenté sur la Figure 21.

4.8 Extension : modèle $u + v$

Le modèle de Rudin-Osher-Fatemi a été revisité par Meyer dans [31]. Il a proposé une nouvelle fonctionnelle :

$$\inf_{f=u+v} J(u) + \|v\|_G \tag{4.55}$$

où G est un espace adapté aux fonctions oscillantes(cf aussi [3] pour une définition de G dans le cas continu).

Une première approche numérique pour résoudre ce problème a été proposée dans [46]. Nous présentons ici l'approche de [8, 9].

On introduit la fonctionnelle suivante définie sur $X \times X$:

$$F_{\lambda,\mu}(u,v) = J(u) + \frac{1}{2\lambda}\|f - u - v\|^2_{L^2(\Omega)} + \chi_{G_\mu}(v) \tag{4.56}$$

$$G = \{\mathrm{div}\,(g)/g \in Y, |g_{i,j}| \leq 1\ \forall i,j\} \tag{4.57}$$

On définit G_μ par :

$$G_\mu = \{v \in G\ /\ \|v\|_G \leq \mu\} \tag{4.58}$$

Comme J^* est la fonction indicatrice de G_1, on peut réécrire (4.56) comme :

$$F_{\lambda,\mu}(u,v) = \frac{1}{2\lambda}\|f-u-v\|_X^2 + J(u) + J^*\left(\frac{v}{\mu}\right) \tag{4.59}$$

Avec cette formulation, on voit bien les rôles symétriques joués par u et v.

Le problème que l'on veut résoudre est :

$$\inf_{(u,v)\in X\times X} F_{\lambda,\mu}(u,v) \tag{4.60}$$

Pour cela, nous résolvons les deux problèmes suivants :

— A v fixé, on cherche

$$\inf_{u\in BV}\left(J(u) + \frac{1}{2\lambda}\|f-u-v\|_2^2\right) \tag{4.61}$$

— A u fixé

$$\inf_{v\in G_\mu} \|f-u-v\|_2^2 \tag{4.62}$$

On sait (Proposition 4.1) que la solution de (4.61) est donnée par :

$$\hat{u} = f - v - P_{G_\lambda}(f-v) \tag{4.63}$$

Et la solution de (4.62) est simplement donnée par :

$$\hat{v} = P_{G_\mu}(f-u) \tag{4.64}$$

4.8.1 Algorithme

Pour résoudre le problème (4.60), nous résolvons les problèmes (4.61) et (4.62) de manière alternée.

1. Initialisation :
$$u_0 = v_0 = 0 \tag{4.65}$$
2. Itérations :
$$v_{n+1} = P_{G_\mu}(f-u_n) \tag{4.66}$$
$$u_{n+1} = f - v_{n+1} - P_{G_\lambda}(f-v_{n+1}) \tag{4.67}$$
3. Test d'arrêt : On arrête lorsque
$$\max(|u_{n+1}-u_n|, |v_{n+1}-v_n|) \leq \epsilon \tag{4.68}$$

Un exemple de décomposition d'image est présenté sur la Figure 22.

Nous renvoyons le lecteur intéressé par ces modèles de décompositions, et par le choix de normes adaptés à chaque composante, à [11] où ces questions sont étudiées.

FIGURE 22 – Exemple de décomposition d"image

5. Segmentation : contours actifs

5.1 Introduction

Un objectif important en traitement d'images est de pouvoir déterminer les contours d'objets 2D ou 3D.
Pour cela, un moyen efficace consiste à utiliser la méthode des contours actifs. On fait évoluer une courbe (un snake) autour (ou à l'intérieur) de l'objet à détecter. Ce snake va progressivement se "coller" sur le contour à détecter.
L'idée fondamentale (introduite par Osher et Sethian) est de considérer le snake comme la courbe de niveau zéro d'une fonction u de dimension supérieure.

Il existe une abondante littérature sur les contours actifs [5, 14, 43, 33].

5.1.1 Rappels de Géométrie différentielle

Nous nous inspirons ici de [5].

Soit $X(p) = (x_1(p), x_2(p))$ une courbe plane paramétrée ($p \in [0, 1]$).

Notations

- Vecteur tangent : $\vec{\tau}(p) = (x_1^{'}(p), x_2^{'}(p))$
- Vecteur normal (extérieur) : $\vec{\mathcal{N}}(p) = (-x_2^{'}(p), x_1^{'}(p))$
- Abcisse curviligne : $s(p) = \int_0^p \|\vec{\tau}(q)\| dq$
- Courbure : κ

Courbes de niveau Supposons que la courbe X(s) paramétrée par son abcisse curviligne soit la courbe de niveau r d'une fonction $u : \mathbb{R}^2 \mapsto \mathbb{R}$. $X = \{(x_1, x_2) \ / \ u(x_1, x_2) = r\}$ Alors :

- Vecteur normal extérieur : ∇u
- Courbure : $\kappa = \text{div}\,(\frac{\nabla u}{\|\nabla u\|})$

5.1.2 Evolution de courbes

Supposons maintenant que :

- $X(p)$ dépend d'un paramètre t ≥ 0. $\Rightarrow X(p, t)$

- X est fermée : $X(0,t) = X(1,t)$
- $\frac{\partial X}{\partial p}(0,t) = \frac{\partial X}{\partial p}(1,t)$

Soit

$$L(t) = \int_0^1 \left\| \frac{\partial X}{\partial p}(p,t) \right\| dp = \text{ longueur de } X(p,t) \tag{5.1}$$

La décroissance de $L(t)$ est maximum quand :

$$\frac{\partial X}{\partial t} = \kappa \vec{\mathcal{N}} \tag{5.2}$$

Il est possible d'introduire un poids positif ϕ dans l'intégrale de longueur :

$$L(t) = \int_0^1 \phi \left\| \frac{\partial X}{\partial p} \right\| dp \tag{5.3}$$

La décroissance de $L(t)$ est maximum quand :

$$\frac{\partial X}{\partial t} = (\phi\kappa - \nabla\phi.\vec{\mathcal{N}})\vec{\mathcal{N}} \tag{5.4}$$

Version courbes de niveaux Supposons que l'on ait un flot général :

$$\frac{\partial X}{\partial t} = F\vec{\mathcal{N}} \tag{5.5}$$

où $F = F(X, X', X'')$ et $X = \{(x_1, x_2) \ / \ u(x_1, x_2) = r\}$
Alors, u suit l'équation d'évolution suivante :

$$\frac{\partial u}{\partial t} = F\|\nabla u\| \tag{5.6}$$

5.2 Applications aux images

5.2.1 Préliminaires

L'objectif est de détecter un contour fermé Γ dans une image.
On part d'une "grande" courbe fermée $X_0(p) = X(p, t = 0)$ entourant Γ, et on fait évoluer X selon une équation d'évolution du type de celles calculées dans la sous-section 5.1.2, de sorte que $X(p,t)$ vienne converger vers Γ quand t tend vers $+\infty$.
Notons I l'intensité (préalablement lissée) de l'image.
On considère un contour Γ comme étant :

$$\Gamma = \{(x_1, x_2) \ / \|\nabla I\|(x_1, x_2) = +\infty\} \tag{5.7}$$

soit encore :

$$\Gamma = \{(x_1, x_2) \ / g(\|\nabla I\|(x_1, x_2)) \simeq 0\} \tag{5.8}$$

où g est typiquement de la forme :

$$g(t) = \frac{1}{1+t^2} \tag{5.9}$$

5.2.2 L'équation d'évolution

Considérons l'équation (5.4) en prenant $\phi(x_1, x_2) = g(\|\nabla I\|(x_1, x_2))$. ϕ est donc un terme d'arrêt du contour actif sur le contour Γ à détecter.
Nous obtenons alors (à l'aide de la version courbes de niveaux de l'équation) :

$$\frac{\partial u}{\partial t} = g(\|\nabla I\|(x))\text{div}\left(\frac{\nabla u}{\|\nabla u\|}\right)\|\nabla u\| + \nabla g.\nabla u \tag{5.10}$$

avec $\vec{\mathcal{N}} = -\frac{\nabla u}{\|\nabla u\|}$ et $k = \text{div}\left(\frac{\nabla u}{\|\nabla u\|}\right)$.
De plus, pour accélérer la convergence du snake et aussi pour pouvoir détecter les parties concaves, on ajoute à la courbure κ une constante c (appelée force ballon) de sorte que κ+c garde un signe constant.
Nous obtenons ainsi le modèle final suivant :

$$\frac{\partial u}{\partial t} = g(\|\nabla I\|(x))\|\nabla u\|(\kappa + c) + \nabla g.\nabla u \tag{5.11}$$

avec $u(x, t = 0) = u_0(x)$ et où on choisit par exemple $u_0(x)$=distance(x,X_0)

Schéma numérique : L'équation (5.10) possède deux types de termes : un terme parabolique (le premier), et des termes hyperboliques (les deux derniers). Ces deux types de terme ne sont pas discrétisés de la même manière. L'idée principale est que les termes paraboliques sont discrétisés par différences finies centrées, alors que l'on utilise des shémas numériques ENO (essentially non-oscillatory) pour les termes paraboliques.

5.3 Explosion de ∇u, et les méthodes pour y remédier

Pour calculer u, c'est donc l'équation (5.11) qui est utilisée. Malheureusement, on constate numériquement que ∇u explose rapidement [25]. De plus les erreurs numériques sont importantes au voisinage du squelette.
Pour palier cet inconvénient, on réinitialise périodiquement u en la fonction distance signée (de manière à avoir $\|\nabla u\|$=1).

5.4 Réinitialisation par EDP

Une méthode efficace consiste à appliquer à u l'équation (cf [38, 43, 36, 1]) :

$$\frac{\partial u}{\partial t} + signe(u_0)(\|\nabla u\| - 1) = 0 \tag{5.12}$$

De plus, pour diminuer les temps de calcul, on a souvent recours à la méthode de la bande étroite [29] : cela consiste à ne calculer les valeurs de u qu'au voisinage de la courbe de niveau {u=0}, dans un tube que l'on déplace pendant les phases de réinitialisation.

On considère une fonction u_0 uniformément continue sur $\mathbb{R}^2$. On note $\Gamma = \{x/u_0(x) = 0\}$. Pour $x \in \mathbb{R}^2$, on écrit $\epsilon_x = signe(u_0(x))$ et $d(x, \Gamma)$ la distance de x à Γ. On considère la fonction $u : \mathbb{R}^2 \times \mathbb{R}_+ \mapsto \mathbb{R}^2$ définie par :

$$u(x,t) = \begin{cases} \epsilon_x \inf_{|y|\leq t}(\epsilon_x u_0(x+y) + t) & \text{si } t \leq t_x \\ \epsilon_x d(x, \Gamma) & \text{si } t > t_x \end{cases} \tag{5.13}$$

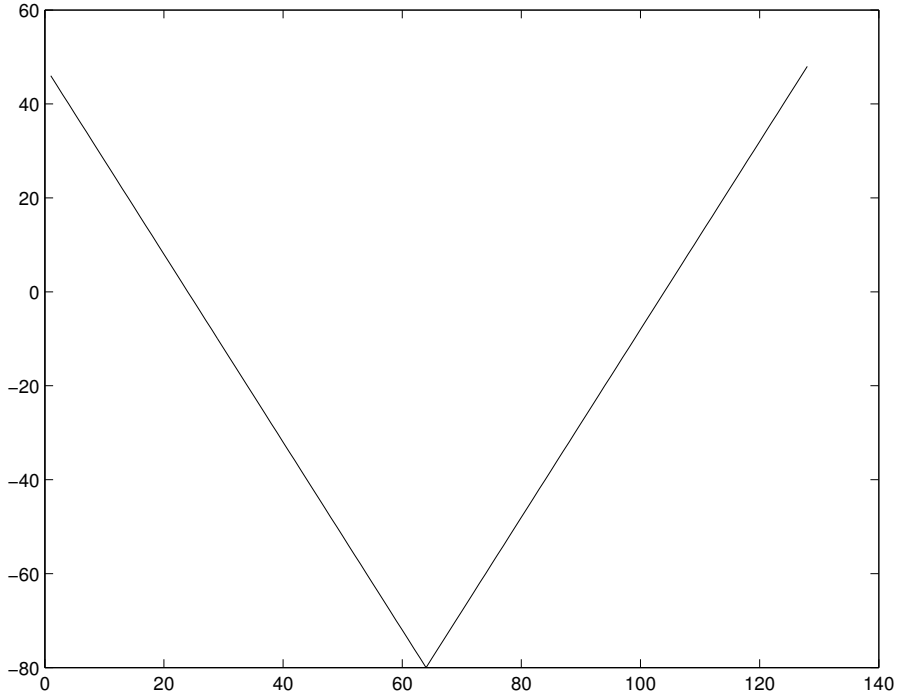

FIGURE 23 – Fonction à réinitialiser

où

$$t_x = \inf\{t \in \mathbb{R}_+ / \inf_{|y|\leq t} (\epsilon_x u_0(x+y)) = 0\} = d(x,\Gamma) \tag{5.14}$$

Théorème 5.1. *Soit u définie par (5.13). Alors u est solution de viscosité de (5.12). De plus, u est unique dans la classe des solutions de viscosité uniformément continues sur* $\mathbb{R}^2 \times [0,T]$, $\forall$ *T>0, et s'annulant sur* Γ, $\forall t \in [0,T]$.

Ce résultat est démontré dans [2, 6].

Shéma numérique

$$u_{i,j}^{n+1} = u_{i,j}^n - \delta t \left(signe(u_{i,j}^n B(u)_{i,j}^n \right) \tag{5.15}$$

où :

$$B(u)_{i,j}^n = \begin{cases} \sqrt{\max((a^+)^2,(b^-)^2) + \max((c^+)^2,(d^-)^2)} - 1 & \text{si } u_{i,j}^n > 0 \\ \sqrt{\max((a^-)^2,(b^+)^2) + \max((c^-)^2,(d^+)^2)} - 1 & \text{si } u_{i,j}^n < 0 \\ 0 & \textit{sinon} \end{cases} \tag{5.16}$$

où $a^+ = \max(a,0)$ et $a^- = \min(a,0)$, et avec :

$$\begin{aligned} a &= \delta_x^- u_{i,j}^n \\ b &= \delta_x^+ u_{i,j}^n \\ c &= \delta_y^- u_{i,j}^n \\ d &= \delta_y^+ u_{i,j}^n \end{aligned}$$

Un exemple de réinitialisation (en 1D) est présenté sur les Figures 23 et 24.

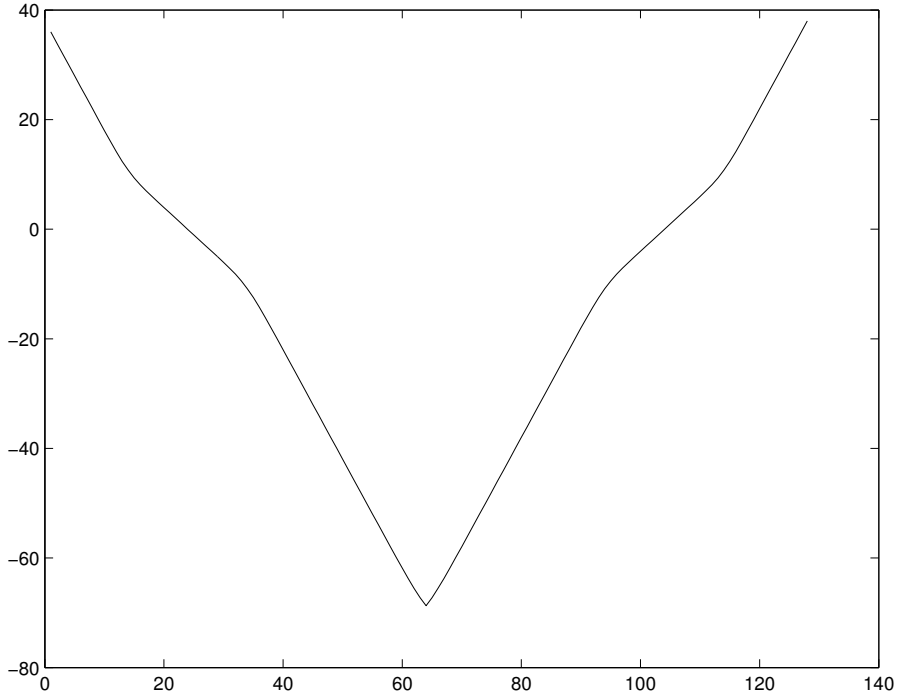

FIGURE 24 – Réinitialisation

5.4.1 Fast Marching

Le but de cette méthode est de résoudre l'équation suivante sur une grille avec $F > 0$:

$$\|\nabla u\| = F \tag{5.17}$$

(dans le cas de la réinitialisation, on a $F = 1$). On discrétise le terme de gauche de l'équation : le gradient. L'idée de la méthode est de propager les valeurs de u des valeurs les plus faibles aux valeurs les plus élévées. La discrétisation utilisée pour le gradient est la suivante :

$$\sqrt{(\max\{u_{i,j} - u_{i-1,j}, u_{i,j} - u_{i+1,j}, 0\})^2 + (\max\{u_{i,j} - u_{i,j-1}, u_{i,j} - u_{i,j+1}, 0\})^2} = F_{i,j} \tag{5.18}$$

Principe L'ensemble des points de la grille où l'équation (5.17) est à résoudre est divisé en trois sous-ensembles. L'ensemble des points *acceptés*, l'ensemble des points *proches*, et celui des points *éloignés*. La méthode consiste à étendre l'ensemble des points *acceptés* jusqu'à ce que les deux autres ensembles soient vides. Les valeurs des points *acceptés* ne changent plus. L'algorithme consiste à itérer les étapes suivantes :

1. Sélectionner le point (i_{min}, j_{min}) de la grille telle que ce point possède la valeur minimale des points qui sont dans l'ensemble *proches*.
2. Ce point passe de l'ensemble *proches* à l'ensemble *accepté*.
3. Les points voisins qui sont dans l'ensemble *éloignés* passent dans l'ensemble *proches*.
4. La valeur des points voisins qui sont dans l'ensemble *proches* sont mis à jour en résolvant l'équation (5.18) en mettant à $+\infty$ la valeur des points des ensembles *proches* et *éloignés* dans l'équation.

Pour plus de détails sur cette méthode, on renvoit le lecteur à [33, 1].

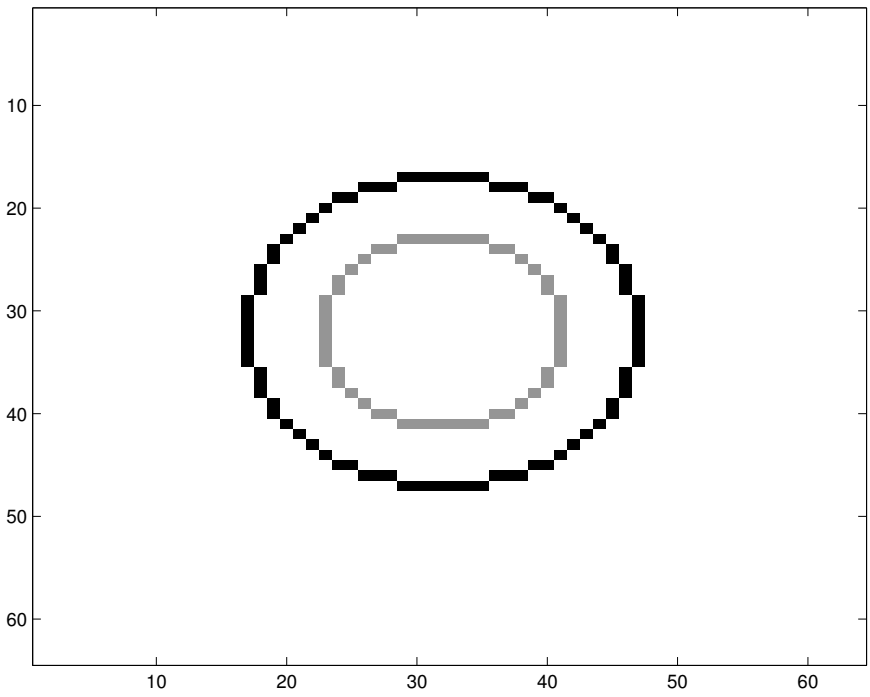

FIGURE 25 – Mouvement par courbure moyenne (en noir, position de la courbe à t=0, en gris, position de la courbe à un instant $t > 0$, et à la fin, la courbe converge vers le centre de l'image)

5.5 Retour sur le mouvement par courbure moyenne

On rappelle l'équation du mouvement par courbure moyenne.

$$\begin{cases} \frac{\partial u}{\partial t} = |\nabla u| \text{div} \left(\frac{\nabla u}{|\nabla u|} \right) \\ u(0,x) = u_0(x) \end{cases} \tag{5.19}$$

La programmer avec l'étape de réinitialisation (cf Figures 25 et 26).

5.6 Equation de vitesse constante

$$\begin{cases} \frac{\partial u}{\partial t} = c|\nabla u| \\ u(0,x) = u_0(x) \end{cases} \tag{5.20}$$

où c est une constante. Cette équation décrit un mouvement dans la direction normale à son front.

Schéma numérique :

$$u_{i,j}^{n+1} = u_{i,j}^n + c\delta t \nabla^+ u_{i,j}^n \tag{5.21}$$

5.7 Equation d'advection

$$\begin{cases} \frac{\partial u}{\partial t} = A(x).\nabla u \\ u(0,x) = u_0(x) \end{cases} \tag{5.22}$$

où $A(x) = (A_1(x), A_2(x))$.

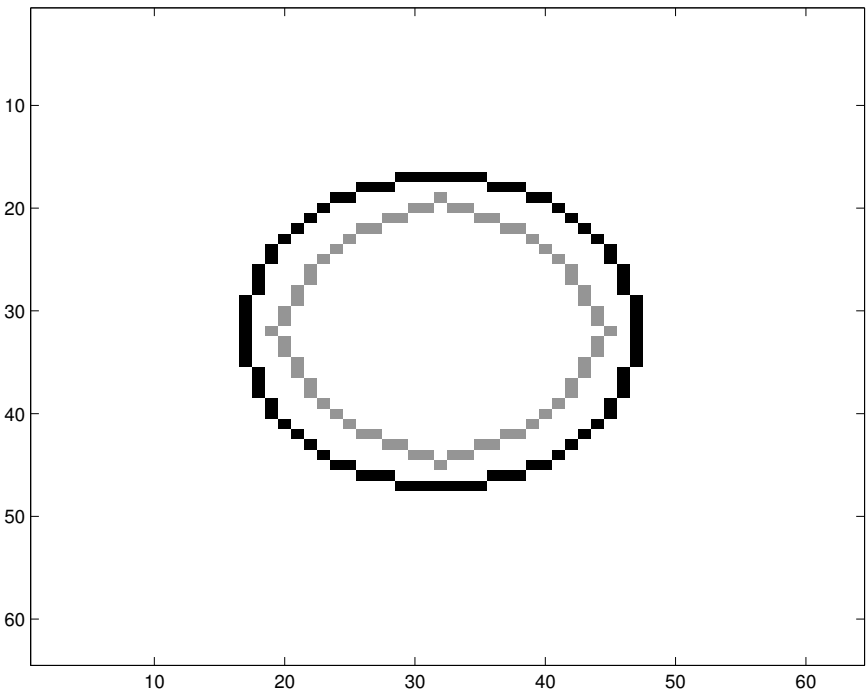

FIGURE 26 – Mouvement par AMSS (en noir, position de la courbe à t=0, en gris, position de la courbe à un instant $t > 0$, et à la fin, la courbe converge vers le centre de l'image)

Schéma numérique :

$$\begin{aligned} u_{i,j}^{n+1} &= u_{i,j}^n + \delta t \left[\max((A_1)_{i,j}^n, 0)\delta_x^- u_{i,j}^n + \min((A_1)_{i,j}^n, 0)\delta_x^+ u_{i,j}^n \right. \\ &\quad \left. + \max((A_2)_{i,j}^n, 0)\delta_y^- u_{i,j}^n + \min((A_2)_{i,j}^n, 0)\delta_y^+ u_{i,j}^n \right] \end{aligned}$$

5.8 Simulation numérique

Programmer maintenant l'équation des contours actifs géodésiques. On rappelle que le schéma s'écrit finalement :

$$\begin{aligned} u_{i,j}^{n+1} &= u_{i,j}^n + \delta t \left[g_{i,j} K_{i,j}^n \sqrt{(\delta_x u_{i,j}^n)^2 + (\delta_y u_{i,j}^n)^2} \right. \\ &\quad + \alpha \left(\max(g_{i,j}, 0)\nabla^+ + \min(g_{i,j}, 0)\nabla^- \right) u_{i,j}^n \\ &\quad + \max((g_x)_{i,j}^n, 0)\delta_x^- u_{i,j}^n + \min((g_x)_{i,j}^n, 0)\delta_x^+ u_{i,j}^n \\ &\quad \left. + \max((g_y)_{i,j}^n, 0)\delta_y^- u_{i,j}^n + \min((g_y)_{i,j}^n, 0)\delta_y^+ u_{i,j}^n \right] \end{aligned}$$

Un exemple de segmentation de l'image sur la Figure 27 est donné sur la Figure 28.

5.9 Extension : classification d'image par courbes de niveaux

5.9.1 Introduction

L'utilisation des contours actifs en classification d'images est développée dans plusieurs articles [10, 7, 17, 45, 44, 35, 40, 41, 42, 5]. Nous reprenons ici l'approche de [41, 10].

La classification consiste à attribuer une étiquette à chaque pixel d'une image, cette étiquette indiquant à quelle classe appartient le pixel. Elle peut être vue comme un problème de partition.

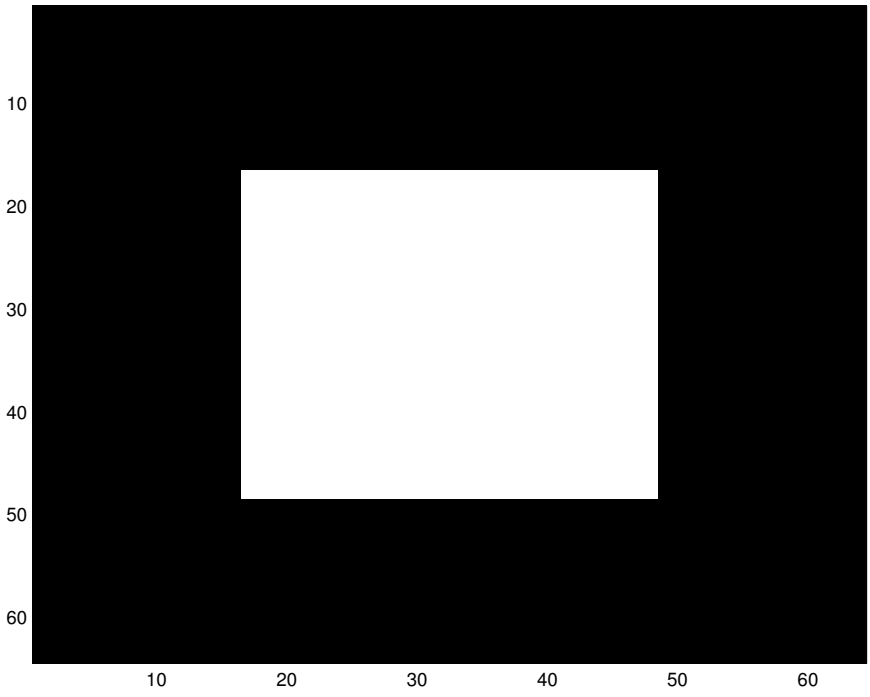

FIGURE 27 – Image initiale

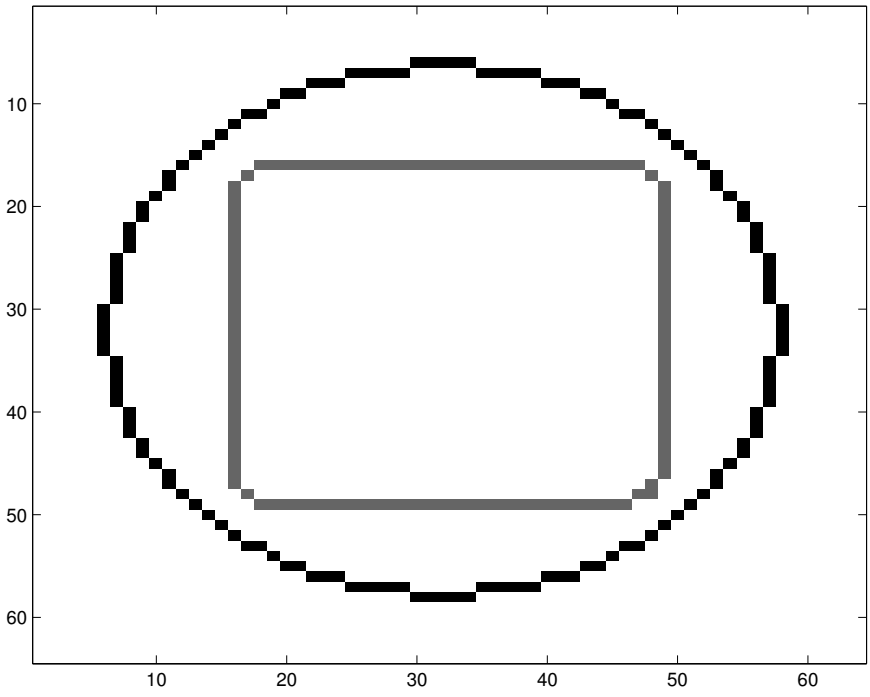

FIGURE 28 – Initialisation du contour actif (courbe en noir), et résultat final (courbe en gris)

C'est un des objectifs de base du traitement d'images. Elle intervient dans de nombreuses applications, comme par exemple la télédétection. La classification est un problème très proche de celui de la segmentation, dans le sens où le but consiste à obtenir une partition de l'image en régions homogènes. Dans la classification, chaque sous-ensemble de la partition obtenue représente une classe.

5.9.2 Principe

Nous supposons connu le nombre K de classes dans l'image, ainsi que leurs caractéristiques. Si $1 \leq k \leq K$, on note C_k la classe correspondante. L'objectif est d'obtenir une partition de l'image par les différentes classes (nous considérons le problème de classification comme un problème de partition). Pour cela, nous allons minimiser une fonctionnelle comportant trois termes :

(i) un terme de partition des classes (i.e. chaque pixel est classé dans une classe et une seule) (en fait, comme nous n'allons considérer que le cas de deux classes, nous n'aurons pas besoin de ce terme).

(ii) un terme de régularisation des contours des classes.

(iii) un terme d'attache aux données.

On utilise une approche variationnelle par ensembles de niveaux. Le domaine de l'image, Ω, est la réunion d'ensembles Ω_k disjoints.

Nous ne considérons ici que le cas de deux classes.

On considère l'image comme une fonction $u_0 : \Omega \mapsto \mathbb{R}$ (où Ω est un ouvert de $\mathbb{R}^2$).

On note $\Omega_k = \{x \in \Omega \,/\, \text{x est dans la classe k}\}$. La collection d'ouverts $\{\Omega_k\}$ forme une partition de Ω si et seulement si :

$$\Omega = \bigcup_k \Omega_k \bigcup_k \Gamma_k \text{ , et si } k \neq l \; \Omega_k \bigcap \Omega_l = \emptyset$$

où on a noté $\Gamma_k = \partial\Omega_k \bigcap \Omega$ le bord de Ω_k (excepté les points communs à $\partial\Omega$), et $\Gamma_{kl} = \Gamma_{lk} = \Gamma_k \bigcap \Gamma_l \bigcap \Omega$ (voir figure 29).

Afin d'obtenir une formulation fonctionnelle plutôt qu'une formulation ensembliste, nous supposerons que pour chaque Ω_k il existe une fonction lipschitzienne $\Phi_k : \Omega \to \mathbb{R}$ telle que :

$$\begin{cases} \Phi_k(x) > 0 & \text{si } x \in \Omega_k \\ \Phi_k(x) = 0 & \text{si } x \in \Gamma_k \\ \Phi_k(x) < 0 & \text{sinon} \end{cases}$$

Ω_k est ainsi entièrement déterminé par Φ_k.

Comme nous considérons le cas de deux classes, nous pouvons nous contenter d'une seule fonction ϕ.

5.9.3 Régularisation

Nous utiliserons plus loin les distributions de Dirac δ et Heaviside H. Pour que ce que nous écrivons ait un sens mathématique, nous aurons besoin des approximations régulières classiques suivantes de ces distributions (voir figure 30) :

$$\delta_\alpha = \begin{cases} \frac{1}{2\alpha}\left(1 + \cos\frac{\pi s}{\alpha}\right) & \text{si } |s| \leq \alpha \\ 0 & \text{si } |s| < \alpha \end{cases} \tag{5.23}$$

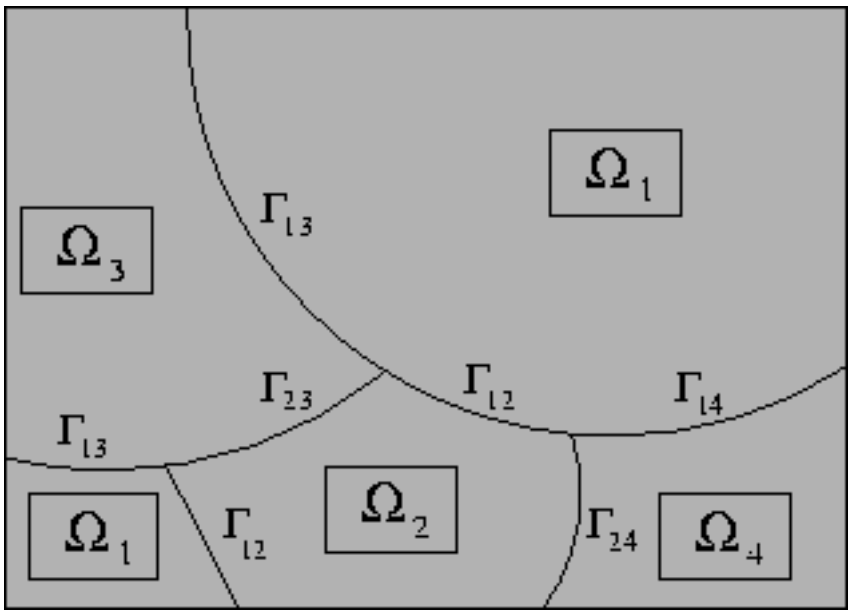

FIGURE 29 – Classification vue comme un problème de partition

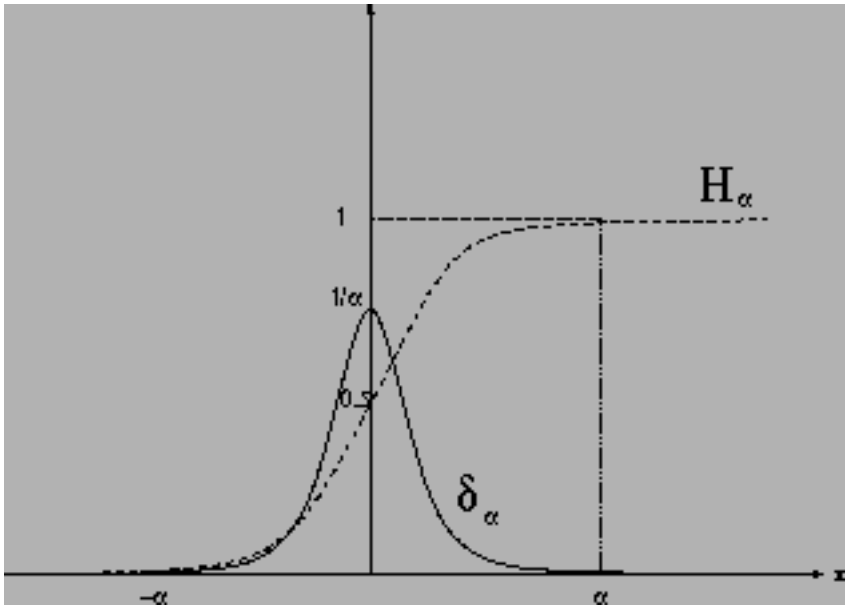

FIGURE 30 – Approximations δ_α et H_α des distributions de Dirac et de Heaviside

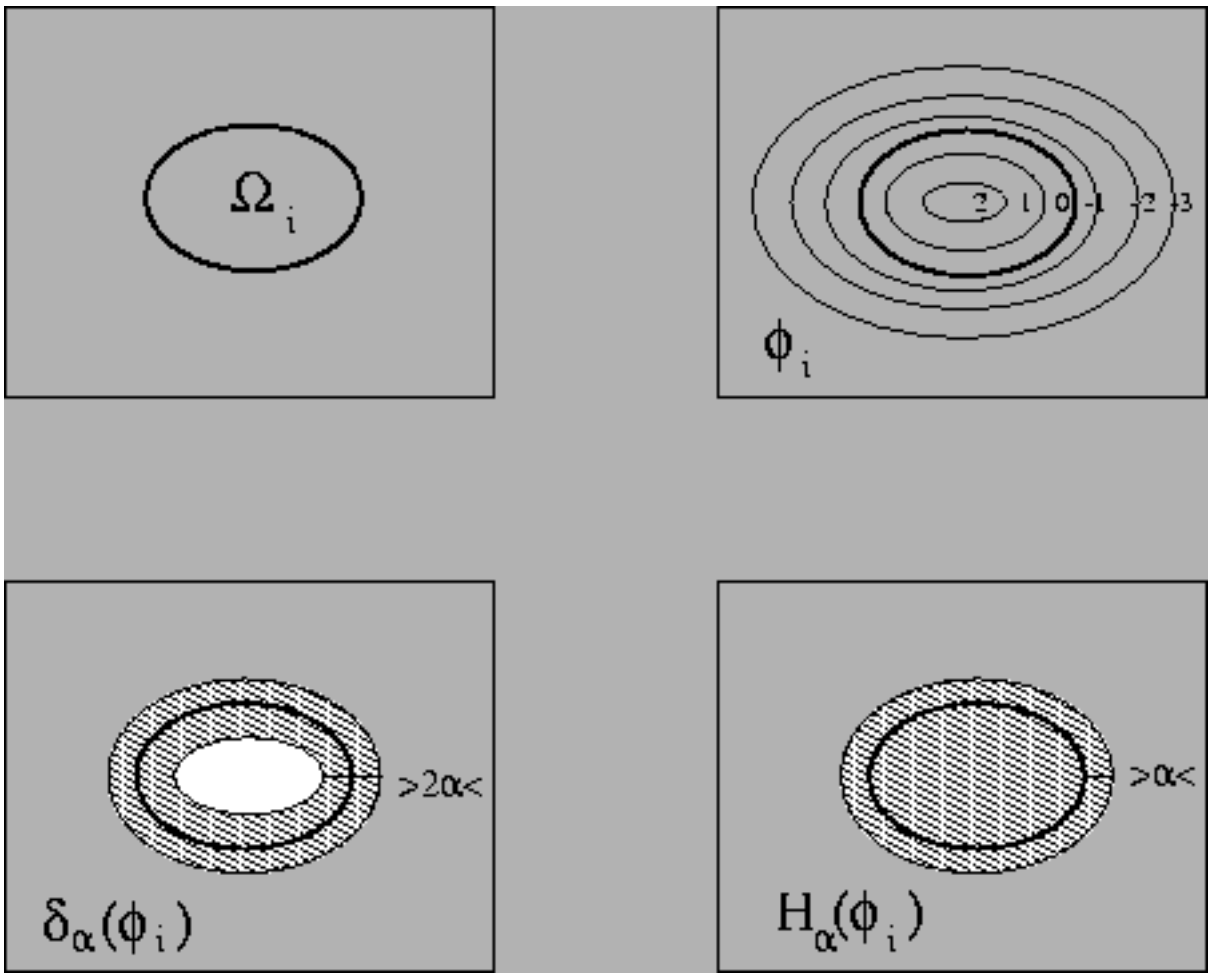

FIGURE 31 – Définitions des régions et de leurs contours par ensemble de niveau.

$$H_\alpha = \begin{cases} \frac{1}{2}\left(1 + \frac{s}{\alpha} + \frac{1}{\pi}\sin\frac{\pi s}{\alpha}\right) & \text{si } |s| \leq \alpha \\ 1 & \text{si } s > \alpha \\ 0 & \text{si } s < -\alpha \end{cases} \tag{5.24}$$

Quand $\alpha \to 0$, on a $\delta_\alpha \to \delta$ et $H_\alpha \to H$ (au sens des distributions).

La figure 31 montre comment les régions sont définies par ces distributions et les ensembles de niveaux. On a :

$$\int_{\Omega_k} f(x)\,dx = \int_\Omega H(\phi_k) f(x)\,dx \tag{5.25}$$

5.9.4 Termes d'énergie

La fonctionnelle va comporter deux termes :

1.
$$F^A(\phi) = \gamma\,|\Gamma| \tag{5.26}$$

$\Longrightarrow$ terme de pénalisation de la longueur des contours. Ce terme pénalise la longueur des contours, ce qui empêche d'avoir des contours trop irréguliers. On peut montrer, en

utilisant la formule de la co-aire, que (la démonstration est donnée dans [41]) :

$$\lim_{\alpha \to 0} \int_{\Omega} \delta_{\alpha}(\phi) |\nabla \phi| dx = |\Gamma| \tag{5.27}$$

En pratique, on cherche donc à minimiser :

$$F_{\alpha}^{A}(\phi) = \gamma \int_{\Omega} \delta_{\alpha}(\phi) |\nabla \phi| \tag{5.28}$$

2.

$$F^{B}(\phi) \tag{5.29}$$

$\Longrightarrow$ terme d'attache aux données.

Chacune des deux classes sera caractérisée par son niveau de gris moyen. Si on suppose que le niveau de gris moyen de chaque classe suit une distribution gaussienne, la méthode du maximum de vraisemblance permet d'écrire :

$$F^{B}(\phi) = \int_{\Omega} H_{\alpha}(\phi) \frac{(u(x) - m_1)^2}{\sigma_1^2} + \int_{\Omega} H_{\alpha}(-\phi) \frac{(u(x) - m_2)^2}{\sigma_2^2} \tag{5.30}$$

où les m_i sont les niveaux de gris moyens, et σ_i les écart-types.

Energie totale : L'énergie que nous voulons minimiser est la somme des trois termes précédents :

$$F(\Phi_1, \ldots, \Phi_K) = F^{A}(\phi) + F^{B}(\phi) \tag{5.31}$$

Pour l'étude théorique de la fonctionnelle, on renvoit le lecteur à [4].

5.9.5 Equations d'Euler-Lagrange

Si ϕ minimise la fonctionnelle F, alors nécessairement (si en plus ϕ vérifie certaines conditions de régularité (voir [24] par exemple)), on a :

$$\frac{\partial F}{\partial \phi} = 0 \tag{5.32}$$

En supposant que des conditions de Neumann sont vérifiées $\left(\frac{\partial \phi}{\partial \vec{n}}(x) = 0 \ , \ \forall x \in \partial\Omega\right)$, l'équations d'Euler Lagrange associées à F (cf [24]) est :

$$\delta_{\alpha}(\phi) \left(\text{div} \left(\frac{\nabla \phi}{|\nabla \phi|} \right) \left(\frac{(u(x) - m_1)^2}{\sigma_1^2} - \frac{(u(x) - m_2)^2}{\sigma_2^2} \right) \right) = 0 \tag{5.33}$$

On utilise le schéma dynamique suivant :

$$\frac{\partial \phi}{\partial t} = \delta_{\alpha}(\phi) \left(\text{div} \left(\frac{\nabla \phi}{|\nabla \phi|} \right) \left(\frac{(u(x) - m_1)^2}{\sigma_1^2} - \frac{(u(x) - m_2)^2}{\sigma_2^2} \right) \right) \tag{5.34}$$

Implémenter cette équation (ne pas oublier l'étape de réinitialisation).
Un exemple de classification est donné sur la Figure 32.

Image à classifier Classification (100 itérations)

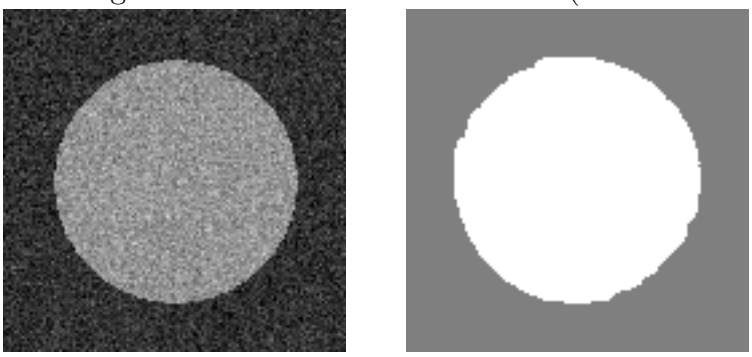

FIGURE 32 – Classification d'une image synthétique

6. Segmentation : Mumford-Shah

Pour cette section, nous renvoyons le lecteur à [5], ainsi qu'aux références données dans l'ouvrage.

6.1 Présentation du problème

On considère Ω un ouvert de $\mathbb{R}^2$, et $u_0(x)$ une image initiale. La fonctionnelle suivante a été introduite en 1989 par Mumford et Shah :

$$F(u,K) = \alpha \int_{\Omega\setminus K} (u-u_0)^2\,dx + \beta \int_{\Omega\setminus K} |\nabla u|^2\,dx + \beta \int_K d\sigma \tag{6.1}$$

où α et β sont des constantes positives, et $\int_K d\sigma$ est la longueur de K (K est supposé contenu dans Ω).

Si (u,K) est une solution de cette fonctionnelle, on peut espérer que K soit formé par les points de l'image de fort gradient, et que u soit une image relativement constante sur $\Omega\setminus K$.

Conjecture *Il existe un minimiseur de F tel que les bords (i.e. l'ensemble de discontinuité K) soient la réunion d'un nombre fini d'ensembles $C^{1,1}$ contenus dans des courbes. De plus, chaque courbe peut soit se terminer au bord de Ω, soit en jonction triple.*

6.2 Approche par fonctionnelles elliptiques

Il existe de nombreuses méthodes pour aprocher la fonctionnelle de Mumford Shah (notamment par différences finies comme proposé par A. Chambolle).

Dans ce cours, nous ne présentons que l'approche par fonctionnelles elliptiques. Elle consiste à introduire une fonction auxiliaire v qui approche la fonction caractéristique $1-\chi_K$. Ambrosio et Tortorelli ont proposé de considérer la suite de fonctionnelles suivantes :

$$F(u,K) = \alpha \int_{\Omega\setminus K} (u-u_0)^2\,dx + \beta \int_\Omega v^2|\nabla u|^2\,dx + \int_\Omega \left(\epsilon|\nabla v|^2 + \frac{1}{4\epsilon}(v-1)^2\right) dx \tag{6.2}$$

Il est possible de montrer (en utilisant la théorie de la Gamma-convergence) que (6.2) permet de résoudre (6.1) lorsque $\epsilon \to 0$.

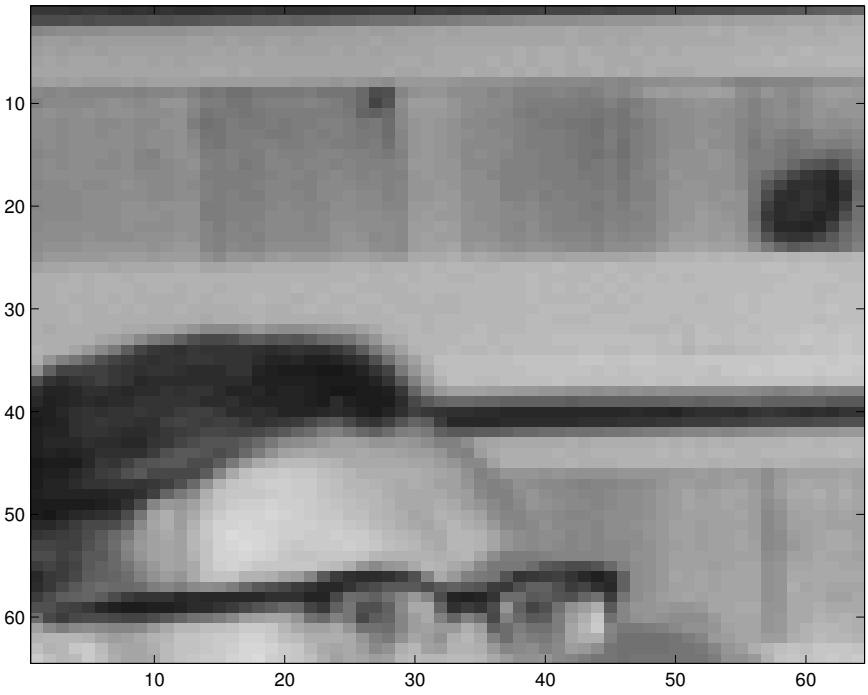

FIGURE 33 – Image originale non bruitée

6.3 Approche numérique

Les conditions d'optimalité pour (6.2) s'écrivent :

$$2\alpha(u-u_0) - 2\beta \mathrm{div}\,(v^2\nabla u) = 0 \tag{6.3}$$

et

$$-2\epsilon\Delta v + \frac{1}{2\epsilon}(v-1) + 2\beta|\nabla u|^2 v = 0 \tag{6.4}$$

On résout alors le système dynamique :

$$\frac{\partial u}{\partial t} = -2\alpha(u-u_0) + 2\beta \mathrm{div}\,(v^2\nabla u) \tag{6.5}$$

$$\frac{\partial v}{\partial t} = 2\epsilon\Delta v - \frac{1}{2\epsilon}(v-1) - 2\beta|\nabla u|^2 v \tag{6.6}$$

On utilise un schéma d'Euler explicite en temps :

$$u^{n+1} = u^n + \delta t\left(-2\alpha(u^n-u_0) + 2\beta \mathrm{div}\,((v^n)^2\nabla u^n)\right) \tag{6.7}$$

$$v^{n+1} = v^n + \delta t\left(2\epsilon\Delta v^n - \frac{1}{2\epsilon}(v^n-1) - 2\beta|\nabla u^{n+1}|^2 v^n\right) \tag{6.8}$$

Exemple On montre un exemple de segmentation obtenue sur un morceau de l'image *gatlin2*. L'image originale est présentée sur la Figure 33, l'image bruitée à segmenter sur la Figure 34, l'image obtenue u en minimisant la fonctionnelle de Mumford Shah sur la Figure 35, et la position des bords donnée par la fonction v sur la Figure 36.

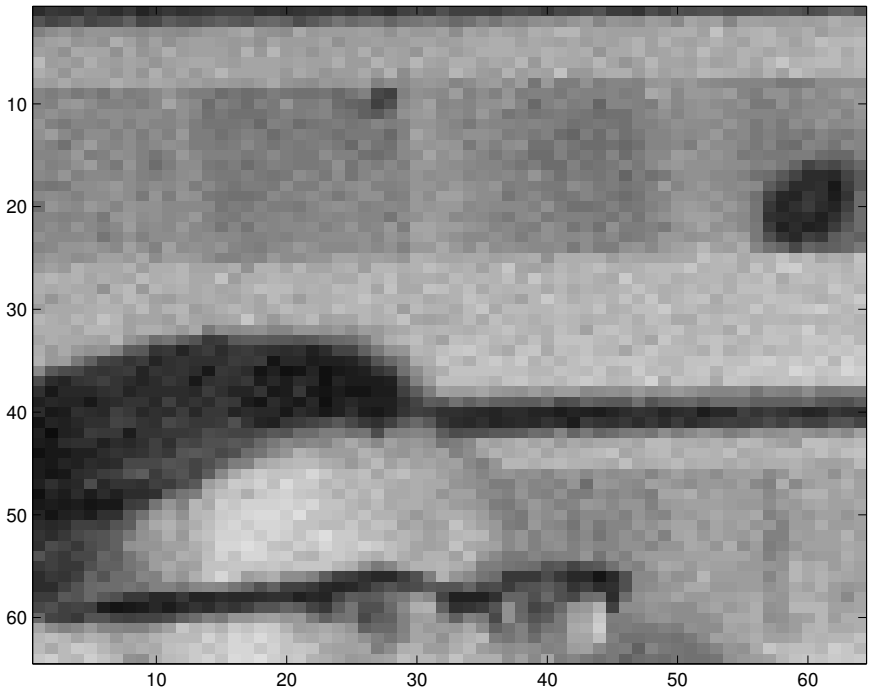

FIGURE 34 – Image bruitée à segmenter

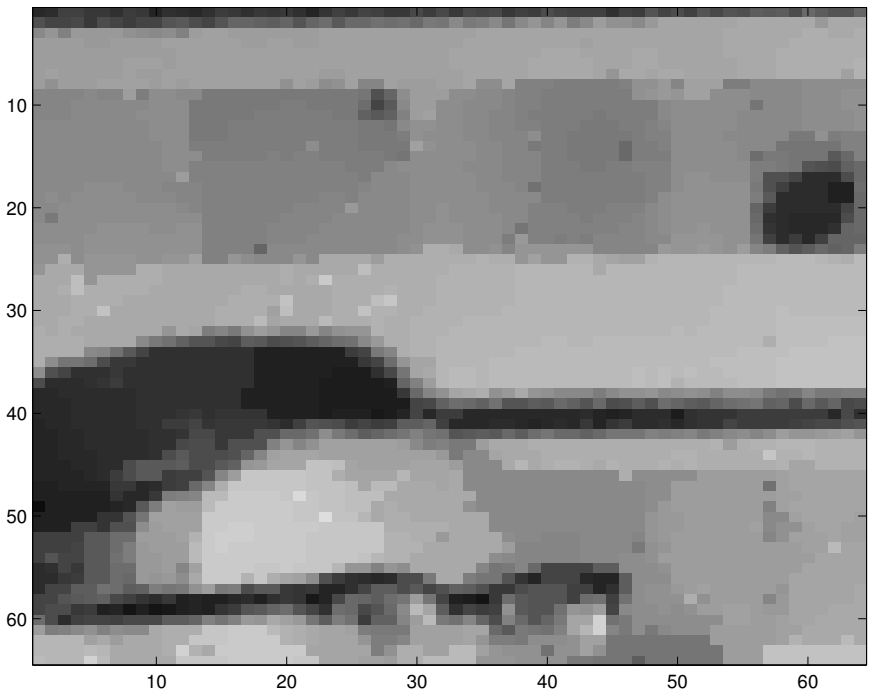

FIGURE 35 – Image restaurée

FIGURE 36 – Position des bords

7. Enoncé des TPs

Les numéros de TPs renvoit au numéro de la section du document dans laquelle les détails sont donnés.

7.1 TP1 : Introduction au traitement des images avec Matlab

(uniquement pour ceux qui n'ont pas l'habitude de manipuler des images sous matlab)

7.1.1 Ouverture d'image

Ouvrir l'image gatlin2. La visualiser en niveau de gris. Visualiser l'image clown dans une autre figure (pour cela, ouvrir une deuxième figure).

7.1.2 Bruitage d'une image

Ecrire une fonction Matlab qui permette de bruiter une image par un bruit blanc gaussien (cf la sous-section 1.2.3).

7.1.3 Extension d'une image par réflexions

Ecrire une fonction Matlab qui étende une image par réflexion par rapport à ses bords (cf la sous-section 1.3.1).

7.1.4 Discrétisations du gradient

Programmer les différentes discrétisation du gradient de la section 1.5. Comparer avec la fonction *grad*. Visualiser les bords de différentes images donnés par le gradient.

7.1.5 Détecteur de bord de Hildrett-Marr

Ecrire une fonction qui calcul le Laplacien d'une image (en se servant de la discrétisation donnée par (2.4).

Ecrire une procédure qui extrait la ligne de niveau zéro d'une fonction.

Principe du détecteur de Hildrett-Marr : on cherche les zéros du Laplacien d'une image u. Si en un point x, Δu change de signe et si ∇u est non nul, alors on considère qu'en x l'image u possède un bord. Programmer un tel détecteur de bord. Comparer avec la fonction *edge* de Matlab.

7.2 TP2 : Equation de la chaleur : filtrage linéaire et non linéaire

7.2.1 Discrétisation

Programmer les discrétisations du gradient, de la divergence et de la courbure donnée dans la section 4.2. Visualiser le gradient et le laplacien d'une image. Que constater vous ?

7.2.2 Equation de la chaleur

EDP de la chaleur Programmer l'équation de la chaleur (cf la sous-section 2.1.1). La tester sur des images non bruitées puis bruitées. Vérifier que les zone homogènes sont bien restaurées, mais par-contre que les bords sont érodés.

Convolution avec une Gaussienne Pour vérifier , effectuer la convolution d'une image par un noyau Gaussien (fonction *filter2*)

Application : Sur une image lissée, on peut plus facilement essayer de détecter les bords d'une image. On peut par exemple utiliser le détecteur de Hildrett-Marr (ou un détecteur de type Canny). Avec cet algorithme, calculer les bords d'une image lissée par l'EDP de la chaleur (comparer les résultats obtenus suivant le temps d'évolution)

7.2.3 Perona-Malik

Sur le même principe que l'EDP de la chaleur, écrire le schéma numérique pour (2.5) (une discrétisation possible de la divergence est donnée en (2.6)). Tester le nouvel algorithme avec les fonctions $c(t) = \frac{1}{\sqrt{1+(t/\alpha)^2}}$ et $c(t) = \frac{1}{1+(t/\alpha)^2}$.Vérifier les propriétés annoncées (comparer avec l'équation de la chaleur).

Convolution du gradient avec une gaussienne Tester l'approche proposé par (2.7), et la comparer avec le modèle précédent sur des images bruitées.

7.3 TP4 : Restauration d'images

7.3.1 Discrétisation

Programmer les discrétisations du gradient, de la divergence et de la courbure donnée dans la section 4.2.

7.3.2 Régularisation de Tychonov

Programmer les deux versions proposés dans la section 4.3 de la régularisation de Tychonov (par descente de gradient, ou par transformée de Fourier). On rappelle que la fonction Matlab pour calculer la transformée de Fourier d'une image est la fonction *fft2*. Vérifier les formules données dans la sous-section 4.3.2.

7.3.3 Modèle de Rudin-Osher-Fatemi

Ecrire une fonction qui réalise le modèle de Rudin-Osher-Fatemi en utilisant une descente de gradient (équation 4.22). Pour éviter les divisions par 0, utiliser un paramètre $\epsilon > 0$ (pour les expériences numériques sur des images en niveau de gris entre 0 et 255, prendre $\epsilon = 1$).

Ecrire une fonction qui réalise le modèle de Rudin-Osher-Fatemi en utilisant l'algorithme de Chambolle (cf (4.34).

Tester cet algorithme de restauration sur des images bruitées (comparer avec la régularisation de Tychonov). En pratique, l'algorithme de chambolle semble converger pour $\tau < 0.25$ (alors que la valeur théorique du théorème donne $\tau < 0.125$).

Mêmes questions avec l'algorithme de gradient projeté, puis l'algorithme de Nesterov. Comparer les vitesses de convergence.

7.3.4 Déconvolution

Programmer la descente de gradient et la méthode de Forward-Backward splitting proposé à la section 4.5. Comparer les deux approches.

7.3.5 Seuillage en ondelettes

Résoudre le problème (4.47) avec (4.48).

Tester l'opérateur de seuillage doux avec différents type d'ondelettes (Haar, dbN, symN, ...) (instruction *helpwin wfilters* pour connaître les différents filtres disponibles). Utiliser d'abord la transformée en ondelettes non invariante par translation (intruction *wavedec2*). Pour le seuillage, on pourra utiliser l'instruction *wthresh*. Comparer ensuite avec la transformée en ondelettes invariante par translation (instruction *swt2*). Comparer avec l'approche par minimisation de la variation totale.

7.4 TP6 : Mumford-Shah

En utilisant le schéma numérique (6.7)–(6.8), écrire une fonction qui réalise le progamme de Mumford-Shah. La tester.

7.5 TP3 : Théorie Scale-Space

7.5.1 Restauration par mouvement par courbure moyenne

Programmer la restauration d'image par mouvement par courbure moyenne (équation (3.13)). Pour la courbure K, utiliser la discrétisation donnée par (3.15). Attention : numériquement, dans l'expression de $K_{i,j}^n$, il peut apparaître des divisions par 0. Il convient de traiter ces cas à part dans le shéma numérique . Regarder les résultats obtenus sur différentes images (bruitées ou non). Comparer avec les résultats obtenus dans le TP précédent. Commentaires ?

7.5.2 Affine Morphological Scale-Space (AMSS)

Programmer l'équation (3.9).

Attention : pour Matlab, $(-27)^{1/3}$ n'est pas égal à -3.

Comparer les résultats obtenus (notamment avec l'équation de la chaleur).

7.6 TP5 : Contours actifs

7.6.1 Equation de réinitialisation

Programmer l'équation de réinitialisation (5.12) en utilisant le schéma (5.16). La tester.

7.6.2 Mouvement par courbure moyenne

Faire évoluer une courbe par mouvement par courbure moyenne (vérifier qu'elle se transforme d'abord en cercle avant de se réduire à un seul point). Comparer avec le mouvement par AMSS.

7.6.3 Contours actifs géodésiques

Programmer l'équation des contours actifs géodésiques (cf section 5.8). Tester cet algorithme pour segmenter des images. Regarder son comportement sur des images bruitées. Commentaires ?

Références

[1] D. Adalsteinsson and J.A. Sethian. The fast constructions of extension velocities in level set methods. *Journal of Computational Physics*, 148 :2–22, 1999.

[2] G. Aubert and J.F. Aujol. Signed distance functions and viscosity solutions of discontinuous hamilton-jacobi equations. Technical Report 4507, INRIA, July 2002.

[3] G. Aubert and J.F. Aujol. Modeling very oscillating signals. Application to image processing. *Applied Mathematics and Optimization*, 51(2), March/April 2005.

[4] G. Aubert and J.F. Aujol. Optimal partitions, regularized solutions, and application to image classification. *Applicable Analysis*, 84(1) :15–35, January 2005.

[5] G. Aubert and P. Kornprobst. *Mathematical Problems in Image Processing*, volume 147 of *Applied Mathematical Sciences*. Springer-Verlag, 2002.

[6] J.F. Aujol and G. Aubert. Signed distance functions and discontinuous Hamilton-Jacobi equations, June 2002. Workshop on Hamilton-Jacobi Equations, Cortona (Italy), TMR Viscosity Solutions and their Applications.

[7] J.F. Aujol, G. Aubert, and L. Blanc-Féraud. Wavelet-based level set evolution for classification of textured images. In *ICIP '03*, 2003.

[8] J.F. Aujol, G. Aubert, L. Blanc-Féraud, and A. Chambolle. Decomposing an image : Application to SAR images. In *Scale-Space '03*, volume 1682 of *Lecture Notes in Computer Science*, 2003.

[9] J.F. Aujol, G. Aubert, L. Blanc-Féraud, and A. Chambolle. Image decomposition into a bounded variation component and an oscillating component. *JMIV*, 22(1) :71–88, January 2005.

[10] J.F. Aujol, G. Aubert, and L. Blanc-Féraud. Wavelet-based level set evolution for classification of textured images. *IEEE Transactions on Image Processing*, 12(12) :1634–1641, 2003.

[11] J.F. Aujol and A. Chambolle. Dual norms and image decomposition models. *IJCV*, 63(1) :85–104, June 2005.

[12] G. Barles. *Solutions de Viscosité des équations de Hamilton-Jacobi*. Springer-Verlag, 1987.

[13] F. Cao. *Geometric curve evolution and image processing*, volume 1805 of *Lecture Notes in Applied Mathematics*. Springer-Verlag, 2003.

[14] V. Caselles, F. Catte, T. Coll, and F. Dibos. A geometric model for active contours. *Numerische Mathematik*, 66 :1–31, 1993.

[15] A. Chambolle. An algorithm for total variation minimization and applications. *JMIV*, 20 :89–97, 2004.

[16] A. Chambolle and P.L. Lions. Image recovery via total variation minimization and related problems. *Numerische Mathematik*, 76(3) :167–188, 1997.

[17] T.F. Chan and L.A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2) :266–77, February 2001.

[18] A. Cohen. *Numerical analysis of wavelet methods*. Elsevier, 2003.

[19] P.L. Combettes and J.C. Pesquet. Image restoration subject to a total variation constraint. *IEEE Transactions on Image Processing*, 13, 2004.

[20] M.G. Crandall. Viscosity solutions : a primer. In *Viscosity Solutions and Applications*, volume 1660 of *Lecture notes in Mathematics*. Springer, 1997.

[21] M.G. Crandall, H. Ishii, and P.L. Lions. User's guide to viscosity solutions of second order partial differential equations. *Bulletin of the American Mathematical Society*, 27(1) :1–67, july 1992.

[22] D.L. Donoho and M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432) :1200–1224, December 1995.

[23] I. Ekeland and R. Temam. *Analyse convexe et problèmes variationnels*, volume 224 of *Grundlehren der mathematischen Wissenschaften*. Dunod, second edition, 1983.

[24] L.C. Evans. *Partial Differential Equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, 1991.

[25] J. Gomes and O. Faugeras. Reconciling distance functions and level sets. *Journal of Visual Communication and Image Representation*, 11 :209–223, Avril 1999.

[26] F. Guichard and J.M. Morel. Image iterative smoothing and P.D.E.'s, 1999.

[27] J.B. Hiriart-Urruty and C. Lemarechal. *Convex Analysis ans Minimisation Algorithms I*, volume 305 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 1993.

[28] F. Malgouyres. Minimizing the total variation under a general convex constraint for image restoration. *IEEE transactions on image processing*, 11(12) :1450–1456, December 2002.

[29] R. Malladi, J.A. Sethian, and B.C. Vemuri. Shape modeling with front propagation : A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2), February 1995.

[30] S.G. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1998.

[31] Yves Meyer. Oscillating patterns in image processing and in some nonlinear evolution equations, March 2001. The Fifteenth Dean Jacquelines B. Lewis Memorial Lectures.

[32] J.M. Morel and S. Solimini. *Variational Methods in Image Segmentation*, volume 14 of *Progress in Nonlinear Differential Equations and Their Applications*. Birkhauser, 1995.

[33] S. Osher and R.P. Fedkiw. Level set methods : An overview and some recent results. *Journal of Computational Physics*, 169 :463–502, 2001.

[34] S.J. Osher, A. Sole, and L.A. Vese. Image decomposition and restoration using total variation minimization and the H^{-1} norm. *Multiscale Modeling and Simulation : A SIAM Interdisciplinary Journal*, 1(3) :349–370, 2003.

[35] N. Paragios and R. Deriche. Geodesic active regions and level set methods for supervised texture segmentation. *International Journal of Computer Vision*, 46(3), 2002.

[36] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A PDE based fast local level set method. *Journal of Computational Physics*, 155(2), November 1998.

[37] T. Rockafellar. *Convex Analysis*, volume 224 of *Grundlehren der mathematischen Wissenschaften*. Princeton University Press, second edition, 1983.

[38] E. Rouy and A. Tourin. A viscosity solutions approach to shape-from-shading. *SIAM Journal of Numerical Analysis*, 29(3) :867–884, 1992.

[39] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60 :259–268, 1992.

[40] C. Samson. *Contribution à la classification d'images satellitaires par approche variationnelle et équations aux dérivées partielles*. PhD thesis, Université de Nice Sophia Antipolis, September 2000.

[41] C. Samson, L. Blanc-Féraud, G. Aubert, and J. Zerubia. A level set method for image classification. *IJCV*, 40(3) :187–197, 2000.

[42] C. Samson, L. Blanc-Féraud, G. Aubert, and J. Zerubia. A variational model for image classification and restoration. *IEEE Transactions on Pattern Analyis and Machine Intelligence*, 22(5) :460–472, May 2000.

[43] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics*, 114 :146–159, 1994.

[44] G. Unal, A. Yezzi, and H. Krim. Information-theoretic active polygons for unsupervised texture segmentation, June 2002. preprint available upon request, submitted.

[45] L.A. Vese and T.F. Chan. A multiphase level set framework for image segmentation using the Mumford and Shah model. *International Journal of Computer Vision*, 50(3) :271–293, 2002.

[46] L.A. Vese and S.J. Osher. Modeling textures with total variation minimization and oscillating patterns in image processing. *Journal of Scientific Computing*, 19 :553–572, 2003.