

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE ROUEN
DÉPARTEMENT GÉNIE MATHÉMATIQUE



2017

Reconstruction d'images par approche $TV - TV^2$

Nathan ROUXELIN

Semestre 8
Rapport de projet semestriel
Encadré par Carole LE GUYADER

TABLE DES MATIÈRES

| | |
|--|-----------|
| Introduction | 2 |
| 1 Contexte mathématique | 4 |
| I Modèle de l'image dégradée | 4 |
| II Quelques régularisations pour le problème de débruitage | 5 |
| 1 Régularisation $H^1(\Omega)$ | 5 |
| 2 Régularisation de Rudin-Osher-Fatemi | 7 |
| 2 Modèle $TV - TV^2$ | 9 |
| I Espaces $BV(\Omega)$ et $BV^2(\Omega)$ | 9 |
| 1 Expression de la variation totale | 9 |
| 2 Espaces des fonctions à variation bornée | 10 |
| 3 Espace des fonctions à hessien borné | 11 |
| 4 Pourquoi travailler dans ces espaces? | 11 |
| II Modèle | 12 |
| III Résolution du problème de débruitage | 13 |
| 1 Itération de Bregman | 13 |
| 2 <i>Split Bregman</i> pour le débruitage | 14 |
| 3 Résolution du problème en u | 15 |
| 4 Résolution des problèmes en v et w | 16 |
| 5 Résultats numériques | 17 |
| IV Résolution du problème d'inpainting | 20 |
| 1 Méthode de Bregman | 20 |
| 2 Résultats numériques | 22 |
| 3 Travail informatique | 24 |
| I Programme réalisés | 24 |
| II La bibliothèque CImg | 24 |
| 1 Présentation | 24 |
| 2 Prise en main sur un exemple | 25 |
| III Implémentation de la méthode de débruitage | 27 |
| IV Implémentation de la méthode d'inpainting | 27 |
| Conclusion | 28 |

INTRODUCTION

Dans ce document nous nous intéressons à deux grands problèmes du traitement d'images : le débruitage et l'inpainting et à leur résolution par des méthodes variationnelles.

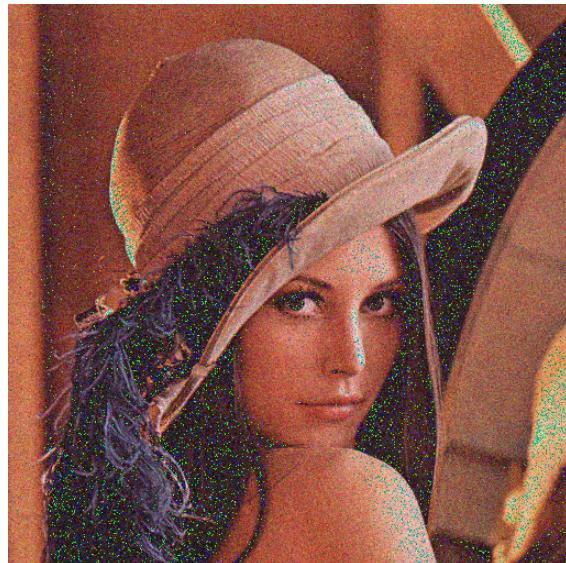


FIGURE 1 – Problème de débruitage

La première méthode débruitage est l'utilisation d'un filtre. Ce procédé est simple mais il a un inconvénient majeur : il lisse l'image. Nous obtenons donc une image débruitée mais floue. Les méthodes présentées dans ce document permettent de limiter grandement la création de flou.

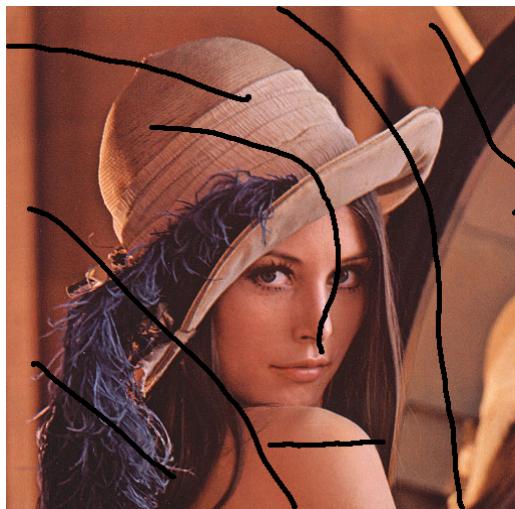


FIGURE 2 – Problème d’inpainting

Nous verrons que le modèle utilisé pour la résolution du problème de débruitage peut aussi être utilisé pour l’inpainting. La résolution de ces deux problèmes par les méthodes présentées ici est très similaires.

Les méthodes et résultats utilisés dans ce document proviennent principalement de deux sources :

- les travaux de doctorat de K. PAPAFITSOROS : *Novel higher order regularisation methods for image reconstruction*,
- le livre de M. BERGOUNIOUX : *Introduction au traitement mathématique des images*.

Notations utilisées :

La bibliothèque de manipulation d’images utilisées pour l’implémentation en C++ des différentes méthodes utilisant les coordonnées cartésiennes (x, y) plutôt que les coordonnées matricielles, nous avons fait le choix d’utiliser cette notation dans le document.

Afin de différencier la notation dans les cas continus et discrets, nous utiliserons les parenthèses pour les coordonnées continues et les crochets pour les coordonnées discrètes.

I Modèle de l'image dégradée

Considérons une image u_0 , un bruit additif v et un opérateur R . Les problèmes de reconstruction d'images consistent à retrouver u_0 à partir de l'observation dégradée

$$u_d := Ru_0 + v.$$

L'opérateur R peut représenter un flou¹ ou une indicatrice, dans le cas de l'inpainting. Dans le cadre du problème de débruitage pur, nous choisirons R tel que $Ru = u$, c'est-à-dire un opérateur de convolution de le noyau est une Dirac.

Pour approcher l'image originale, une approche classique est de considérer le problème

$$\inf_u \|u_d - Ru\|_{L^2(\Omega)}^2 \tag{1.1}$$

Cette minimisation apparaît lorsque l'on fait l'hypothèse du bruit gaussien : il s'agit alors de la solution du maximum de vraisemblance.

Le problème (1.1) n'est cependant pas satisfaisant : il est mal posé. Rappelons la définition d'un problème bien posé.

Définition 1.I.1 : PROBLÈME BIEN POSÉ AU SENS DE HADAMARD

On dit qu'un problème est **bien posé** (au sens de Hadamard) si il vérifie les trois conditions suivantes :

1. existence de la solution,
2. unicité de la solution,
3. stabilité de la solution (càd. dépendance continue de la solution aux données initiales).

Dans le cadre de la résolution de (1.1) :

- L'opérateur R n'est pas forcément inversible. L'existence ou l'unicité de la solution n'est alors pas garantie.
- Si R est inversible, le calcul numérique de son inverse est difficile. La stabilité de la solution n'est alors pas acquise.

1. Il s'agit alors d'un opérateur de convolution.

Il est possible d'obtenir un problème bien posé à partir de (1.1) en faisant une hypothèse a priori sur la solution u . Il faut bien noter l'inconvénient de cette méthode : on fait une hypothèse sur la régularité de u , on risque donc de détruire une partie des contours de l'image originale. En pratique, cela consiste à ajouter un terme de régularisation² au problème.

$$\inf_u \|u_d - Ru\|_{L^2(\Omega)}^2 + L(u)$$

Dans la prochaine partie, nous présenterons deux modèles de régularisation classiques pour le problème de débruitage.

II Quelques régularisations pour le problème de débruitage

Dans toute cette partie, on considère que $Ru = u$.

1 Régularisation $H^1(\Omega)$

Dans un premier temps, posons $V := H^1(\Omega)$. Nous rappelons également que la norme sur V est définie par

$$\forall u \in V, \|u\|_V^2 = \|u\|_{1,\Omega}^2 := \|u\|_{L^2(\Omega)}^2 + \|\nabla u\|_{L^2(\Omega)}^2.$$

Le problème de minimisation se réécrit

$$\min_{u \in V} \|u - u_d\|_{L^2(\Omega)}^2 \quad (1.2)$$

Il est facile de vérifier que la fonctionnelle $u \mapsto \mathcal{J}_0(u) := \|u - u_d\|_{L^2(\Omega)}^2$ n'est pas coercive sur V . En effet, soient

$$\Omega :=]0, 1[, u_n := x^n \text{ et } u_d := 0.$$

On a alors

$$\|u_n\|_{L^2(\Omega)} = \frac{1}{\sqrt{2n+1}} \text{ et } \|\nabla u_n\|_{L^2(\Omega)} = \frac{n}{\sqrt{2n-1}}.$$

D'où

$$\lim_{n \rightarrow \infty} \|u_n\|_{1,\Omega} = +\infty \text{ mais } \lim_{n \rightarrow \infty} \mathcal{J}_0(u_n) = 0.$$

Cela ne nous permet pas de conclure à l'existence d'une solution au problème (1.2).

Soit $\alpha > 0$. Considérons maintenant la fonctionnelle

$$\mathcal{J}_\alpha : u \mapsto \mathcal{J}_0(u) + \alpha \|\nabla u\|_{L^2(\Omega)}^2.$$

Nous allons montrer que le problème

$$\min_{u \in V} \mathcal{J}_\alpha(u) \quad (1.3)$$

est bien posé.

Il est facile de vérifier que \mathcal{J}_α est continue, coercive et convexe sur V . Nous pouvons donc conclure que le problème (1.3) admet au moins une solution dans V .

2. On parle aussi de pénalisation.

Pour la calculer, montrons dans un premier temps que \mathcal{J}_α est Gâteaux-différentiable sur V . Soient $u, v \in V$ et $\varepsilon > 0$,

$$\begin{aligned}\mathcal{J}_\alpha(u + \varepsilon v) - \mathcal{J}_\alpha(u) &= \varepsilon^2 \int_\Omega |v|^2 + |\nabla v|^2 dx + 2\varepsilon \int_\Omega (u - u_d)v + \nabla u \cdot \nabla v dx \\ \frac{\mathcal{J}_\alpha(u + \varepsilon v) - \mathcal{J}_\alpha(u)}{\varepsilon} &= \varepsilon \int_\Omega |v|^2 + |\nabla v|^2 dx + 2 \int_\Omega (u - u_d)v + \nabla u \cdot \nabla v dx \\ \lim_{\varepsilon \rightarrow 0^+} \frac{\mathcal{J}_\alpha(u + \varepsilon v) - \mathcal{J}_\alpha(u)}{\varepsilon} &= 2 \int_\Omega (u - u_d)v + \nabla u \cdot \nabla v dx \\ &= \mathcal{J}'_\alpha(u; v)\end{aligned}$$

Notons u^* un point de minimum de \mathcal{J}_α . D'après la condition d'Euler, on a

$$\forall v \in V, \quad \mathcal{J}'_\alpha(u^*; v) = 0.$$

En appliquant la formule de Green, on obtient

$$\forall v \in V, \quad \int_\Omega (u^* - u_d)v dx = \alpha \int_\Omega v \Delta u^* dx - \alpha \int_{\partial\Omega} v \frac{\partial u^*}{\partial n} d\sigma. \quad (1.4)$$

On reconnaît l'équation (1.4) comme étant la formulation variationnelle de l'EDP

$$\begin{cases} -\alpha \Delta u + u = u_d & \text{dans } \Omega \\ \frac{\partial u}{\partial n} = 0 & \text{sur } \partial\Omega \end{cases} \quad (1.5)$$

Remarque : Il est classique d'ajouter la condition de Neumann homogène pour simplifier les calculs.

D'après le théorème de Lax-Milgram, il existe donc un unique u^* solution (faible) de (1.3).

Le système (1.5) permet une implémentation facile de la régularisation $H^1(\Omega)$: il suffit de considérer l'équation dynamique suivante

$$\frac{\partial u}{\partial t} - \alpha \Delta u + u = u_d$$

et d'implémenter une discrétisation par différences finies³. Il faut alors ajouter la condition initiale

$$\forall x \in \Omega, \quad u(x, t=0) = u_d(x).$$

Cette équation étant un exemple classique de problème stable⁴, la problème de minimisation (1.3) est donc bien posé.

La régularisation $H^1(\Omega)$ présente un inconvénient majeur : la solution u^* obtenue est très régulière. En utilisant cette méthode, on lisse donc l'image et on perd beaucoup d'information sur les contours. Nous allons donc étudier d'autres méthodes de régularisation plus adaptées au traitement d'images.

3. Cela revient en fait à utiliser une méthode de descente de gradient.

4. Avec les conditions aux limites et la condition initiale.



FIGURE 1.1 – Image dégradée



FIGURE 1.2 – Image restaurée

2 Régularisation de Rudin-Osher-Fatemi

La régularisation $H^1(\Omega)$ étant trop brutale pour les applications en traitement d'images, nous allons nous concentrer sur un autre modèle de régularisation : le modèle ROF. C'est un des premiers modèles utilisant le concept de variation totale en traitement d'images.

Le problème de minimisation considéré ici est

$$\min_{u \in BV(\Omega)} \|u - u_d\|_{L^2(\Omega)}^2 + \frac{\lambda}{2} TV(u). \quad (1.6)$$

Nous définirons rigoureusement les termes « $BV(\Omega)$ » et « $TV(u)$ » dans le chapitre 2.I. Pour cette introduction, il suffit de considérer que $BV(\Omega)$ est l'espace des fonctions «à variation bornée» et que cette variation peut être mesurée par la semi-norme TV . Pour des fonctions suffisamment régulières, on a

$$TV(u) = \int_{\Omega} |\nabla u(x)| dx.$$

Nous ne détaillerons pas ce modèle ici, mais il semblait essentiel d'évoquer le plus célèbre des modèles basé sur la variation totale.

CHAPITRE 2

MODÈLE $TV - TV^2$

I Espaces $BV(\Omega)$ et $BV^2(\Omega)$

1 Expression de la variation totale

La définition de la variation totale d'une fonction

$$TV(f) := \sup \left\{ \int_{\Omega} f(x) \operatorname{div} \varphi(x) dx \mid \varphi \in \mathcal{C}_c^1(\Omega, \mathbb{R}^2), \|\varphi\|_{\infty} \leq 1 \right\} \quad (2.1)$$

n'étant pas naturelle, nous allons tenter de mieux la comprendre.

Considérons tout d'abord une fonction $f : [a, b] \rightarrow \mathbb{R}$. Soit p une partition de $[a, b]$, c'est-à-dire

$$p := (x_i)_{i=0}^n \quad \text{avec } a =: x_0 < x_1 < \dots < x_n =: b.$$

Notons \mathcal{P} l'ensemble des partitions de $[a, b]$. La variation de f sur la partition p est donnée par

$$V_p(f) := \sum_{i=0}^n |f(x_{i+1}) - f(x_i)|.$$

Nous pouvons alors définir la **variation totale** de f de la manière suivante

$$TV(f) := \sup_{p \in \mathcal{P}} V_p(f). \quad (2.2)$$

Si f est dérivable, la relation (2.2) devient

$$TV(f) = \int_a^b |f'(x)| dx. \quad (2.3)$$

Revenons maintenant au cas multidimensionnel. Soit $f : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}$. La relation (2.3) se généralise naturellement (lorsque les calculs ont un sens)

$$TV(f) = \int_{\Omega} |\nabla f(x)| dx \quad (2.4)$$

où $|\cdot|$ désigne la norme euclidienne de \mathbb{R}^2 . Cette écriture nécessite de faire des hypothèses sur le gradient de f . Nous allons donc chercher à la relaxer. Il faut, bien sûr, que la relaxation coïncide avec (2.4) lorsque celle-ci est définie.

Attention : Dans la suite, le but est de comprendre ce qu'il se passe. Les calculs ne sont donc pas tous rigoureux. De même, la notion de «convergence» utilisée n'est pas bien définie.

Supposons que $f \in W^{1,1}(\Omega)$.

Soit $\varphi := (\varphi_1, \varphi_2) \in C_c^1(\Omega, \mathbb{R}^2)$ telle que $\|\varphi\|_\infty \leq 1$. On rappelle que

$$\forall x := (x_1, x_2) \in \Omega, \quad \operatorname{div}\varphi(x) = \frac{\partial \varphi_1}{\partial x_1}(x) + \frac{\partial \varphi_2}{\partial x_2}(x).$$

Ainsi, on a¹

$$\begin{aligned} \int_{\Omega} f(x) \operatorname{div}\varphi(x) dx &= \int_{\Omega} f(x) \left(\frac{\partial \varphi_1}{\partial x_1}(x) + \frac{\partial \varphi_2}{\partial x_2}(x) \right) dx \\ (\text{IPP}) \quad &= - \int_{\Omega} \left(\frac{\partial f}{\partial x_1}(x) \varphi_1(x) + \frac{\partial f}{\partial x_2}(x) \varphi_2(x) \right) dx \\ &= - \int_{\Omega} \nabla f \cdot \varphi dx \end{aligned}$$

où \cdot désigne le produit scalaire usuel de \mathbb{R}^2 . Cela permet d'établir la majoration suivante

$$\int_{\Omega} f(x) \operatorname{div}\varphi(x) dx = - \int_{\Omega} \nabla f \cdot \varphi dx \leq \int_{\Omega} |\nabla f| dx \quad (2.5)$$

car $\|\varphi\|_\infty \leq 1$.

Par densité de $C_c^1(\Omega, \mathbb{R}^2)$ dans $L^1(\Omega, \mathbb{R}^2)$, il existe une suite $(\varphi_n)_n$ telle que

$$\lim_{n \rightarrow \infty} \varphi_n = - \frac{\nabla f}{|\nabla f|}.$$

On a alors

$$\begin{aligned} \lim_{n \rightarrow \infty} \int_{\Omega} f(x) \operatorname{div}\varphi_n(x) dx &= \lim_{n \rightarrow \infty} \int_{\Omega} \frac{\nabla f \cdot \nabla f}{|\nabla f|} dx \\ &= \int_{\Omega} |\nabla f| dx. \end{aligned}$$

Il faudrait, bien sûr, justifier avec soin l'utilisation du théorème de convergence dominée dans ce calcul.

On a donc, d'après (2.5),

$$\sup_{\varphi} \int_{\Omega} f(x) \operatorname{div}\varphi(x) dx = \int_{\Omega} |\nabla f(x)| dx.$$

La définition de la variation totale semble donc pertinente.

2 Espaces des fonctions à variation bornée

On définit l'espace des fonctions à variations bornées

1. Dans le cas général, on démontre ce résultat en utilisant le théorème de Green-Ostrogradsky.

Définition 2.I.1 : FONCTIONS À VARIATIONS BORNÉES

On dit que $f \in L^1(\Omega)$ est à **variations bornées**, ou encore $f \in BV(\Omega)$, si

$$TV(f) < +\infty.$$

TV est définie par la relation (2.1).

Théorème 2.I.1 :

Muni de la norme

$$\|\cdot\|_{BV} := \|\cdot\|_{L^1(\Omega)} + TV(\cdot),$$

l'espace $BV(\Omega)$ est un espace de Banach.

Dans le cadre du travail numérique de nous allons effectuer, nous utiliserons des fonctions $u \in BV(\Omega)$ vérifiant

$$TV(u) = \int_{\Omega} |\nabla u(x)| dx.$$

3 Espace des fonctions à hessien borné

Définition 2.I.2 : FONCTIONS À HESSIEN BORNÉ

On dit que $f \in W^{1,1}(\Omega)$ est à **hessien borné**, ou encore que $f \in BV^2(\Omega)$ si

$$\nabla f \in BV(\Omega).$$

Comme précédemment, on peut définir une «variation totale seconde» de f . Elle s'exprime ainsi

$$TV^2(f) := \sup \left\{ \int_{\Omega} f(x) \operatorname{div}^2 \varphi(x) dx \mid \varphi \in \mathcal{C}_c^1(\Omega, \mathbb{R}^2), \|\varphi\|_{\infty} \leq 1 \right\}.$$

Quand cela a un sens, on a

$$TV(f) = \int_{\Omega} |\nabla^2 f| dx.$$

Nous travaillerons uniquement avec des fonctions vérifiant cette dernière propriété.

Théorème 2.I.2 :

Muni de la norme

$$\|\cdot\|_{BV^2(\Omega)} := \|\cdot\|_{BV(\Omega)} + TV^2(\cdot),$$

l'espace $BV^2(\Omega)$ est un espace de Banach.

4 Pourquoi travailler dans ces espaces ?

Dans les cadres du traitement d'images, il faut trouver un compromis sur la régularité de nos images. Si on travaille dans un espace de fonctions trop régulières, on obtient des images lissées, c'est-à-dire floues. Nous avons mis ce phénomène en évidence en travaillant sur la régularisation $H^1(\Omega)$. À l'inverse, si les fonctions de notre espace ne sont pas assez régulières, on obtient des phénomènes oscillatoires. Ces résultats ne sont pas non plus satisfaisants.

Nous voulons des images qui présentent des discontinuités, c'est-à-dire des contours marqués, mais nous ne voulons pas d'oscillations. Les espaces choisis semblent donc adéquat. En effet,

- les fonctions $BV(\Omega)$ obtenues sont constantes par morceaux ;
- les fonctions $BV^2(\Omega)$ obtenues sont affines par morceaux.

Il y a encore beaucoup à dire sur ces espaces d'un point de vue plus mathématique. Les différentes topologies qu'on peut y définir sont très intéressantes. De même, il faudrait évoquer les résultats liant les variations totales et les mesures de Radon finie. Malheureusement ce n'est pas l'objet de ce travail. Nous retiendrons tout de même leur structure d'espace de Banach qui permet d'utiliser les grands résultats de l'optimisation.

II Modèle

Dans le cadre de l'approche $TV - TV^2$, le modèle sur lequel nous allons travailler fait apparaître des termes de régularisation portant sur les dérivées d'ordre 1 et d'ordre 2 de l'image. Plus précisément, le modèle s'écrit

$$\min_{u \in BV^2(\Omega)} \frac{1}{2} \int_{\Omega} |Ru - f|^2 dx + \alpha TV(u) + \beta TV^2(u). \quad (2.6)$$

À partir de maintenant, nous n'allons plus travailler sur un problème continu, mais nous allons rendre ce problème discret.

Pour des raisons que nous détaillerons plus tard, nous allons périodiser nos images.

Afin de rendre le problème discret, nous devons définir des opérateurs «différences fines».

Pour le gradient, nous utiliserons un schéma *forward*. On a

$$\nabla u[x, y] := (D_x u[x, y], D_y u[x, y])^T$$

avec

$$\begin{aligned} D_x u[x, y] &:= u[x+1, y] - u[x, y] \\ D_y u[x, y] &:= u[x, y+1] - u[x, y]. \end{aligned}$$

Nous aurons également besoin des opérateurs *backwards*

$$\begin{aligned} \overleftarrow{D}_x u[x, y] &:= u[x, y] - u[x-1, y] \\ \overleftarrow{D}_y u[x, y] &:= u[x, y] - u[x, y-1]. \end{aligned}$$

Pour le hessien, on utilisera les opérateurs centrés et la convention $D_{xy} = D_{yx}$.

$$\begin{aligned} D_{xx} u[x, y] &:= u[x+1, y] - 2u[x, y] + u[x-1, y] \\ D_{yy} u[x, y] &:= u[x, y+1] - 2u[x, y] + u[x, y-1] \\ D_{xy} u[x, y] &:= u[x, y] - u[x+1, y] - u[x, y+1] + u[x+1, y+1] \\ \overleftarrow{D}_{xy} u[x, y] &:= u[x, y] - u[x-1, y] - u[x, y-1] + u[x-1, y-1] \end{aligned}$$

Notre problème discret s'écrit donc

$$\min_{u \in \mathcal{M}_{NM}(\mathbb{R})} \frac{1}{2} \|u - u_d\|_2^2 + \alpha \|\nabla u\|_1 + \beta \|\nabla^2 u\|_1. \quad (2.7)$$

Pourquoi combiner les variations totales premières et secondes ?

En jouant sur les paramètres α et β , on peut obtenir un compromis entre les propriétés des régularisations TV pures et TV^2 pures.

Par exemple, dans le cadre du problème d'inpainting la régularisation TV^2 permet de rendre connexe certaines images présentant un «trou» assez large, mais elle a tendance à générer du flou. La régularisation TV ne possède pas ces propriétés de connexité, mais elle ne génère pas de flou. En combinant les deux, on tent d'obtenir la connexité et peu de flou.

III Résolution du problème de débruitage

Pour résoudre le problème (2.7), nous allons utiliser la méthode de Bregman.

1 Itération de Bregman

Considérons un problème de minimisation avec contraintes sous forme standard

$$\min \{E(u) \mid Au = b\}. \quad (2.8)$$

On suppose que la fonction coût E est convexe et que la contrainte A est linéaire. Il est possible de transformer le problème (2.8) en un problème sans contraintes de la forme

$$\sup_{\lambda} \min_u E(u) + \frac{\lambda}{2} \|Au - b\|_2^2. \quad (2.9)$$

Dans ce problème, la contrainte est représentée par un terme d'attache aux données. Elle est satisfaite lorsque λ tend vers l'infini.

En pratique, il n'est possible de faire tendre λ vers l'infini. Nous allons donc utiliser un algorithme itératif, nommé «Itération de Bregman» pour résoudre le problème (2.9).

Itération de Bregman

$$u_{n+1} = \arg \min_u E(u) + \frac{\lambda}{2} \|Au - b_n\|_2^2$$

$$b_{n+1} = b_n + (b - Au_{n+1})$$

Il est possible de démontrer les théorèmes suivants :

Théorème 2.III.0 :

Supposons que la première étape de l'itération de Bregman ait une unique solution pour tout $n \in \mathbb{N}$. On a

1. $\exists C_1 > 0, \forall n \geq 2, \|Au_n - b\|_2^2 < \frac{C_1}{n-1}$
2. $\forall n \in \mathbb{N}, \|Au_{n+1} - b\|_2^2 \leq \|Au_n - b\|_2^2$
3. $\sum_{n=0}^{\infty} \|Au_n - b\|_2^2 < \infty$
4. $\exists C_2 > 0, \forall n \in \mathbb{N}, E(u_n) < C_2$.

Théorème 2.III.0 :

L'itération de Bregman converge vers une solution optimale du problème (2.9) dans les cas suivants :

1. si la contrainte est satisfaite en un nombre fini d'itérations;
2. si la fonctionnelle E est coercive.

2 Split Bregman pour le débruitage

Nous allons transformer le problème (2.7) en un problème de minimisation avec contraintes.

$$\min_{u,v,w} \left\{ \frac{1}{2} \|u - u_d\|_2^2 + \alpha \|v\|_1 + \beta \|w\|_1 \mid v = \nabla u, w = \nabla^2 u \right\} \quad (2.10)$$

Pour alléger l'écriture, nous n'écrirons pas les espaces de matrices dans lesquels nous travaillons. Il faut bien les préciser au moins une fois, les voici donc

$$u \in \mathcal{M}_{NM}(\mathbb{R}), \quad v \in (\mathcal{M}_{NM}(\mathbb{R}))^2, \quad w \in (\mathcal{M}_{NM}(\mathbb{R}))^4.$$

Les opérateurs gradient et hessien discrets étant linéaires, on peut appliquer l'itération de Bregman au problème (2.10).

Nous obtenons l'algorithme suivant.

Itération de Bregman

$$\begin{aligned} (u_{n+1}, v_{n+1}, w_{n+1}) &= \arg \min_{u,v,w} \frac{1}{2} \|u - u_d\|_2^2 + \alpha \|v\|_1 + \beta \|w\|_1 \\ &\quad + \frac{\lambda}{2} \|b_{1,n} + \nabla u - v\|_2^2 \\ &\quad + \frac{\lambda}{2} \|b_{2,n} + \nabla^2 u - w\|_2^2 \\ b_{1,n+1} &= b_{1,n} + \nabla u_{n+1} - v_{n+1} \\ b_{2,n+1} &= b_{2,n} + \nabla^2 u_{n+1} - w_{n+1} \end{aligned}$$

les deux dernières sont explicites, il reste à résoudre le premier problème. Pour cela, on va le découper² en trois sous-problèmes selon les variables.

Voici l'algorithme obtenu.

Split Bregman – Débruitage

$$\begin{aligned} u_{n+1} &= \arg \min_u \frac{1}{2} \|u - u_d\|_2^2 + \frac{\lambda_1}{2} \|b_{1,n} + \nabla u - v_n\|_2^2 + \frac{\lambda_2}{2} \|b_{2,n} + \nabla^2 u - w_n\|_2^2 \\ v_{n+1} &= \arg \min_v \alpha \|v\|_1 + \frac{\lambda_1}{2} \|b_{1,n} + \nabla u - v_n\|_2^2 \\ w_{n+1} &= \arg \min_w \beta \|w\|_1 + \frac{\lambda_2}{2} \|b_{2,n} + \nabla^2 u - w_n\|_2^2 \\ b_{1,n+1} &= b_{1,n} + \nabla u_{n+1} - v_{n+1} \\ b_{2,n+1} &= b_{2,n} + \nabla^2 u_{n+1} - w_{n+1} \end{aligned}$$

2. D'où le nom de *Split Bregman*

Les trois problèmes de minimisation sont plutôt simple à résoudre. On remarque déjà que les problèmes en v et w ont la même forme, ils vont donc se résoudre avec la même méthode.

3 Résolution du problème en u

Le problème en u est le plus compliqué à résoudre : il faut passer par les équations d'Euler-Lagrange et la condition d'optimalité obtenue est une équation aux dérivées partielles d'ordre 4.

Dans le cadre discret, elle s'exprime ainsi

$$\begin{aligned} & u_{n+1} - \lambda_1 (D_{xx}(u_{n+1}) + D_{yy}(u_{n+1})) \\ & + \lambda_2 (D_{4x}(u_{n+1}) + D_{4y}(u_{n+1}) + 2D_{2x2y}(u_{n+1})) \\ & = u_d + \lambda_1 (D_x(b_{1,n}^x - v_n^x) + D_y(b_{1,n}^y - v_n^y)) \\ & - \lambda_2 (D_{xx}(b_{2,n}^{xx} - w_n^{xx}) + D_{yy}(b_{2,n}^{yy} - w_n^{yy}) + 2D_{xy}(b_{2,n}^{xy} - w_n^{xy})). \end{aligned}$$

Il semble donc difficile de résoudre ce problème en s'attaquant directement à cette équation. Nous allons présenter deux méthodes résolution.

Résolution par la méthode de Gauss-Seidel

En réorganisant la condition d'optimalité, on obtient un système à diagonale strictement dominante³. On peut donc appliquer la méthode de Gauss-Seidel.

La méthode de Gauss-Seidel est convergente, mais la convergence est plutôt lente dans ce cas. Nous allons donc voir une autre méthode plus rapide (environ 10 fois plus rapide empiriquement).

Résolution par transformée de Fourier

Il est assez classique, en imagerie, de voir les opérateurs différentiels comme des filtres. On sait également que les produits de convolution sont représentés par des produits classiques dans l'espace de Fourier. C'est pour exploiter les propriétés de la transformation de Fourier que nous devons rendre notre image périodique.

Par transformée de Fourier, et en réorganisant, on obtient

$$FD \cdot \mathcal{F}(u_{n+1}) = \mathcal{F}(R).$$

On peut alors résoudre notre problème de minimisation grâce à la formule

$$u_{n+1} = \mathcal{F}^{-1}(\mathcal{F}(R)/FD)$$

où «/» désigne la division terme-à-terme des matrices et FD la transformée de Fourier du facteur dans le membre de gauche.

3. Je ne l'ai pas retrouvé dans mes brouillons, je ne peux donc pas l'écrire explicitement.

Pour pouvoir appliquer cette formule, il faut calculer les noyaux de convolution associés aux opérateurs différentiels. Ce calcul est assez simple, nous allons le faire pour l'opérateur D_{xx} . Notons h ce noyau.

On sait que

$$D_{xx}u[x, y] = u[x + 1, y] - 2u[x, y] + u[x - 1, y].$$

Écrivons le produit de convolution en deux dimensions

$$D_{xx}u[x, y] = \sum_{k=0}^N \sum_{\ell=0}^M h[k, \ell]u[x - k, y - \ell].$$

Par identification les $h[k, \ell]$ sont tous nuls sauf

$$\begin{aligned} h[-1, 0] &= 1 \\ h[0, 0] &= -2 \\ h[1, 0] &= 1. \end{aligned}$$

Les autres noyaux de convolution s'obtiennent de la même manière.

4 Résolution des problèmes en v et w

Ces deux problèmes se résolvent par la méthode du seuillage doux. Nous allons justifier cette méthode dans le cas scalaire. Son extension au cas vectoriel est naturelle.

Soient $\varphi(x) := \lambda|x|$ et $F(x) := \frac{1}{2}(x - y)^2 + \varphi(x)$. Le paramètre y est supposé connu. On chercher à résoudre le problème

$$\min_{x \in \mathbb{R}} F(x).$$

À l'optimum, on a

$$\begin{aligned} F'(x) = 0 \iff y &= x + \varphi'(x) \\ &= x + \lambda \text{signe}(x). \end{aligned}$$

Il est donc facile de tracer y en fonction de x .

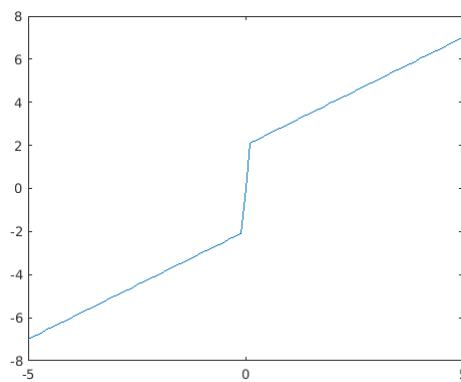


FIGURE 2.1 – $y = x + \varphi'(x)$

Pour obtenir x en fonction de y , il suffit d'échanger les axes.

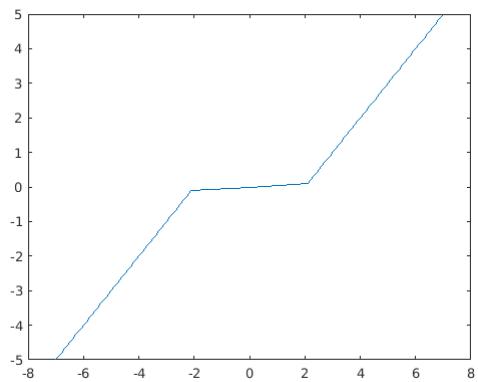


FIGURE 2.2 – $x^*(y)$

Le point de minimum de F est donc

$$x^* := \max(|y| - \lambda, 0) \operatorname{signe}(y).$$

5 Résultats numériques

Les résultats numériques obtenus pour le débruitage sont plutôt satisfaisants. Voici quelques exemples.

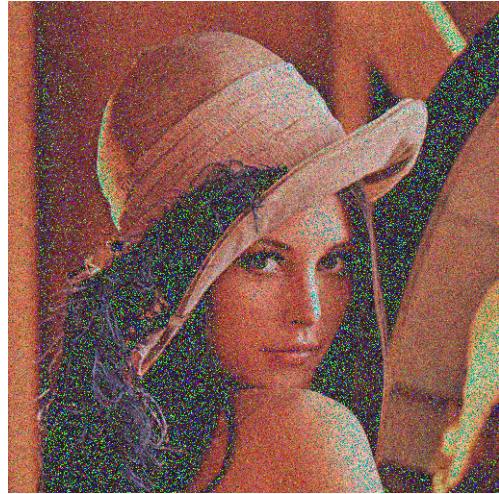


FIGURE 2.3 – Image dégradée – Bruit gaussien



FIGURE 2.4 – Image restaurée

Avec du bruit «poivre et sel», on obtient une amélioration de l'image. Cependant le résultat obtenu ne justifie pas le sur-coût de calcul par rapport à l'utilisation d'un filtre «médiane».

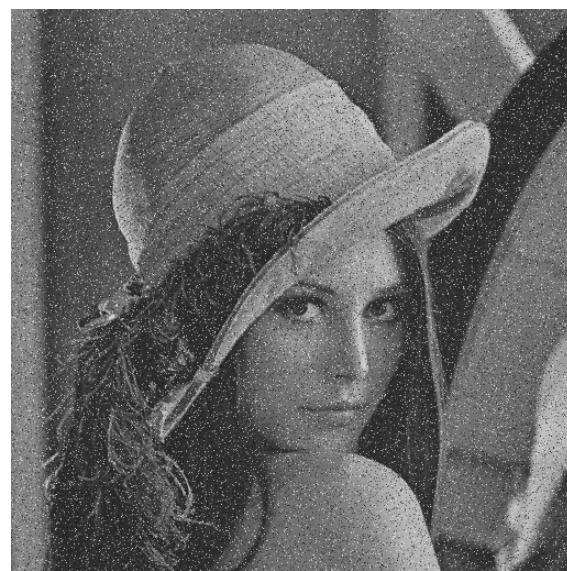


FIGURE 2.5 – Image dégradée – Bruit poivre et sel



FIGURE 2.6 – Image restaurée – Bruit poivre et sel

Avec du bruit de Laplace, on reconstruit bien l'image. Toutefois un léger flou apparaît sur l'image reconstruite.



FIGURE 2.7 – Image dégradée – Bruit de Laplace



FIGURE 2.8 – Image restaurée – Bruit de Laplace

IV Résolution du problème d'inpainting

1 Méthode de Bregman

Considérons une image en niveaux de gris

$$u : \Omega \rightarrow \mathbb{R},$$

et un domaine $D \subset \Omega$. Nous supposons connaître parfaitement⁴ l'image sur $\Omega \setminus D$. Nous n'avons par contre aucune information sur u dans le domaine D . On cherche à «remplir» au mieux ce domaine. Plus formellement, cela signifie que notre observation est de la forme

$$u_d = \mathbb{1}_{\Omega \setminus D} u_0.$$

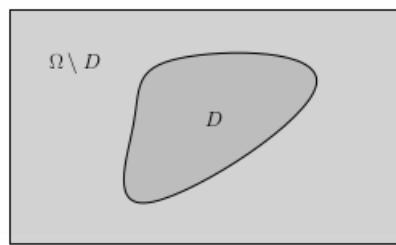


FIGURE 2.9 – Problème d'inpainting

Pour résoudre le problème d'inpainting, nous utiliserons un modèle très similaire à celui utilisé pour résoudre le problème de débruitage

$$\min_{u \in BV^2(\Omega)} \frac{1}{2} \|\mathbb{1}_{\Omega \setminus D} u - u_d\|_{L^2(\Omega)}^2 + \alpha TV(u) + \beta TV^2(u). \quad (2.11)$$

Cette formulation permet de conserver les données dans les zones connues et de «remplir» le domaine D .

4. C'est-à-dire qu'il n'y a ni bruit ni flou.

Il n'est toutefois pas possible d'utiliser exactement la même méthode de résolution numérique. En effet, on a

$$\mathcal{F}(\mathbb{1}_{\Omega \setminus D} u) \neq \mathcal{F}(u) \mathbb{1}_{\Omega \setminus D}.$$

Cela ne permet pas de conserver les données du domaine connue.

Il est en fait assez simple de contourner ce problème : il suffit d'ajouter une variable \tilde{u} supplémentaire vérifiant la condition $\tilde{u} = u$ et on sépare le premier problème en u et en \tilde{u} .

On obtient l'algorithme de résolution suivant pour le problème d'inpainting.

| <u>Split Bregman – Inpainting</u> | |
|-----------------------------------|---|
| u_{n+1} | $= \arg \min_u \frac{1}{2} \ \mathbb{1}_{\Omega \setminus D} u - u_d\ _2^2 + \frac{\lambda_0}{2} \ b_{0,n} + \tilde{u}_n - u\ _2^2$ |
| \tilde{u}_{n+1} | $= \arg \min_{\tilde{u}} \frac{\lambda_0}{2} \ b_{0,n} + \tilde{u} + u_{n+1}\ _2^2 + \frac{\lambda_1}{2} \ b_{1,n} + \nabla \tilde{u} - v_n\ _2^2 + \frac{\lambda_2}{2} \ b_{2,n} + \nabla^2 \tilde{u} - w_n\ _2^2$ |
| v_{n+1} | $= \arg \min_v \alpha \ v\ _1 + \frac{\lambda_1}{2} \ b_{1,n} + \nabla u - v_n\ _2^2$ |
| w_{n+1} | $= \arg \min_w \beta \ w\ _1 + \frac{\lambda_2}{2} \ b_{2,n} + \nabla^2 u - w_n\ _2^2$ |
| $b_{0,n+1}$ | $= b_{0,n} + \tilde{u}_{n+1} - u_{n+1}$ |
| $b_{1,n+1}$ | $= b_{1,n} + \nabla u_{n+1} - v_{n+1}$ |
| $b_{2,n+1}$ | $= b_{2,n} + \nabla^2 u_{n+1} - w_{n+1}$ |

Tous les problèmes se résolvent comme dans le cas du débruitage, sauf le problème en u que nous allons détailler.

Résolution du problème en u

En écrivant explicitement le problème, on obtient

$$u_{n+1} = \arg \min_u \sum_{x,y} \mathbb{1}_{\Omega \setminus D} (u[x, y] - u_d[x, y])^2 + \frac{\lambda_0}{2} (b_{0,n}[x, y] + \tilde{u}_n[x, y] - u[x, y])^2.$$

On peut séparer ce problème et le résoudre pour chaque point $[x, y]$, on a alors

$$u_{n+1}[x, y] = \arg \min_{z \in \mathbb{R}} \sum_{x,y} \mathbb{1}_{\Omega \setminus D} (z - u_d[x, y])^2 + \frac{\lambda_0}{2} (b_{0,n}[x, y] + \tilde{u}_n[x, y] - z)^2.$$

La solution de ce problème est triviale (il s'agit de minimiser un polynôme de degré 2 à une seule variable).

$$u_{n+1}[x, y] = \left(\frac{2\mathbb{1}_{\Omega \setminus D}[x, y]}{\lambda_0 + 2} \right) u_d[x, y] + \left(\frac{\lambda_0 + 2(1 - \mathbb{1}_{\Omega \setminus D}[x, y])}{\lambda_0 + 2} \right) (b_{0,n}[x, y] + \tilde{u}_n[x, y]).$$

2 Résultats numériques

Les résultats obtenus dans le cadre du problème d'inpainting ne sont pas très satisfaisants. Ils sont beaucoup moins bons que les exemple présentés dans l'article original.

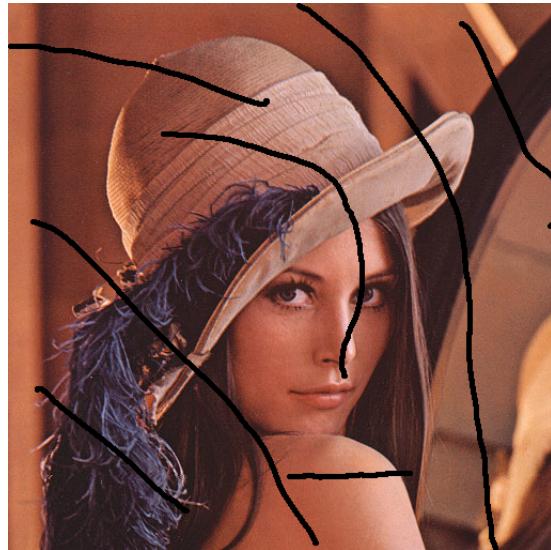


FIGURE 2.10 – Image dégradée

La figure suivante a été obtenue en 1000 itérations.

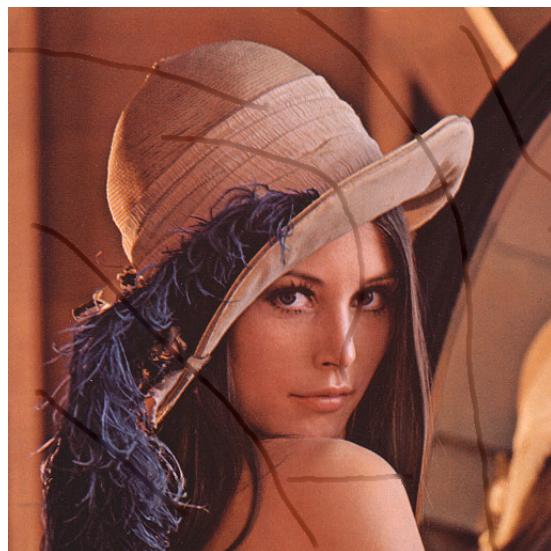


FIGURE 2.11 – Image restaurée

Toutefois certains paramètres peuvent présenter des propriétés intéressantes. Ainsi dans la figure suivante, on a réussi à reconstruire la texture du chapeau.



FIGURE 2.12 – Caption

CHAPITRE 3

TRAVAIL INFORMATIQUE

I Programme réalisés

Tous les programmes réalisés lors de ce travail sont disponible à l'adresse suivante :
<https://github.com/nrouxelin/tv-tv2-2017>.

Ils sont fournis avec un fichier README qui détaille la procédure d'utilisation et avec un Makefile qui permet de faciliter la compilation.

Ce Makefile a été écrit pour une compilation sous Linux (il faut lier les bibliothèques graphiques du système pour générer un affichage). Pour compiler sur d'autres plateformes, voir ce document :

http://cimg.eu/reference/group__cimg__overview.html.

II La bibliothèque CImg

1 Présentation

La bibliothèque *CImg* est une bibliothèque libre de manipulation d'images écrite en C++. Elle est développée par l'équipe IMAGE du laboratoire GREYC de Caen.

Son avantage principal est sa légèreté : toute la bibliothèque tient dans un seul fichier header (.h). Il est donc très facile de l'utiliser. Ce format (un seul fichier) a toutefois un inconvénient : il faut compiler la bibliothèque. Le temps de compilation du programme est donc plus important que lors de l'usage de bibliothèque plus classiques, c'est-à-dire précompilées.

Cette bibliothèque est écrite en C++, elle est donc organisée selon le principe de la programmation orientée objets. Elle définit trois classes :

- la classe **CImg** qui permet de représenter une image,
- la classe **CImgList** qui permet de représenter un tableau d'images,
- la classe **CImgDisplay** qui permet de représenter un affichage (fenêtre à l'écran).

Cette bibliothèque est libre et elle est disponible ici <http://cimg.eu/>.

2 Prise en main sur un exemple

Dans cette partie, on se propose d'illustrer quelques fonctionnalités de la bibliothèque *CImg* en implémentant un programme simple : la régularisation $H^1(\Omega)$ pour le problème de débruitage. Cela nous permettra de mettre en avant la simplicité d'utilisation de cette bibliothèque. Les codes écrits dans le cadre ce projet n'utilisent presque pas d'autres fonctionnalités, ils sont simplement plus longs ce qui les rend compliqué à décrire ici.

Pour implémenter cette méthode, nous effectuons une discréétisation par différences finies du problème aux limites suivant.

$$\begin{cases} \frac{\partial u}{\partial t} - \alpha \Delta u + u = u_d \text{ dans } \Omega \\ \frac{\partial u}{\partial n} = 0 \text{ sur } \partial\Omega \\ u = u_d \text{ en } t = 0 \end{cases}$$

Structure du programme

Notre programme suit la structure basique des programmes en C++. Il faut simplement importer la bibliothèque et utiliser le *namespace* associé.

```
#include <iostream>
#include "CImg.h"
using namespace cimg_library;
using namespace std;

int main(){
    //Notre code ici

    return 0;
}
```

Listing 3.1 – Structure du programme

Pour alléger ce document, nous n'inclurons pas ces lignes dans les futurs extraits de code.

Ouverture d'une image

On commence par ouvrir une image. Nous faisons quelques traitements sur cette image (ajout de bruit, passage en niveaux de gris). Enfin, on copie notre image de départ pour initialiser notre schéma.

```
string filename("img/input/lena.bmp");//Nom du fichier
CImg<float> ud(filename.c_str());//Cr ation de l'image
ud.noise(30);//Ajout de bruit
ud = ud.get_RGBtoYCbCr().get_channel(0);//RGB vers niveaux de gris
CImg<float> u(ud);//On copie l'image observ e

//Creation d'un affichage
CImgDisplay disp(u,"Regularisation\u21d3H^1",0,false,false);
```

Listing 3.2 – Préparation des images

Nous avons également créé un affichage qui nous permettra de suivre l'évolution de nos calculs en temps réel.

Création de quelques paramètres du schéma

Nous créons quelques variables utiles pour mettre en œuvre notre schéma de discréétisation.

```
//Parametres
float a = 5.0;
float dt = 0.01;
int t = 0;

//Dimensions de l'Image
int M = u.height(); //Largeur
int N = u.width(); //Longueur

//Couleur blanc (utile pour l'affichage du nombre d'iterations)
const float white[] = { 255, 255, 255 };
```

Listing 3.3 – Paramètres du programme

Parcours de l'image et mise en œuvre du schéma

```
while(!disp.is_closed()){
    t++;

    cimg_forXYC(u,x,y,c){
        //Interieur de l'image
        if(x>1 && x<N-1 && y>1 && y<M-1){
            u(x,y,c) += dt*(ud(x,y,c)-u(x,y,c))
                +(a*dt)*(u(x+1,y,c)-4.0*u(x,y,c)+u(x-1,y,c)+u(x,y+1,c)+u(x,y-1,c));
        }else{
            //Conditions de Neumann
            if(x==0)
                u(x,y,c) = u(1,y,c);
            else if(x==N-1)
                u(x,y,c) = u(N-2,y,c);
            if(y==0)
                u(x,y,c) = u(x,1,c);
            else if(y==M-1)
                u(x,y,c) = u(x,M-2,c);
        }
    }

    cimg_library::CIImg<>(u).draw_text(2,2,"iter=",
        ,0,1,13,t).display(disp.wait(100));
}
cimg_library::CIImg<>(u).save("img/output/h1.png",t);
```

Listing 3.4 – mise en œuvre du schéma

Nous n'avons pas implémenté de critère d'arrêt. La boucle while s'arrête lorsque la fenêtre d'affichage est fermée.

L'instruction `cimg_forXYC` permet de parcourir une image en deux dimensions. Les variables `x` et `y` représentent les coordonnées spatiales et la variable `c` représente le canal (R, G ou B par exemple). Les bornes à utiliser pour ces valeurs sont déterminées dynamiquement à partir des paramètres de l'instance, c'est-à-dire en fonction de l'image utilisée.

Il est également possible de définir un parcours par voisinage plutôt que pixel par pixel. Nous n'avons pas utilisé cette possibilité ici.

À la fin du programme, on enregistre l'image obtenue.

La bibliothèque *CImg* étant très riche, il est bien sûr impossible de la décrire exhaustivement ici. On renvoie donc à la documentation (<http://cimg.eu/reference/index.html>) pour plus de détails.

III Implémentation de la méthode de débruitage

Pour résoudre le problème de débruitage, nous avons écrit la classe `BregmanDenoiser`. Pour exécuter le programme de débruitage, il suffit d'appeler la méthode `BregmanDenoiser::solve`.

Les attributs et les méthodes de cette classe sont conformes aux notations de l'article original.

La résolution de chaque sous-problèmes est traitée dans une fonction différente. Cela permet de rendre le programme plus conforme à l'article, mais ce n'est pas optimal. En effet, il est possible de réduire le nombre de parcours de l'image (et donc la complexité en temps) en résolvant simultanément les sous problèmes 3 et 4.

La méthode `solve_subproblem1_GS` permet d'utiliser la méthode de résolution de Gauss-Seidle plutôt que la méthode utilisant la transformation de Fourier rapide.

Pour compiler ce programme, on utilise l'instruction `make denoiser`.

IV Implémentation de la méthode d'inpainting

L'implémentation de la résolution du problème d'inpainting est très similaire à celle du problème de débruitage. La classe `BregmanInpainter` est donc très semblable.

Dans la version actuelle du code, il y a beaucoup de redondances entre ces deux classes. Il est possible d'améliorer cela en utilisant la notion d'héritage. Une réorganisation du code est donc prévue afin de régler cela.

Il est possible de compiler ce programme en utilisant la commande `make inpainter`.

CONCLUSION

Les résultats obtenus dans le cadre du débruitage sont globalement satisfaisant. L'image bruitée est visiblement améliorée par le traitement et les contours de l'image ne sont pas érodés de manière visible.

À l'inverse, la résolution du problème d'inpainting ne donne pas de résultats de très bonne qualité. Ils sont d'autant plus décevants que les exemples présentés dans l'article original sont très bien reconstruits.

Un problème important en traitement d'images est le choix des paramètres du modèle. Il est possible d'obtenir des résultats très convaincants si les paramètres sont bien choisis.

Une approche statistique semble appropriée : on dégrade artificiellement une base de données d'images puis on cherche les paramètres reconstruisant au mieux ces images. Cela semblerait presque facile, mais nous allons vite être confronté à une nouvelle difficulté. Pour choisir au mieux ces paramètres, il faut être capable de mesurer la qualité de la reconstruction et ce n'est pas évident.

Dans l'article original, quelques pages sont consacrées à ce sujet. Elles montrent bien les problèmes qui peuvent se poser lors d'une choix d'une telle «mesure». Deux cas peuvent se présenter :

- On a obtenu une reconstruction de qualité suffisante de l'image, mais pas la convergence au sens de notre «mesure». Le programme continue donc d'effectuer de calculs inutiles.
- Au sens de notre «mesure», la convergence a été obtenue et le programme s'arrête donc. Cependant la reconstruction n'est visuellement pas satisfaisante et le programme devrait continuer.

Il est également probablement possible d'améliorer nos résultats en choisissant un ordre de remplissage des pixels pertinent (au sens qu'il dépend des caractéristiques de notre image). Actuellement, nous effectuons un balayage standard (de gauche à droite et de haut en bas) de l'image et nous remplissons les pixels au fur et à mesure. Il serait plus intéressant de remplir d'abord les pixels sur lesquels nous avons beaucoup d'information (beaucoup de voisins connus) et ceux situés sur les contours (zones de fort gradient).