Projet CLANU 2013: Filtrage d'image par EDP

C. Reichert, E. Bretin, T. Grenier, P. Delacharte, T. Redarce

28 février 2014

Voici l'énoncé du projet CLANU. Ce projet a pour objectif principal de vous faire concevoir un logiciel en langage Matlab/C++ illustrant une méthode mathématique spécifique : le filtrage d'image par EDP (équations aux dérivées partielles). Les méthodes sont introduites dans cet énoncé et différentes implémentations sont à faire pour répondre aux questions.

La qualité des résultats et de l'analyse mathématique d'une part, ainsi que la qualité de la conception et de la démarche de conception d'autre part, seront prises en considérations par les enseignants d'informatique et de mathématiques.

1 Introduction

Dans le cadre de ce projet on s'intéresse à une image de taille $n \times n$ en niveaux de gris représentée par la fonction I définie sur $[1, n]^2$ et à valeur dans $[0, 1]$.

L'objectif de ce projet est de filtrer une image (précédemment bruitée) en utilisant l'approche des EDPs. Plus précisément, l'idée est de déterminer le minimum de l'énergie suivante

$$J(u) = \frac{1}{2} \int |\nabla u|^2 dx + \frac{1}{2\eta} \int (u - I)^2 dx,$$

où η est un paramètre d'attache aux données de l'image I et $\int |\nabla u|^2 dx$ est un terme de régularisation. On peut associer à la fonctionnelle J une équation d'évolution dont la solution stationnaire permet de minimiser J. L'équation d'évolution pour J ci-dessus s'écrit

$$\partial_t u(x,t) = \Delta u(x,t) + \frac{1}{\eta}(I(x) - u(x,t)), t > 0.$$

On peut prendre comme condition initiale à l'instant $t = 0$, $u(x, 0) = I(x)$.

2 Discrétisation de l'équation de diffusion linaire

Il s'agit tout d'abord de discrétiser l'image $I(x)$, comme une matrice $(I_{i,j})_{1\leq i,j\leq n}$ et définie comme l'approximation de $I(x)$ au pixel $x_{i,j}$ définie par $x_{i,j} = (i, j)$. De même, la solution $u(x, t)$ sera discrétisée par $u^k_{i,j}$ comme une approximation de $u(x_{i,j}, t_k)$ au pixel $x_{i,j}$ et à l'instant $t_k = k\Delta t$, où Δt est un pas de temps à définir et $k \in [0, T/\Delta t]$ avec T un temps maximal.
L'équation peut alors se discrétiser de la manière suivante

$$\begin{cases} \partial_t u(x_{i,j}, t_k) & \simeq \frac{u^{k+1}_{i,j} - u^k_{i,j}}{\Delta t} \\ \Delta u(x_{i,j}, t_k) & \simeq u^k_{i+1,j} + u^k_{i-1,j} + u^k_{i,j+1} + u^k_{i,j-1} - 4u^k_{i,j}, \end{cases}$$

ce qui conduit au système d'équations pour $(i, j) \in [1, n]^2$ et $k \in \mathrm{N}$,

$$u^{k+1}_{i,j} = u^k_{i,j} + \Delta t \left(u^k_{i+1,j} + u^k_{i-1,j} + u^k_{i,j+1} + u^k_{i,j-1} - 4u^k_{i,j}\right) + \frac{\Delta t}{\eta}\left(I_{i,j} - u^k_{i,j}\right).$$

Pour régler les problèmes de bord, nous ne considérons pas l'évolution de la solution de l'équation à l'extérieur du domaine de définition de l'image, ce qui nous amène à utiliser comme condition de bord

$$u_{0,j} = u_{1,j}, u_{i,0} = u_{i,1}, u_{n+1,j} = u_{n,j} \text{ et } u_{i,n+1} = u_{i,n},$$

pour $(i, j) \in [1, n]^2$.

3 Equation avec une diffusion non-linéaire

Une conséquence de la diffusion linéaire est de rendre l'image de plus en plus flou (faire le lien avec le produit de convolution avec un noyau Gaussien), et ainsi de lisser les contours. Pour corriger cet effet, on modifie l'équation de diffusion linéaire précédente en introduisant un opérateur de diffusion non linéaire :

$$\partial_t u(x, t) = \mathrm{div}(\mathbf{G}(\nabla u)\nabla u) + \frac{1}{\eta}(I(x) - u(x, t)),$$

où

$$\mathbf{G}(\nabla u) = \begin{pmatrix} g(|\partial_{x_1} u|) & 0 \\ 0 & g(|\partial_{x_2} u|) \end{pmatrix}$$

et où la fonction $g(s)$ est supposée décroissante. L'intérêt est que plus la norme $\|\nabla u\|$ est grande, moins le terme de diffusion agit. Ceci permet, par un choix judicieux de la fonction g, de préserver les contours. Trois exemples de fonctions inspirées des travaux

de Perona et Malik [**?**] sont

$$g_1(s) = \begin{cases} 1 & \text{si } s^2 \leq \lambda^2 \\ 0 & \text{sinon} \end{cases}$$
$$g_2(s) = \frac{1}{1+(s/\lambda)^2}$$
$$g_3(s) = e^{-(\frac{s}{2\lambda})^2}$$

où λ est un paramètre à choisir en fonction du bruit. Comme précédemment, nous utilisons une approximation $u_{i,j}^k$ de la solution u au pixel $x_{i,j}$ et à l'instant $t_k = k\Delta t$. Une discrétisation de cette équation de diffusion non linéaire conduit alors au système d'équations suivant : pour $(i,j) \in [1,n]^2$ et $k \in \mathrm{N}$,

$$\begin{aligned} u_{i,j}^{k+1} &= u_{i,j}^k + \Delta t \left((g_N)_{i,j} \nabla_N u_{i,j}^k + (g_S)_{i,j} \nabla_S u_{i,j}^k + (g_E)_{i,j} \nabla_E u_{i,j}^k + (g_W)_{i,j} \nabla_W u_{i,j}^k \right) \\ &\quad + \frac{\Delta t}{\eta} \left(I_{i,j} - u_{i,j}^k \right). \end{aligned}$$

où

$$\begin{cases} \nabla_N u_{i,j} &= u_{i-1,j} - u_{i,j} \\ \nabla_S u_{i,j} &= u_{i+1,j} - u_{i,j} \\ \nabla_E u_{i,j} &= u_{i,j+1} - u_{i,j} \\ \nabla_W u_{i,j} &= u_{i,j-1} - u_{i,j} \end{cases} \quad \text{et} \quad \begin{cases} (g_N)_{i,j} &= g(|\nabla_N u_{i,j}|) \\ (g_S)_{i,j} &= g(|\nabla_S u_{i,j}|) \\ (g_E)_{i,j} &= g(|\nabla_E u_{i,j}|) \\ (g_W)_{i,j} &= g(|\nabla_W u_{i,j}|) \end{cases}$$

4 Questions d'analyse numérique

1. Implémenter l'algorithme de la diffusion linéaire sous `Matlab` et le tester sur une image « échiquier » que l'on créera avec la commande `I = checkerboard(32)` et que l'on bruitera avec la commande `I = I + 0.1*randn(size(I))`. Les images devraient ressembler a celles de la figure 1. Tester l'algorithme sur l'image `lena256.bmp` bruitée. Astuce : on peut importer une image dans une matrice en `Matlab` avec la commande `imread`, par exemple `I = double(imread('lena256.bmp'))`. On utilisera cette fois un bruit plus faible `I = I + 0.05*randn(size(I))`. Remarques ?
2. Implémenter et tester l'algorithme de diffusion non-linéaire sur les mêmes images avec différents jeux de paramètres η, λ et différents choix de g. Remarques ?

On effectuera les tests numériques avec quelques jeux de paramètres représentatifs où $0.1 \leq \lambda \leq 2,\ 0.1 \leq \eta \leq 5,\ T = 1000,\ \Delta t = 0.05$.

5 Questions d'informatique

Proposition : sous matlab travailler sur des petites matrices simples (on peut leur donner quelques .mat ou sous forme de petites images : un dirac, un bruit gaussien

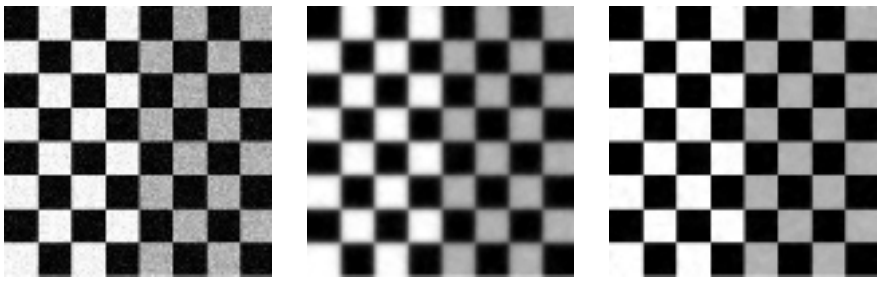

FIGURE 1: Image « échiquier » bruitée filtrer avec diffusion linéaire et non-linéaire.

additif sur une image binaire) et faire un travail de validation.
Ils valident leur code C++ sur les mêmes données et conditions que sous matlab.
Puis ils utilisent le C++ pour des traitements d'images de bonne taille.

1. En vous appuyant sur la classe *QImage* de *Qt*, faire un programme qui ouvre une image, calcule le négatif et enregistre ce résultat. Une fois la lecture du fichier faite, il est recommandé de gérer la matrice de pixel dans un objet du C++ de votre convenance (*double* $**$, *vector* <>, ...) puis, quand les calculs sont terminés, de repasser les valeurs dans un objet *QImage* en vue de la création d'un fichier image. Nous vous recommandons l'utilisation des formats d'image png et bmp qui sont des formats sans pertes et sans interpolation des données (attention au jpeg...).
2. Implémenter en langage $C/C++$ vos algorithmes de diffusion linéaire et non-linéaire dans deux fonctions distinctes. Tester et valider vos algorithmes en les comparant aux résultats obtenus avec matlab.
3. La description mathématique précédente peut être étendue pour permettre le filtrage d'images couleur. Classiquement les images couleurs sont RGB, c'est à dire que l'intensité de chaque pixel est un vecteur de 3 composantes (on notera ce vecteur $\mathbf{u}_{i,j}$). Ainsi il faudra appliquer le filtre à chacune des trois composantes couleur et remplacer les différences d'intensité par des distances euclidiennes. Par exemple : $\nabla_N \mathbf{u}_{i,j} = \sqrt{(\mathbf{u}_{i-1,j} - \mathbf{u}_{i,j})^T.(\mathbf{u}_{i-1,j} - \mathbf{u}_{i,j})}$. Après avoir donné la description mathématiques de votre solution, vous implémenterez la diffusion non linéaire d'image couleur. Vous testerez votre fonction sur les images corrompues données sous *moodle* et vous comparerez vos résultats avec les images non corrompues.
4. Question bonus : Étudier l'évolution des temps de calcul en changeant les options de compilation. Le code produit par la compilation C++ peut être optimisé afin d'obtenir des exécutions plus rapides. Ces optimisations sont faites par le

compilateur qui va analyser et optimiser votre code. L'option qui contrôle l'optimisation est "-O" suivi du degré d'optimisation : 1, 2 3 et fast (par exemple *-O3* ou *-Ofast* pour la suite *gnu*). A noter que ces optimisations sont d'autant plus significatives que votre code C++ est bien écrit (passage par pointeurs ou références notamment) et qu'elles ne réduisent pas la complexité algorithmique de votre programme. Il vous faudra modifier votre fichier '.pro' pour changer cette option.

Consignes informatiques

Pour répondre à ces question, vous veillerez à respecter les consignes suivantes :
- La réalisation informatique se fera exclusivement en $C/C++$ sous $QtCreator$.
- La réalisation d'une IHM graphique n'est pas obligatoire.
- L'utilisation d'une bibliothèque graphique externe autre que Qt n'est pas autorisée.
- L'utilisation d'une bibliothèque de calcul tierce n'est pas autorisée.
- L'utilisation des classes et objets de la *std* et de *Qt* sont autorisés.

6 Évaluations et rendus

6.1 Organisation

Ce projet est évalué dans le cadre des modules IF2 et MA2 par une note de projet et une évaluation individuelle. Le projet se fera en binôme. Les binômes sont ceux établis pour les TP par l'équipe pédagogique.

Chaque binôme déposera sur *moodle* :

1. le rapport au format pdf,
2. la réalisation informatique : un fichier zip des sources.

Le rapport (pdf) et les sources (zip) sont à rendre **au plus tard le lundi 6 juin**. Attention, les rendus se faisant sous moodle, aucun délai supplémentaire ne sera accordé.

6.2 Réalisation informatique

Un fichier **.zip** comportant uniquement les fichiers matlab et C++ nécessaires à la compilation et le(s) fichier(s) de projet **.pro** sera a déposé sur moodle. Dans l'archive, il ne faut pas de mettre vos exécutables : votre projet sera recompilé. Assurez-vous que les fichiers présents dans l'archive permettent la création des exécutables. De même, ne pas mettre dans cette archive les résultats. Le nom du fichier déposé sera : `BXX_NOM1_NOM2_Source.zip` où vous aurez remplacer les XX par votre numéro de binôme et les $NOM1$ et $NOM2$ par vos noms.

6.3 Rapport

Le rapport comportera les parties suivantes :

1. une introduction présentant l'étude et l'organisation du rapport.
2. une partie *Analyse Numérique* dans laquelle les réponses aux questions seront données.
3. une partie *Réalisation Informatique* qui détaillera les solutions utilisées, les algorithmes implémentés, l'organisation du code ainsi que les réponses aux questions.
4. une conclusion résumant le travail réalisé avec les résultats obtenus, ainsi que l'organisation du déroulement du projet et la répartition du travail.
5. au besoin, une partie bibliographie.

Les qualités de rédaction et de présentation ainsi que de votre pertinence d'analyse des résultats seront grandement prises en compte dans l'évaluation du projet. Afin d'éviter certaines dérives du travail en binôme, il vous est demandé de préciser le partage des tâches dans le rapport et l'auteur des sources. Le nom du fichier déposé sera : `BXX_NOM1_NOM2_Rapport.pdf` où vous aurez remplacer les XX par votre numéro de binôme et les $NOM1$ et $NOM2$ par vos noms.

6.4 Evaluation individuelle

Elle aura lieu pendant l'épreuve de IF2 et/ou de MA2. Elle est obligatoire et sera sans document. Elle durera au minimum 30 minutes.

Références