Université Paris Descartes
UFR de Mathématiques et Informatique
45, rue des Saints-Pères 75270 Paris cedex 06

UNIVERSITÉ PARIS DESCARTES

MASTER 1

PROGRAMMATION

Projet "Interpolation d'image par variation totale" - `Scilab`

Rédaction du rapport

Le rapport devra contenir :

- Une introduction expliquant et illustrant le problème.
- Une explication détaillée des algorithmes et codes et leur interprétation.
- Plusieurs exemples d'application.

Introduction

Ce projet porte sur la restauration d'images comportant des pixels manquants.

Dans ce qui suit, on considère des image discrètes à $n \times m$ pixels et prenant leurs valeurs dans $[0, 255]$. On rappelle qu'une image peut se représenter dans Scilab sous la forme d'une matrice $n \times m$.
On suppose qu'on observe une image v dont certains pixels sont cachés (leur valeurs sont à 0). Le but est d'interpoler cette image pour retrouver les valeurs en ces pixels manquants.

On commence par créer l'image à interpoler. On part pour cela d'une image complète u, dont on masque aléatoirement une proportion p de ses pixels. On note v l'image ainsi obtenue. Par exemple, si u est l'image `simpson.bmp`, et qu'on choisit de cacher la moitié des pixels ($p = 0.5$) :

```
p=0.5;
u = read_bmp('simpson.bmp');
u = u/max(u(:));      // normalisation de l'image entre 0 et 1
mask = (rand(u)<p);   // création du masque
v = u.*(1-mask)  ;        // on masque les pixels et on remplace leur valeur par 0
figure(1);fview(u);
figure(2);fview(v);
```

Méthode

On oublie maintenant u et on va essayer à partir de l'image v de retrouver les valeurs des pixels masqués. On propose pour cela de trouver une image f assez régulière dont les valeurs coïncident avec celles de v sur les pixels non cachés.
On note C l'ensemble des images qui prennent les mêmes valeurs que v en dehors du masque. Pour mesure la régularité de f on décide d'utiliser la norme du gradient de f. On cherche donc l'image f qui minimise le problème d'optimisation suivant :

$$\min_{f \in C} E(f),$$

où l'énergie E est soit

$$E_2(f) = \sum_{i=1}^{n} \sum_{j=1}^{m} \|(\nabla f)_{i,j}\|_2^2,$$

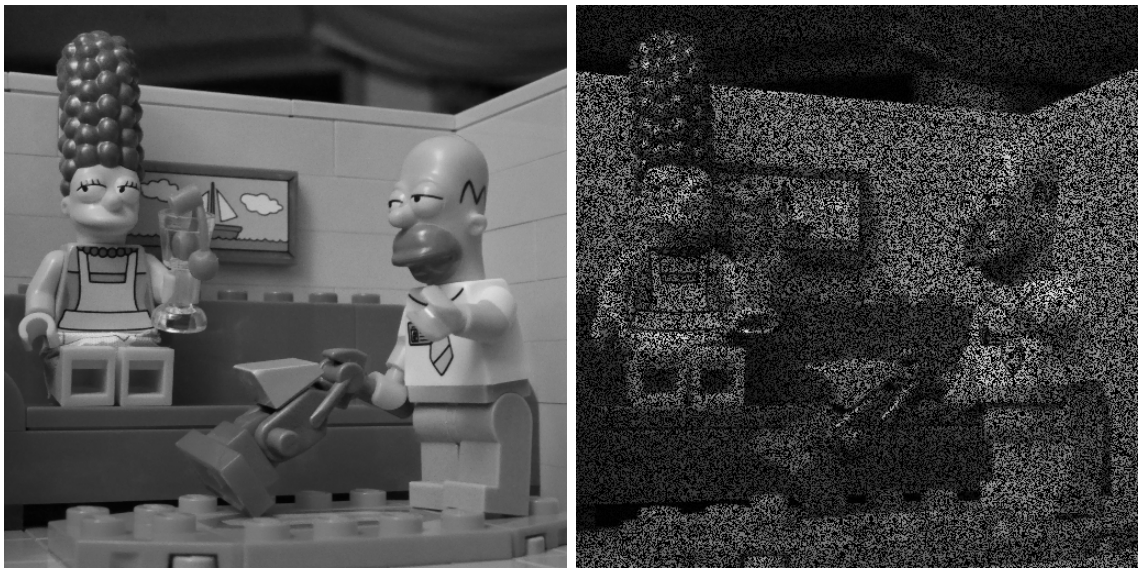

FIGURE 1 – A gauche l'image originale u. A droite l'image v qui a été construite en masquant aléatoirement 50% des pixels de u. On veut, à partir de v, essayer de retrouver l'image u le mieux possible.

soit

$$E_1(f) = \sum_{i=1}^{n}\sum_{j=1}^{m} \|(\nabla f)_{i,j}\|_2.$$

L'algorithme qu'on se propose d'utiliser pour minimiser cette énergie est un algorithme de **gradient projeté**. Il consiste à faire une descente de gradient sur $E(f)$, et à projeter à chaque pas de la descente le résultat obtenu sur la contrainte C. Pour pouvoir faire la descente de gradient, on a besoin de calculer la dérivée de la fonction $E(f)$.

Dans le cas de E_2, on obtient :

$$\frac{\partial E_2}{\partial f} = -2\Delta f,$$

où $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$ est le laplacien de la fonction f.

Dans le cas de E_1 on obtient

$$\frac{\partial E_1}{\partial f} = -\mathrm{div}\left(\frac{\nabla f}{\|\nabla f\|_2}\right).$$

On rappelle que la divergence d'une fonction $g : \mathbb{R}^2 \to \mathbb{R}^2$ est $\mathrm{div}(g) = \frac{\partial g_x}{\partial x} + \frac{\partial g_y}{\partial y}$. Pour éviter de diviser par 0 aux points ou le gradient est nul, on remplace en pratique la quantité $\|\nabla f\|_2$ qui est au dénominateur par $\sqrt{\|\nabla f\|_2^2 + \varepsilon^2}$ avec une valeur de ε très petite.

Le pseudo-code de l'algorithme d'interpolation par variation totale peut s'écrire de la manière suivante :

Pseudo-code du gradient projeté

Entrée : image v, pas α, ε, nombre d'itérations M.
Sortie : image restaurée f
Initialisez $f = v$.
for $k = 1$ **to** M **do**
  Descente de gradient : $f = f - \alpha\frac{\partial E(f)}{\partial f}$
  Projection sur C : $f = \pi_C(f)$

Fin du pseudo-code

1. Ecrire une fonction $[fx, fy] = grad(f)$ qui prenne en entrée une image f et renvoie deux images de même taille que f représentant le gradient ∇f de f : fx est la dérivée en x et fy la dérivée en y. On utilisera le schéma suivant :

$$fx(i,j) = \begin{cases} f(i+1,j) - f(i,j) & \text{si } i < n \\ 0 & \text{sinon} \end{cases}, \quad fy(i,j) = \begin{cases} f(i,j+1) - f(i,j) & \text{si } j < m \\ 0 & \text{sinon.} \end{cases}$$

2. Ecrire une fonction $d = div(gx, gy)$ qui étant donné un champ de vecteurs $g = [gx, gy]$ (g est une matrice $n \times m \times 2$) renvoie la divergence de ce champ de vecteurs. On pourra utiliser le schéma discret suivant :

$$div(g)_{(i,j)} = \begin{cases} gx(i,j) - gx(i-1,j) & \text{si } 1 < i < n \\ gx(i,j) & \text{si } i = 1 \\ -gx(i-1,j) & \text{si } i = n \end{cases} + \begin{cases} gy(i,j) - gy(i,j-1) & \text{si } 1 < j < n \\ gy(i,j) & \text{si } j = 1 \\ -gy(i,j-1) & \text{si } j = n \end{cases}$$

3. Ecrire une fonction $l = laplacien(f)$ qui calcule le laplacien de l'image discrète f. On rappelle que $\Delta f = \text{div}(\nabla f)$ donc vous pouvez l'écrire très simplement à partir des fonctions **grad** et **div** précédentes.
4. Ecrire une fonction $f = projection(f, v, mask)$ qui étant donnée une image f la projette sur la contrainte C. Ceci revient simplement à changer les valeurs de l'image f en dehors du masque pour qu'elles soient égales à celles de v.
5. Utiliser les fonctions précédentes pour coder l'algorithme de gradient projeté. Coder d'abord une version de l'algorithme pour l'énergie E_2, puis pour l'énergie E_1. Pour les valeurs des paramètres, vous pouvez commencer par prendre $\varepsilon = 10^{-2}$, un pas $\alpha = 10^{-2}/5$ et plusieurs centaines d'itérations.
6. Tester les deux méthodes d'interpolation pour plusieurs valeurs de p et des paramètres de la méthode. Tester sur plusieurs images, par exemple **simpson.bmp** et **barbara.bmp**. Comparer les images interpolées obtenues avec les images originales et commentez les résultats obtenus.