Année universitaire
2020 - 2021

IMI – Optimisation et problèmes inverses – S8

TP3 – Optimisation pour le traitement d'image 2D et 3D

Marion Foare

Objectifs.

- Application des notions d'**optimisation** lisse et non lisse à des problématiques de traitement d'image 2D et 3D
- Compréhension des modèles et de leurs limites
- Optimisation multi-variées

Déroulement. Ce TP se déroule sur **1 séance** et est à effectuer par binôme sous Matlab/Octave, sur **un seul des sujets au choix**. Vous trouverez sur CPe-campus une archive contenant l'ensemble des fichiers nécessaires à la réalisation de ce TP.

Chaque sujet est proposé au format bureau d'étude, pour compléter votre cours sur les méthodes d'optimisation. Il doit vous permettre de mettre en application les méthodes et modèles d'optimisation, abordés aux TP1 et TP2, pour différents traitements d'image en 2D et 3D, et d'étudier les limites de ces approches. Il permet aussi d'aller plus loin dans la compréhension du design d'un problème d'optimisation en fonction de l'application visée. Il est dès lors indispensable que vous preniez le temps d'étudier l'influence des paramètres et, le cas échéant, de vos choix d'implémentation.

Pour chacun des sujets proposés, vous trouverez :

- le contexte,
- le problème d'optimisation considéré,
- une aide pour la résolution du problème.

Configuration. Ce TP nécessite les librairies suivantes :

- Matlab : *Image Processing Toolbox*

- Octave : *Image toolbox* puis, en début de script :

```
pkg install -forge image
pkg load image
```

Compte-rendu. A la fin de la séance, vous devrez avoir rédigé un compte-rendu (max. 10 pages) permettant d'identifier clairement votre démarche pour la résolution du problème choisi :

- contexte, position du problème
- analyse et compréhension du problème
- solution proposée et mise en œuvre
- résultats

L'évaluation de TP portera sur la démarche proposée, et en particulier sur les votre argumentaire concernant la solution proposée. Une attention particulière sera portée à la clarté du compte-rendu final.

Ce TP a pour vocation d'être formateur, et peut nécessiter de travailler en groupe (fortement encouragé, qui plus est !). Toutefois, il vous est demandé **un compte-rendu par quadrinôme**, correspondant à **votre** restitution des notions. Il en va de même pour le code. Tout travail emprunté à un (ou plusieurs) autre(s) groupe(s) doit être **explicitement identifié**.

Sujets proposés

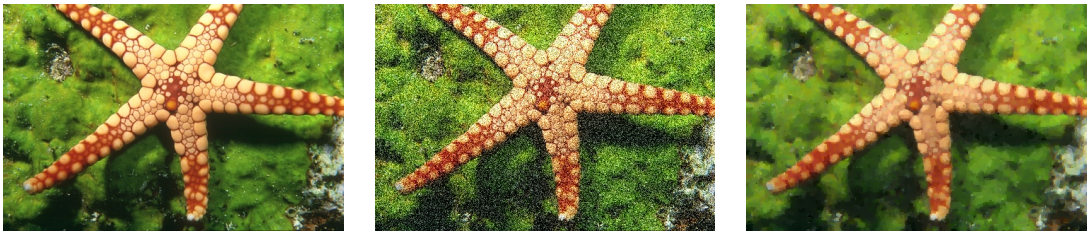
Nous rappelons que toute tentative de copie entrainera une sanction de l'ensemble des binômes concernés.

1 Débruitage d'image couleur ⋆

Nous avons considéré dans les TP1 et TP2 des fonctions de coût de la forme

$$f(x) = \|Hx - z\|_2^2 + \lambda R(x), \tag{1}$$

où R est une régularisation choisie en fonction du type de résultat souhaité.
Cette fonction de coût n'est cependant pas appropriée dans le cas du débruitage d'images couleur. Aussi, on se propose d'étendre cette fonction de coût aux images couleur (ou multicanaux de manière générale).



Vérité terrain $\bar{x}$ | Observation z | Estimation $\hat{x}$

FIGURE 1 – Débruitage d'image couleur.

Objectifs.

- extension multicanaux
- résolution multivariée

Problème. Soit $\bar{x} = (\bar{x}_1, \bar{x}_2, ..., \bar{x}_C) \in \mathbb{R}^{N \times C}$ une image de taille $N = h \times w$ possédant C canaux. En particulier,

- si $\bar{x}$ est une image en niveau de gris, $C = 1$;
- si $\bar{x}$ est une image RGB, $C = 3$ et $\bar{x} = (\bar{x}_1, \bar{x}_2, \bar{x}_3) = (r, g, b)$;

Soit $n \in \mathbb{R}^M \sim \mathcal{N}(0, \sigma)$ un bruit blanc Gaussien additif. On considère le modèle de dégradation :

$$\forall c = 1, ..., C, \quad z_c = \bar{x}_c + n \tag{2}$$

(Le cas avec flou pourra être traité dans un second temps, en fonction du choix de la régularisation R).

Étant donnée l'observation $z \in \mathbb{R}^{M \times C}$, on cherche une approximation $\hat{x} \in \mathbb{R}^{N \times C}$ de $\bar{x}$, solution du problème d'optimisation :

$$\hat{x} = (\hat{x}_1, ..., \hat{x}_C) \in \underset{x \in \mathbb{R}^{N \times C}}{\operatorname{argmin}} \underbrace{\sum_{c=1}^{C} \|x_c - z_c\|_2^2 + \lambda R(x_c)}_{f(x) = f(x_1, ..., x_C)} \tag{3}$$

Difficulté. Pour minimiser numériquement une telle fonction de coût, on adopte très souvent une stratégie de minimisation alternée. Autrement dit, à chaque itération, on fixe les variables x_j pour $j \neq i$ et on minimise par rapport à x_i. Dans le cas RGB, cela conduit aux trois sous-problèmes suivants :

$$r_{k+1} = \text{argmin}_{r \in \mathbb{R}^N} f(r, g_k, b_k) \quad \text{(R)}$$
$$g_{k+1} = \text{argmin}_{g \in \mathbb{R}^N} f(r_{k+1}, g, b_k) \quad \text{(G)}$$
$$b_{k+1} = \text{argmin}_{b \in \mathbb{R}^N} f(r_{k+1}, g_{k+1}, b) \quad \text{(B)}$$

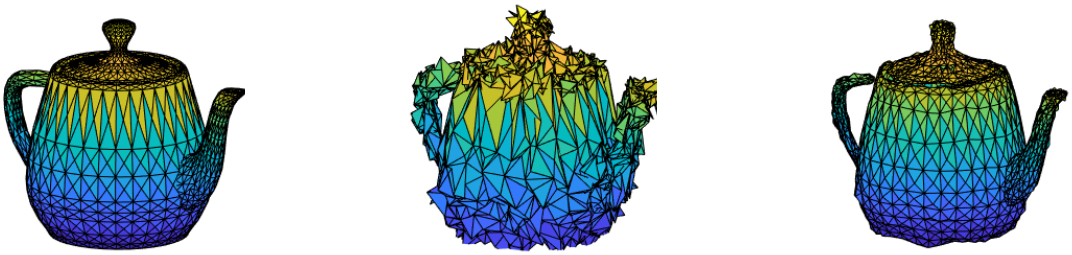
L'algorithme qui en résulte est le suivant :

```
Initialiser r0, g0, b0;
Choix des paramètres pour la résolution de (R), (G), (B);
For k=1:niter
   Résoudre (R);
   Résoudre (G);
   Résoudre (B);
end
```

2 Débruitage de maillage ⋆

Nous avons considéré dans les TP1 et TP2 des applications en 1D et 2D. Toutefois, les modèles considérés sont valables en N dimensions. Il est possible d'utiliser les modèles étudiés pour régulariser des objets de l'espace : surfaces de l'espace, nuages de points, volumes IRM, etc.
Ainsi, le domaine de la synthèse d'images nécessite régulièrement d'utiliser des techniques de traitement d'images originellement pensées en 2D ; en particulier, les techniques de débruitage et de lissage.

Aussi, on se propose de s'intéresser au débruitage/lissage d'un maillage triangulé, ce dernier étant une surface de l'espace. La particularité dans ce cas vient du fait qu'un point ne possède pas un nombre constant de voisins.



Vérité terrain $\bar{x}$ | Observation z | Estimation $\hat{x}$

FIGURE 2 – Débruitage d'un maillage.

Objectifs.

- extension 3D
- extension graphe

Problème. Soit $\bar{x} \in \mathbb{R}^{N\times 3}$ un maillage triangulé à N points, chaque point étant un point dans l'espace.

Soit $n \in \mathbb{R}^{N\times 3} \sim \mathcal{N}(0,\sigma)$ un bruit blanc Gaussien additif. On considère le modèle de dégradation :

$$z = \bar{x} + n \tag{4}$$

Étant donnée l'observation $z \in \mathbb{R}^{N\times 3}$, on cherche une approximation $\hat{x} \in \mathbb{R}^{N\times 3}$ de $\bar{x}$, solution du problème d'optimisation :

$$\hat{x} \in \underset{x\in\mathbb{R}^{N\times 3}}{\operatorname{argmin}} \underbrace{\|x-z\|_2^2 + \lambda R(x)}_{f(x)} \tag{5}$$

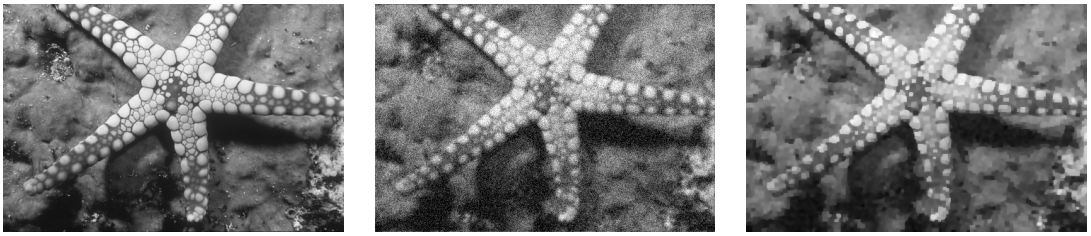
Difficulté. Les maillages triangulés fournis pour ce sujet sont encodés au format `.off`. Des fonctions de lecture (`loadOff.m`) et d'écriture (`exportOff.m`) spécifiques pour ce format sont fournies.
L'affichage se fait au moyen de la fonction `trisurf(TRI,X,Y,Z)`.
De plus, il faut porter une attention particulière au choix des paramètres en 3D, le maillage étant très sensible à des changements d'amplitude trop importants.

3 Restauration d'image par Variation Totale ⋆⋆

Nous avons considéré au TP2 un modèle classique en traitement d'image, connu sous le nom de modèle TV (*Total Variation*). Il est composé d'une attache à la donnée et d'une régularisation non lisse :

$$\hat{x} \in \underset{x \in \mathbb{R}^N}{\operatorname{argmin}} \ \|x - z\|_2^2 + \lambda \|Dx\|_1 \tag{6}$$

Le modèle (6) permet de débruiter des images en niveaux de gris, en faisant l'*a priori* que l'image à approcher est constante par morceaux. Mais sans opérateur H dans l'attache à la donnée, ce modèle ne permet pas de prendre en compte une dégradation de type flou.
Aussi, on se propose d'étendre le modèle TV à la déconvolution.



Vérité terrain $\bar{x}$ Observation z Estimation $\hat{x}$

FIGURE 3 – Déconvolution et débruitage par Variation Totale.

Objectifs.

- ajout du flou H au modèle TV
- résolution multivariée

Problème. Soit $\bar{x} \in \mathbb{R}^N$ une image de taille $N = h \times w$. Et soient $H \in \mathbb{R}^{M \times N}$ un opérateur de dégradation linéaire (e.g. un flou), et $n \in \mathbb{R}^M \sim \mathcal{N}(0, \sigma)$ un bruit blanc Gaussien additif. On considère le modèle de dégradation :

$$z = H\bar{x} + n \tag{7}$$

Étant donnée l'observation $z \in \mathbb{R}^M$, on cherche une approximation $\hat{x} \in \mathbb{R}^N$ de $\bar{x}$, solution du problème d'optimisation :

$$\hat{x} \in \underset{x \in \mathbb{R}^N}{\operatorname{argmin}} \ \|Hx - z\|_2^2 + \lambda \|Dx\|_1 \tag{8}$$

Difficulté. La résolution du problème (8) comme un simple modèle TV (sans flou) nécessiterait le calcul de la conjuguée convexe de $\|Hx - z\|_2^2$. Cela n'est pas possible à cause de l'opérateur H.
On pose alors une variable auxiliaire $y = Dx$, et on réécrit le problème sous la forme d'un problème contraint

$$\begin{cases} (\hat{x}, \hat{y}) \in \underset{x,y}{\operatorname{argmin}} \ \|Hx - z\|_2^2 + \lambda \|y\|_1 \\ \text{subject to } y = Dx \end{cases} \tag{9}$$

En pratique, la résolution du problème (8) est donc réalisée en cherchant une solution au problème

$$(\hat{x},\hat{y}) \in \underset{x,y}{\operatorname{argmin}} \underbrace{\|Hx-z\|_2^2 + \lambda\|y\|_1 + \frac{\mu}{2}\|y-Dx\|_2^2}_{f(x,y)}, \tag{10}$$

Cette méthode est appelée *Alternated Direction Method of Multipliers* (ADMM).

Pour minimiser numériquement une telle fonction de coût, on adopte très souvent une stratégie de minimisation alternée. Autrement dit, à chaque itération, on fixe la valeur de y et on minimise par rapport à x, puis inversement. Cela conduit aux deux sous-problèmes suivants :

$$x_{k+1} = \operatorname{argmin}_{x\in\mathbb{R}^N} f(x, y_k) \tag{Px}$$
$$y_{k+1} = \operatorname{argmin}_{y\in\mathbb{R}^N} f(x_{k+1}, y) \tag{Py}$$

L'algorithme qui en résulte est le suivant :

```
1: Initialiser x0, y0;
2: Choix des paramètres pour la résolution de (Px);
3: Choix des paramètres pour la résolution de (Py);
4: For k=1:niter
5:   Résoudre (Px);
6:   Résoudre (Py);
7: end
```

4 Décomposition cartoon + texture ⋆⋆

Étant donnée une image, on souhaite pouvoir la décomposer en une image de texture et une image "cartoon", qui donne uniquement l'information de couleur. Dit autrement, on souhaiterait séparer les informations de texture (hautes fréquences), de l'intensité des régions constantes (plutôt basse fréquence) de l'image.

Donnée $z = \bar{x}$ Texture estimée $\hat{u}$ Cartoon estimé $\hat{v}$ Estimation $\hat{x} = \hat{u} + \hat{v}$

FIGURE 4 – Décomposition cartoon+texture.

Objectifs.

- résolution multivariée

Problème. Soit $\bar{x} \in \mathbb{R}^N$ une image de taille $N = h \times w$. Pour cette application, l'observation n'est pas dégradée par rapport à la vérité terrain, aussi

$$z = \bar{x} \tag{11}$$

Étant donnée l'observation $z \in \mathbb{R}^N$, on cherche une approximation $\hat{x} \in \mathbb{R}^N$ de $\bar{x}$, telle que $\hat{x} = \hat{u} + \hat{v}$. Les composantes u et v représentent respectivement la texture et la partie cartoon de l'image. On se propose de les estimer via le problème d'optimisation :

$$(\hat{u}, \hat{v}) \in \underset{(u,v)\in\mathbb{R}^N}{\operatorname{argmin}} \underbrace{\|u + v - z\|_2^2 + \lambda \|Bu\|_2^2 + \mu \|Dv\|_1}_{f(u,v)} \tag{12}$$

Les opérateurs `B` et `Badj` spécifiques à ce sujet sont disponibles dans le dossier "opérateurs".

Difficulté. Pour minimiser numériquement une telle fonction de coût, on adopte très souvent une stratégie de minimisation alternée. Autrement dit, à chaque itération, on fixe une valeur de y et on minimise par rapport à x, puis inversement. Cela conduit aux deux sous-problèmes suivants :

$$u_{k+1} = \operatorname{argmin}_{u\in\mathbb{R}^N} f(u, v_k) \tag{T}$$
$$v_{k+1} = \operatorname{argmin}_{v\in\mathbb{R}^N} f(u_{k+1}, v) \tag{C}$$

L'algorithme qui en résulte est le suivant :

```
Initialiser u0, v0;
Choix des paramètres pour la résolution de (T);
Choix des paramètres pour la résolution de (C);
For k=1:niter
  Résoudre (T);
  Résoudre (C);
end
```

5 Index des fonctions

Normes.

Si x est un **vecteur** :

`norm(x)`	calcule la norme 2 d'un vecteur $\|x\|_2 = \sqrt{\sum_i x_i^2}$
`norm(x,2)`	calcule la norme 2 d'un vecteur $\|x\|_2 = \sqrt{\sum_i x_i^2}$

Si A est une **matrice** :

`norm(A)`	calcule la norme d'opérateur $\|\|\|A\|\|\| = \max(\mathrm{SVD}(A))$
`norm(A,2)`	calcule la norme d'opérateur $\|\|\|A\|\|\| = \max(\mathrm{SVD}(A))$
`norm(A,'fro')`	calcule la norme de Frobenius $\|A\|_2 = \sqrt{\sum_{i,j} A_{ij}^2}$

Remarque : dans le cas d'une matrice, `norm(A,'fro')` = `norm(A(:),2)`.

Matrices.

`H = matH(size(x),type,N)`	Crée la matrice H de flou de type `type` et de taille N
`D = matGamma(s,'gradient')`	Crée la matrice D de gradient (`s = size(x)`)
`L = matGamma(s,'laplacian')`	Crée la matrice L de Laplacien (`s = size(x)`)
`D = matGamma3D(C,F,'gradient')`	Crée la matrice D du gradient d'un maillage (C,F)

Opérateurs (à privilégier).

`y = H(x,type)`	Calcule l'opération matricielle Hx (x **non** vectorisé)
`y = Hadj(x,type)`	Calcule l'opération matricielle $H^\top x$ (x **non** vectorisé)
`y = D(x)`	Calcule l'opération matricielle Dx (x **non** vectorisé)
`y = Dadj(x)`	Calcule l'opération matricielle $D^\top x$ (x **non** vectorisé)
`y = L(x)`	Calcule l'opération matricielle Lx (x **non** vectorisé)
`y = Ladj(x)`	Calcule l'opération matricielle $L^\top x$ (x **non** vectorisé)
`y = B(x)`	Calcule l'opération matricielle Bx (x **non** vectorisé)
`y = Badj(x)`	Calcule l'opération matricielle $B^\top x$ (x **non** vectorisé)

Import/Export de fichier `.off`.

`[C,F] = loadOff(filename)`	Importe le maillage (C,F) du fichier `filename`
`exportOff(filename,C,F)`	Exporte le maillage (C,F) dans le fichier `filename`