Tutoriel UNLOCBOX : Débruitage d'Image par Variation Totale (TV Denoising)

Octobre 2025

1 But du Tutoriel

Ce tutoriel vise à utiliser la boîte à outils **Unlocbox** pour trouver une image débruitée $\mathbf{x}$ en minimisant une fonction objectif spécifique. L'accent est mis sur la compréhension du modèle **ROF** et de l'algorithme **Forward-Backward** (Descente de Gradient Proximal).

2 Analyse Détaillée du Problème d'Optimisation

2.1 Forme Générale d'Unlocbox et Proximal Splitting

Unlocbox résout les problèmes d'optimisation convexe pouvant être exprimés comme la minimisation d'une somme de K fonctions de coût :

$$\mathbf{x}_{\text{sol}} = \arg\min_{\mathbf{x}} \sum_{i=1}^{K} f_i(\mathbf{x})$$

2.2 Équation Spécifique : Le Modèle ROF

Le problème de Débruitage par Variation Totale (Modèle **Rudin-Osher-Fatemi**, ou **ROF**) est formulé comme suit ($K = 2$) :

$$\mathbf{x}_{\text{sol}} = \arg\min_{\mathbf{x}} \quad \underbrace{\lambda \|\nabla \mathbf{x}\|_1}_{f_1(\mathbf{x})} + \underbrace{\frac{1}{2}\|\mathbf{x} - \mathbf{y}\|_2^2}_{f_2(\mathbf{x})}$$

- $\mathbf{x}$: La **solution** recherchée (l'image débruitée).
- $\mathbf{y}$: L'**image d'entrée** bruitée.
- $f_1(\mathbf{x})$: Le **Terme de Régularisation TV**. Il impose que la solution soit lisse (débruite) tout en **préservant les bords nets**.
- $f_2(\mathbf{x})$: Le **Terme de Fidélité aux Données**. Il assure que la solution $\mathbf{x}$ reste **proche** de l'entrée bruitée $\mathbf{y}$.
- λ : Le **Paramètre d'Équilibre** qui pondère le débruitage (f_1) par rapport à la fidélité (f_2).

2.3 Nature des Fonctions : Lisse vs. Non-Lisse

La distinction est cruciale pour le choix du solveur :

- **Fonction Lisse** (f_2) : Sa pente (dérivée) est continue. Elle est gérée par le **Gradient** (`.grad`), qui donne la direction de descente la plus rapide.
- **Fonction Non-Lisse** (f_1) : Elle présente des "coins" (points où la dérivée n'existe pas, ex: la norme ℓ_1). Elle est gérée par l'**Opérateur Proximal** (`.prox`), qui projette la solution sur l'espace des contraintes de manière efficace.

2.4 Le Solveur : Forward-Backward

Pourquoi Forward-Backward est-il choisi ?

Le Forward-Backward est l'algorithme de choix car il est basé sur la méthode du **Proximal Splitting** (séparation proximale). Cette méthode est idéale pour les problèmes d'optimisation où la fonction objectif est la somme d'une fonction facile à dériver (f_2, lisse) et d'une fonction difficile à dériver mais facile à projeter (f_1, non-lisse).

Les propriétés qui le rendent optimal pour le Débruitage TV sont :

- **Efficacité** : Il évite les calculs coûteux du gradient pour la partie non-lisse (f_1).
- **Stabilité** : Il garantit la convergence vers la solution optimale $\mathbf{x}_{\text{sol}}$, à condition que le pas τ soit correctement choisi.

L'algorithme permet de résoudre un problème difficile en alternant deux opérations simples, garantissant ainsi une convergence rapide et stable vers la solution optimale.

L'Itération Principale : Décomposition des Tâches

L'algorithme Forward-Backward calcule la nouvelle solution $\mathbf{x}^{(k+1)}$ à partir de la solution précédente $\mathbf{x}^{(k)}$ en deux étapes, définies par la formule d'itération :

$$\mathbf{x}^{(k+1)} = \text{prox}_{\tau f_1}(\mathbf{x}^{(k)} - \tau \nabla f_2(\mathbf{x}^{(k)}))$$

- **Solution Actuelle** : $[\mathbf{x}^{(k)}]$: C'est l'image (l'approximation de la solution) obtenue à l'itération précédente k.
- **Le Gradient de f_2 (Fidélité)** : $[\nabla f_2(\mathbf{x}^{(k)})]$: Il représente la direction et la force avec lesquelles la solution s'éloigne de l'image bruitée $\mathbf{y}$.
- **Pas de Convergence** : [τ (`param.tau`)] : C'est la taille du pas que nous faisons dans la direction du gradient. Il contrôle la vitesse de convergence et doit être choisi avec soin ($\tau < 2/\beta$).
- **Phase 1 : Forward (Avance)**: $[\mathbf{z} = \mathbf{x}^{(k)} - \tau \nabla f_2(\mathbf{x}^{(k)})]$: Cette expression exécute la descente du gradient sur la partie lisse f_2. Ceci est une pré-estimation de la solution qui minimise l'erreur aux données (notée $\mathbf{z}$).
- **Phase 2 : Backward (Opérateur Proximal)** : $[\mathbf{x}^{(k+1)} = \text{prox}_{\tau f_1}(\mathbf{z})]$: L'opérateur $\text{prox}_{\tau f_1}(\cdot)$ corrige le résultat de la Phase 1 en appliquant la contrainte de régularisation TV (f_1). Cette étape "projette" la solution en lissant le bruit et en préservant les bords.

3 Étapes Détaillées du Tutoriel avec Code MATLAB

3.1 Étape 1 : Initialisation et Préparation des Données

Ce bloc de code configure l'environnement, charge une image de base, puis simule l'ajout de bruit Gaussien pour obtenir l'image bruitée $\mathbf{y}$.

```
% ---  tape  1 : Initialisation et Preparation des Donnees ---
verbose = 2;    % Niveau d'affichage detaille

% 1. Charger et normaliser une image
Im = rescale(double(imread('cameraman.tif'))); % Image originale
N = size(Im);

% 2. Generer une image bruitee (y)
sigma = 0.1;
rng(1);
y = Im + sigma * randn(N); % Ajout du bruit
y = rescale(y);

% 3. Definir le point de d part de la solution
x0 = y;
```

3.2 Étape 2 : Définition des Fonctions f_1 et f_2

Ce bloc définit les deux fonctions (f_1 et f_2) en fournissant à chacune l'opérateur spécifique requis par le solveur Forward-Backward (`.prox` pour f_1, `.grad` pour f_2).

```
% ---  tape  2 : Definition des Fonctions f1 et f2 ---

% --- 2.1. Definition de f1 : Terme TV (Non-Lisse) ---
lambda = 0.05;
param_tv.verbose = verbose - 1;

% L'operateur proximal (prox_tv) est utilise pour gerer la non-lissite.
f1.prox = @(x, T) prox_tv(x, lambda * T, param_tv);
f1.eval = @(x) lambda * norm_tv(x);

% --- 2.2. Definition de f2 : Terme d'Erreur Quadratique (Lisse) ---
% Le gradient est (x - y).
f2.grad = @(x) (x - y);          % Le Gradient requis
f2.eval = @(x) 0.5 * norm(x - y, 'fro')^2;
f2.beta = 1;                     % Constante de Lipschitz L=1, necessaire pour
    tau
```

3.3 Étape 3 : Configuration des Paramètres et Exécution du Solveur

Ce bloc paramètre les critères d'arrêt (`maxit`, `tol`) et définit le pas de convergence (`tau`), qui est crucial pour la stabilité de l'algorithme Forward-Backward. Enfin, il exécute le solveur universel `solvep`.

```
% ---  tape  3 : Configuration et Execution ---
param.verbose = verbose;
param.maxit = 100;
param.tol = 1e-4;

% Le pas tau doit etre < 2/beta.
param.tau = 1.9 / f2.beta;
param.method = 'forward_backward';

% Execution du solveur universel solvep
sol = solvep(x0, {f1, f2}, param);
```

3.4 Étape 4 : Affichage et Analyse des Résultats

Ce bloc final calcule et affiche la Mesure de l'Erreur Quadratique Moyenne (MSE) pour quantifier la réduction du bruit.

```
% ---  tape  4 : Affichage et Analyse des Resultats ---

% Calcul de l'erreur MSE
MSE_bruit = norm(y - Im, 'fro')^2 / numel(Im);
MSE_denoised = norm(sol - Im, 'fro')^2 / numel(Im);

fprintf("\n--- Analyse des performances ---\n");
fprintf("MSE Image Bruitee : %.4f\n", MSE_bruit);
fprintf("MSE Image Debruitee : %.4f\n", MSE_denoised);
```

4 Résultats de la Simulation

4.1 Sortie de l'Algorithme (Console MATLAB)

```
Algorithm selected: FORWARD_BACKWARD
Iter 001:       Prox_TV: obj = 1.924334e+02, rel_obj = 8.999143e-04, TOL_EPS, iter = 18
  f(x*) = 1.924334e+02, rel_eval = 1.153062e+00
Iter 002:       Prox_TV: obj = 1.924334e+02, rel_obj = 8.999143e-04, TOL_EPS, iter = 18
  f(x*) = 1.924334e+02, rel_eval = 0.000000e+00

 FORWARD_BACKWARD:
  f(x*) = 1.924334e+02, rel_eval = 0.000000e+00
 2 iterations
 Stopping criterion: REL_NORM_OBJ

--- Analyse des performances ---
MSE Image Bruitee : 0.0158
MSE Image Debruitee : 0.0137
```

4.2 Interprétation des Résultats

Les sorties de l'algorithme traduisent la convergence du processus d'optimisation vers la solution optimale $\mathbf{x}_{\text{sol}}$.

- **Algorithm selected: FORWARD_BACKWARD** : Confirme que l'approche théorique (basée sur la séparation d'une fonction lisse et non-lisse) est correctement mise en œuvre.
- **Iter 001, 002 : Convergence Rapide** : L'algorithme a trouvé la solution avec une tolérance suffisante en seulement 2 itérations. Bien que le `Prox_TV` lui-même ait nécessité 18 sous-itérations pour chaque étape Backward, la boucle externe Forward-Backward est très rapide.
- **f(x*) = 1.924334e+02** : Représente la valeur finale minimale de la fonction objectif totale ($\lambda\|\nabla\mathbf{x}\|_1 + \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2$). C'est le compromis optimal entre le lissage (TV) et la fidélité aux données (Erreur Quadratique) atteint par le solveur.
- **Stopping criterion: REL_NORM_OBJ** : L'algorithme s'est arrêté parce que la variation relative de la fonction objectif est devenue inférieure à la tolérance fixée (`param.tol`), assurant la convergence.

- **MSE Image Bruitee : 0.0158** : Mesure l'erreur quadratique initiale. Elle est la référence du niveau de bruit.
- **MSE Image Debruitee : 0.0137** : Mesure l'erreur de la solution finale $\mathbf{x}_{\text{sol}}$ par rapport à l'image originale sans bruit (Im).

4.3 Conclusion sur l'Efficacité

Le processus d'optimisation a permis de réduire l'erreur quadratique moyenne (MSE) de l'image de **0.0158** à **0.0137**. Cette réduction de l'erreur témoigne du succès du débruitage TV par l'algorithme Forward-Backward.

4.4 Image Résultante

Figure 1: Image Bruitée vs image débruitée