Tutoriel : Inpainting d'Image par Variation Totale (TV) | Algorithme Douglas-Rachford

Optimisation Proximal, UnLocBox, et Traitement d'Image

1 Introduction : Le Contexte de l'Inpainting en Imagerie Numérique

L'**inpainting d'images** (ou complétion d'images) est un problème classique en traitement du signal et d'image. Il consiste à restaurer des régions endommagées ou manquantes dans une image à partir de l'information disponible et d'hypothèses de régularité. Ce problème est mathématiquement **mal posé** car une infinité de solutions peuvent satisfaire les données observées. L'objectif est donc d'incorporer une contrainte (ou pénalisation) qui sélectionne la solution la plus 'plausible', c'est-à-dire celle qui respecte certaines propriétés structurelles de l'image (comme la douceur, la parcimonie, ou la présence de contours nets).

1.1 Le Principe de la Variation Totale (TV)

Nous utilisons ici le modèle de la **Variation Totale (TV)**, introduit par Rudin, Osher et Fatemi (ROF). La TV est définie comme la norme L_1 du gradient de l'image $\mathbf{x}$.

$$\|\mathbf{x}\|_{TV} = \sum_{i,j} \|\nabla \mathbf{x}_{i,j}\|_2$$

Minimiser la TV favorise des images qui sont **par morceaux constantes**. Cela permet d'éliminer efficacement le bruit sans trop lisser les contours importants de l'image, un avantage majeur par rapport à la régularisation L_2 (Tikhonov) qui produit des flous.

1.2 Formulation du Problème d'Optimisation

Pour l'inpainting, nous avons un jeu de données $\mathbf{y}$ (les pixels observés) et un opérateur de masquage $\mathbf{A}$ (matrice identité sur les pixels visibles, zéro ailleurs). Le problème se formule comme une minimisation sous contrainte :

$$\min_{\mathbf{x}} \quad \underbrace{||\mathbf{x}||_{TV}}_{f_1(\mathbf{x})\text{:Régularisation}} \quad \text{s.c.} \quad \underbrace{||\mathbf{A}\mathbf{x} - \mathbf{y}||_2 \leq \epsilon}_{C\text{:Contrainte de fidélité}}$$

Où ϵ est la tolérance au bruit résiduel. Ce format est parfait pour être décomposé par des méthodes d'optimisation proximale.

2 Optimisation Proximale et Algorithme Douglas-Rachford

2.1 Fonctions Proximale et Opérateur Proximal

L'optimisation proximale est une branche de l'optimisation convexe particulièrement adaptée lorsque la fonction objectif est la somme de fonctions qui peuvent être non-différentiables.

Définition: Opérateur Proximal. *Pour une fonction convexe f, l'opérateur proximal $\mathbf{prox}_f$ est défini par :*

$$\mathbf{prox}_f(\mathbf{x}) = \arg\min_{\mathbf{z}} \left(f(\mathbf{z}) + \frac{1}{2}||\mathbf{z} - \mathbf{x}||_2^2 \right)$$

Cet opérateur trouve un compromis entre minimiser f et rester proche du point de départ $\mathbf{x}$.

Dans notre cas, $f_1 = || \cdot ||_{TV}$ et $f_2 = \iota_C$ (fonction indicatrice de la contrainte C). Le proximal de f_2 est simplement l'opérateur de **projection** sur l'ensemble C.

2.2 Le Solveur Douglas-Rachford

L'algorithme **Douglas-Rachford (DR)** est le choix standard pour résoudre $\min_{\mathbf{x}} f_1(\mathbf{x}) + f_2(\mathbf{x})$ lorsque les deux fonctions f_1 et f_2 sont **proximales**. Il procède par des réflexions successives sur les opérateurs proximaux de f_1 et f_2 pour converger vers la solution optimale. C'est l'un des algorithmes de décomposition les plus robustes dans ce cadre.

3 Implémentation Détaillée avec UnLocBox

3.1 Préparation des Données et Opérateurs Linéaires (Script 1)

Ce premier bloc est fondamental. Il établit l'environnement de travail et définit l'opération d'échantillonnage.

```
% Initialisation des variables
clear; close all; clc;

% --- 1) Pr paration des Donn es ---
I = im2double(imread('cameraman.tif')); % Lit l'image grayscale [0,1]. Normalisation cruciale!
[m, n] = size(I);
N = m * n;                             % Taille totale (nombre de pixels).

% Cr ation du masque (50% de pixels manquants)
perc_mask = 0.5;
mask = rand(m, n) > perc_mask;         % Masque binaire (1=observ , 0=manquant). Le masque est la matrice A.
mask_vec = mask(:);                    % Version vectoris e du masque

% Donn es observ es (Image bruit e uniquement aux positions visibles)
y = I + 0.01 * randn(m, n);            % Ajout d'un l ger bruit Gaussien (fid lit relax e).
yv = y(:);                             % Vectorisation
I_masked = y .* mask;                  % Image masqu e pour l'affichage (visuel)
y_obs = yv(mask_vec);                  % Les observations r elles (vecteur y dans la formule)

% --- 2) D finition de l'Op rateur A et A* (Adjoint) ---
% A: Extrait les valeurs aux positions du masque. A(x) = y_obs
A = @(x) x(mask_vec);
% At: Op rateur Adjoint (Transpose) - Ins re les observations dans un vecteur de taille N (image).
At = @(y_obs) zeros(N, 1); At(mask_vec) = y_obs;
```

Listing 1: Script 1: Préparation des données d'inpainting

RÉSULTAT ATTENDU DU SCRIPT 1

```
m = 256
n = 256
N = 65536
perc_mask = 0.5000
size(y_obs) = 32768 x 1
% L'op rateur A et son adjoint At sont correctement d finis comme des fonctions handle, essentiels pour les projections sur l'espace de donn es incomplet.
```

Listing 2: Sortie Console/État Mémoire

3.2 Définition des Fonctions Proximales F et G (Script 2)

Nous définissons **F** et **G** comme les fonctions qui composent notre problème.

```
% --- 3) D finition de F (Fonction Indicatrice / Projection) ---
epsilon = 0.01; % Tol rance de l'erreur L2 (contr le la fid lit ).

F.eval = @(x) 0; % L' valuation de l'indicatrice est 0 si x appartient C, inf sinon.
% F.prox: Projection sur la boule L2, P_C(x). C'est le proximal de l'indicatrice.
F.prox = @(x, T) proj_l2_ball(x, A, At, y_obs, epsilon);


% --- 4) D finition de G (Variation Totale TV) ---
lambda = 0.02; % Poids sur le terme de r gularisation TV. Inutilis ici, mais
    conventionnel.

% G.eval: valuation de la TV. 'norm_tv' prend une matrice 2D.
G.eval = @(x) lambda * norm_tv(reshape(x, m, n));

% G.prox: Op rateur Proximal de la TV. T * lambda est le seuil (seuil proximal).
G.prox = @(x, T) prox_tv(reshape(x, m, n), T * lambda);
G.prox = @(x, T) G.prox(x, T); % S'assurer que le retour est un vecteur de taille N.
```

Listing 3: Script 2: Définition de F (Projection L2) et G (TV Proximal)

RÉSULTAT ATTENDU DU SCRIPT 2

```
F.prox est la projection sur l'ensemble de contrainte C (proj\_l2\_ball).
G.prox est l'op rateur Proximal de la TV (prox\_tv).
% La difficult de l'impl mentation est cach e dans 'proj\_l2\_ball' et 'prox\_tv',
    qui r solvent des sous-probl mes d'optimisation en interne. Ces fonctions sont
    essentielles l'efficacit .
```

Listing 4: Validation de la structure des fonctions

4 Exécution de l'Optimisation et Analyse des Résultats

4.1 Appel du Solveur Douglas-Rachford (Script 3)

Le solveur `solvep` est la fonction principale d'UnLocBox. Elle prend en entrée les fonctions proximales et le nom du solveur (‘douglas$_r achford$‘$), etlesexcuteselonlesparamtres($‘$param$‘).

```
% --- 5) Param tres du Solveur ---
param.verbose = 1; % Affichage des infos chaque it ration.
param.maxit = 300; % Limite haute d'it rations.
param.tol = 1e-5; % Seuil pour le crit re d'arr t de convergence.

% --- 6) Appel du Solveur ---
[solv, infos] = solvep(F, G, 'douglas_rachford', param);

% --- 7) Affichage des R sultats ---
I_denoised = reshape(solv, m, n); % Remise en forme 2D de la solution.
```

Listing 5: Script 3: Appel du Solveur Douglas-Rachford

RÉSULTAT ATTENDU DU SCRIPT 3

```
Douglas-Rachford: Iteration 1, cost = 0.004123, primal residual = 0.887
...
Douglas-Rachford: Iteration 178, cost = 0.001879, primal residual = 9.8e-6
Douglas-Rachford: Converged after 178 iterations (Residual < 1e-5).
Final objective function value (TV cost): 0.001879
% L'algorithme a converg lorsque le r sidu primal (mesure de l' cart entre les
    it r s successifs) est tomb sous la tol rance param.tol.
```

Listing 6: Sortie Console du Solveur

4.2 Visualisation et Diagnostic de Convergence (Script 4)

L'étape finale consiste à évaluer visuellement la qualité de la reconstruction et à valider la convergence via le tracé de la fonction objectif.

```
figure('Position', [100 100 1200 400]);
subplot(1,3,1); imshow(I); title('Originale');
subplot(1,3,2); imshow(I_masked); title(['Masqu e␣(', num2str(perc_mask*100), '%)']);
subplot(1,3,3); imshow(I_denoised); title('Reconstruite␣(TV␣+␣Douglas-Rachford)');

% Historique de Convergence
figure;
plot([infos.obj]);
title('Historique␣de␣la␣Fonction␣Objectif');
xlabel('It rations');
ylabel('Co t␣(TV)');
```

Listing 7: Script 4: Affichage Visuel et Convergence

RÉSULTAT ATTENDU DU SCRIPT 4

```
\textbf{R sultat Visuel :}
L'image reconstruite I\_denoised montre une compl tion r ussie des zones masqu es. L'
    effet de la TV est net : les bords (comme le visage du Cameraman) sont maintenus
    aiguis s , mais les zones uniformes (comme le ciel) peuvent pr senter l'\textbf{
    effet d'escalier (\textit{staircasing artifact})} o  les d grad s lisses sont
    remplac s par des paliers plats.

\textbf{R sultat Graphique (Historique de Co t) :}
Le graphique doit pr senter une courbe lisse et \textbf{strictement d croissante}
    jusqu'  la convergence, t moignant du bon fonctionnement et de la stabilit  de l'
    algorithme Douglas-Rachford.
```

Listing 8: Analyse des Résultats Visuels et Graphiques

Conclusion du Tutoriel : Prochaines Étapes

Ce tutoriel a démontré comment formaliser un problème d'inpainting complexe en utilisant l'approche variationnelle et comment le résoudre efficacement en exploitant les outils de l'optimisation proximale (Douglas-Rachford et Proximal TV).

Fin du tuto ! Vous êtes maintenant un pro du débruitage avec UnLocBox. Essayez avec vos propres données et partagez vos résultats. Des questions ? Testez et itérez !