Restauration d'image par Variation Totale

SINGARIN-SOLE — BELGUIDOUM

Avril 2024

Contents

1 Contexte

Nous avons découvert au TP2 la méthode de résolution du modèle de la Variation Totale qui permet de déflouter une image en séparant la fonction de coût en somme de deux fonctions: une différentiable et une autre fonction non différentiable :

$$\hat{x} \in \arg\min_{x \in R^N} \|x - z\|_2^2 + \lambda \|Dx\|_1$$

où :

- $\hat{x}$ est la solution recherchée,
- z est l'observation,
- λ est un paramètre,
- D représente le Gradient, et
- $\|\cdot\|_2$ et $\|\cdot\|_1$ représentent respectivement la norme L^2 et la norme L^1.

On observe qu'elle a la particularité de contenir dans sa partie non différentiable un opérateur linéaire. Or, l'algorithme de gradient proximal qu'on avait utilisé plus tôt lors de ce TP, bien que permettant la minimisation de la somme de deux fonctions, n'est pas adapté lorsque la fonction non-différentiable contient un opérateur linéaire.

En effet, on ne peut pas calculer l'opérateur proximal de $\|\Gamma \cdot\|_1$. On avait dû alors considérer le problème dual, faisant intervenir la fonction conjuguée pour pouvoir résoudre le problème.

Dans le cadre de notre sujet "Restauration d'image par Variation Totale", on va explorer les subtilités du modèle de restauration d'image en termes de régularisation non lisse en intégrant un opérateur de flou, étendant ainsi les capacités du modèle de la Variation Totale classique. Cet opérateur qu'on note H intervient dans la formule ci dessous:

$$\hat{x} \in \operatorname*{argmin}_{x \in R^N} \|Hx - z\|_2^2 + \lambda \|Dx\|_1$$

avec H est l'opérateur de flou.

Ce léger changement va nous empêcher d'appliquer la méthode précédente dans la mesure où l'opérateur H, représentant un flou ou une autre forme de dégradation linéaire, complique le problème. En effet, le calcul de la conjuguée convexe, nécessaire pour formuler le problème dual, devient complexe en raison de la présence de ce dernier.

Notre objectif est donc de définir une méthode pour minimiser cette fonction tout en contournant ce problème.

2 Méthode utilisée et résolution mathématique

2.1 Alternated Direction Method of Multipliers (ADMM)

2.1.1 Impact de l'opérateur de flou sur le problème

Le dual d'un problème d'optimisation est une formulation alternative qui peut parfois être plus facile à résoudre que le problème primal.

Dans le cas de la restauration d'image par variation totale, si nous n'avions pas de flou, c'est-à-dire pas d'opérateur H, le terme d'attache aux données serait simplement $\|x - z\|_2^2$ et nous pourrions définir une variable duale associée à Dx (où D est l'opérateur différentiel), permettant ainsi une séparation des variables dans le dual.

Cependant, l'ajout de l'opérateur H transforme notre terme en $\|Hx - z\|_2^2$, où H n'est pas l'identité ni un opérateur diagonal facilement inversible. Ce terme ne se sépare donc pas facilement dans le dual.

Comme dit précedemment, le calcul de la conjuguée convexe devient complexe et nous avons ainsi besoin d'implémenter une nouvelle méthode pour notre résolution.

2.1.2 Principe de la méthode ADMM

L'Alternated Direction Method of Multipliers consiste à séparer problème difficile à résoudre en deux sous problèmes plus faciles à calculer.

Ainsi, on introduit une variable auxiliaire dans le problème primal et on utilise des techniques comme l'ADMM pour traiter la contrainte. On pose donc : $y = Dx$

Ce changement de variable permet de mettre le problème sous la forme :

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

Pour pouvoir réduire cette fonction de coût on va adopter 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. Ainsi selon la méthode de la ADMM, on obtient le système d'équation suivant :

$$\begin{cases} E(x, y) = \|Hx - z\|_2^2 + \lambda \|y\|_1 + \frac{\mu}{2} \|y - Dx\|_2^2 \\ x_{k+1} = \text{argmin}_{x \in R^N} \, E(x, y_k) \\ y_{k+1} = \text{argmin}_{y \in R^{2N}} \, E(x_{k+1}, \text{y}) \end{cases}$$

2.2 Calcul de x_{k+1}

2.2.1 Méthode de descente des gradients

Pour calculer x_{k+1}, cela n'est pas compliqué. En effet, on obverse qu'une fois y_k fixé, nous obtenons une fonction totalement différentiable. Ainsi, il nous suffit de calculer son gradient afin de pouvoir lui appliquer, par la suite, une descente de gradient (convergente car étant convexe).

Calculons $\nabla(E(x, y_k)$:

$$\nabla E(x, y_k) = \nabla(\|Hx - z\|_2^2 + \lambda\|y\|_1 + \frac{\mu}{2}\|y_k - Dx\|_2^2) \tag{1}$$
$$= 2H^T(Hx - z) - \mu D^T(y_k - Dx) \tag{2}$$

Nous pouvons ainsi simplement utiliser ce gradient afin d'effectuer notre descente de gradient pour récupérer notre x, c'est-à-dire notre x_{k+1}.

2.2.2 Calcul du pas pour la méthode de descente de gradient

Le pas γ de la méthode de descente des gradients nous permet de trouver le minimum de la fonction avec un nombre plus ou moins grand de tour de boucle dans l'algorithme.

On peut calculer le gamma maximale pour une convergence optimale. On le note γ_{max}. Pour cela on détermine la constante de Lipschitz de la fonction de coût. Elle s'obtient par le $\sup(\nabla^2(E(x, y_k))$

Calculons cette dernière :

$$\sup(\nabla^2 E(x, y_k)) = \sup(\nabla^2(\|Hx - z\|_2^2 + \lambda\|y\|_1 + \frac{\mu}{2}\|y_k - Dx\|_2^2)) \tag{3}$$
$$= \sup(\nabla(2H^T(Hx - z) - \mu D^T(y_k - Dx))) \tag{4}$$
$$= \sup(2|||H|||^2 - \mu|||D|||^2) \tag{5}$$
$$= 2|||H|||^2 - \mu|||D|||^2 = LDf_x \tag{6}$$

On obtient alors :

$$\gamma_{max_x} = \frac{1}{LDf_x}$$

2.3 Calcul de y_{k+1}

2.3.1 Descente de gradient sur la partie différenciable de la fonction

Pour calculer y_{k+1}, on remarque que notre fonction coût n'est différentiable. Nous allons ainsi la diviser en deux fonction $f(y)$ et $g(y)$:

$$\begin{cases} E(x_{k+1}, y) = f(y) + g(y) \\ f(y) = \|Hx_{k+1} - z\|_2^2 + \frac{\mu}{2}\|y - Dx_{k+1}\|_2^2 \\ g(y) = \lambda\|y\|_1 \end{cases}$$

On va devoir faire appel à la méthode de l'algorithme du gradient proximal pour pouvoir minimiser nos 2 fonctions.

On obtient ainsi pour y_n avec n l'itération dans la boucle de descente de gradient :

$$prox_{f+g}(y_n) = prox_g(y_n - \gamma\nabla f(y_n))$$

avec γ le pas de la descente.

Calculons ainsi $\nabla f(y)$:

$$\nabla f(y) = \nabla(\|Hx_{k+1} - z\|_2^2 + \frac{\mu}{2}\|y - Dx_{k+1}\|_2^2) \tag{7}$$
$$= \nabla(\frac{\mu}{2}\|y - Dx_{k+1}\|_2^2) \tag{8}$$
$$= \frac{\mu}{2}(2y - 2Dx_{k+1}) = \mu(y - Dx_{k+1}) \tag{9}$$

2.3.2 Calcul du pas idéal pour la descente de gradient

Comme précedemment, nous pouvons calculer γ_{max} :

$$\sup(\nabla^2 f(y)) = \sup(\nabla^2(\|Hx_{k+1} - z\|_2^2 + \frac{\mu}{2}\|y - Dx_{k+1}\|_2^2)) \tag{10}$$
$$= \sup(\mu(y - Dx_{k+1})) \tag{11}$$
$$= \mu(y - Dx_{k+1}) \tag{12}$$
$$= \mu = LDf_y \tag{13}$$

On obtient alors :

$$\gamma_{max_y} = \frac{1}{LDf_y}$$

2.3.3 Calcul du proximal pour la partie non differentiable

Maintenant, il nous reste à calculer le proximal de g. Pour cela, nous définissons le sous gradient de de g en fonction du pas γ :

$$prox_g(y) = prox_{\lambda\|\cdot\|_1}(y) = \begin{cases} y - \gamma\lambda & \text{si } y \leq \gamma\lambda \\ y + \gamma\lambda & \text{si } y < -\gamma\lambda \\ 0 & \text{sinon} \end{cases}$$

Cette implémentation fonctionne correctement pour les vecteurs. Cependant, pour des images, il y a quelques compliquations. Il faut ainsi redéfinir notre proximal :

$$prox_g(y) = prox_{\lambda\|\cdot\|_1}(y) = max(min(0, y + \gamma\lambda), y - \gamma\lambda)$$

Cette égalité se démontre graphiquement.

Au départ, nous avons constaté une incohérence lorsque nous avons observé qu'aucun changement significatif n'était produit en faisant varier le paramètre λ, bien que nous aurions dû observer une différence notable pour des valeurs élevées de lambda. En réévaluant notre formule de point proximal, nous avons réalisé qu'elle ne dépendait pas de lambda, ce qui indiquait une erreur dans notre approche initiale. Nous avons donc révisé nos calculs pour aboutir à la formulation corrigée.

Ainsi, avec une descente de gradient, nous pouvons déterminer notre y, c'est à dire y_{k+1}.

3 Implémentation informatique via python

3.1 Explication brève du code

Dans cette partie, il n'est pas question de répéter le code du programme mais bien d'expliquer certains éléments essentielles à son bon fonctionnement.

Tout d'abord, on remarque qu'il est nécessaire de réaliser une descende de gradient sur x et une autre pour y. Cependant, cela complexifie grandement les calculs.

En effet si on pose N et le nombre d'itération de chaque descente tout en effectuant une boucle sur ces descentes afin de converger vers (x, y), on obtiendrait une complexité de $\mathcal{O}(2N^2)$. Et cela sans même prendre en compte les calculs de gradients et du proximal.

Ainsi, on remarque qu'il est inutile d'effectuer des boucles suplémentaires sur les deux descentes de gradients car l'algorithme permet sa convergence dans tout les cas si le pas est bien choisi. On se positionne alors avec une de complexité de $\mathcal{O}(N)$.

Une fois nos paramètres choisi, l'algorithme principal se code ainsi :

```
for k in range(Niter):
    # Calcul de xk+1
        X.append(X[k] - gamma_x*Gradient_1(X[k],z,Y[k],mu))

    # Calcul de yk+1
        Y.append(Y[k] - gamma_y*Gradient_2(Y[k],X[k+1],mu))
        Y[k+1] = prox_g(Y[k+1],coeffLambda,gamma_y)
Iapprox = X[-1]
```

Avec X et Y, des listes stockant nos x_k et y_k respectivement.

On peut également noter que pour retrouver le minimum ou le maximum d'une image, on utilise la bibliothèque *numpy* et on définit ainsi notre $prox_g(y)$ de cette manière :

```
def prox_g(y,coeffLambda,gamma):
    yGamma = np.ones(y.shape)*gamma
    return np.maximum(np.minimum(np.zeros(y.shape),y+yGamma),y-yGamma)
```

De plus, il faut faire attention à la dimension de y. En effet, $y \in R^{2N}$ et il faut donc le définir correctement.

3.2 Exemple de résultat

Nous pouvons ainsi effectuer quelques essais en choisissant paramètres différents :

Figure 1: Niter $= 500$ — $\mu = 1$ — $\lambda = 0.05$ — $\gamma = \frac{3}{4}\gamma_{max}$

Figure 2: Niter $= 500$ — $\mu = 20$ — $\lambda = 1000$ — $\gamma = \frac{1}{2}\gamma_{max}$

On remarque des modifications du résultat final en fonction des paramètres choisis. En effet, sur la première image, l'estimation est nettement meilleure.

On peut également tester la performance du programme sur d'autres images avec des paramètres similaires :

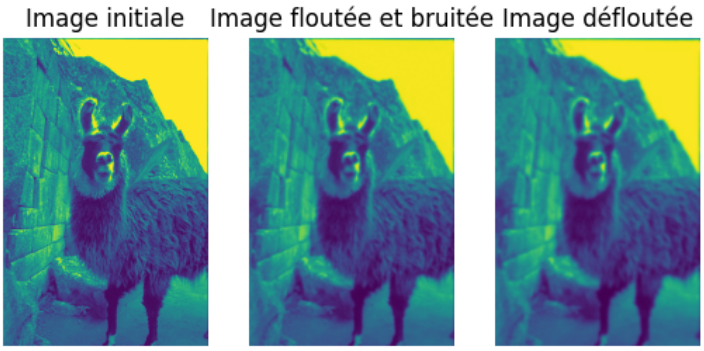

Figure 3: Niter = 200 — $\mu = 1000$ — $\lambda = 0.05$ — $\gamma = \frac{1}{2}\gamma_{max}$

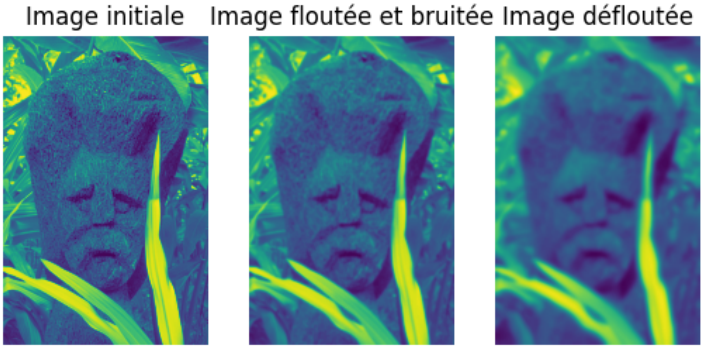

Figure 4: Niter = 200 — $\mu = 1000$ — $\lambda = 1000$ — $\gamma = \frac{1}{2}\gamma_{max}$

On peut déja conjecturer que si λ ou μ est trop grand, le filtrage sera moins efficace.

4 Etude de l'influence des paramètres

Dans cette section, nous allons nous pencher sur l'influence des différents paramètres. Réalisons tout d'abord une liste des paramètres modifiables :

- $Niter$ le nombre d'itération
- N la dimension du noyau de h (pour le flou)
- λ, paramètre de régularisation
- μ, paramètre de régularisation
- γ_x et γ_y
- x_0 et y_0

On note que la dimension du noyau de h associé au flou n'est pas très importante ici donc nous allons la garder constante (N = 9 car doit être impaire). Ensuite, x_0 et y_0 vont juste diminuer le temps convergence et sont propres à chaque image donc il n'est pas rigoureux d'étuduier leur influence ici. Enfin, le nombre d'itération et les pas auront également un rôle similaire donc nous les gardons constants, c'est à dire Niter = 20 , $\gamma_x = \frac{1}{2}\gamma_{max_x}$, $\gamma_y = \frac{1}{2}\gamma_{max_y}$. Il nous reste donc à étudier l'influence de λ et μ.

Nous allons maintenant étudier l'influence de λ et μ en utilisant trois types d'erreurs : MSE,SNR et SSIM.

Le SNR (Rapport Signal sur Bruit) et la MSE (Erreur Quadratique Moyenne) évaluent les différences entre une image d'origine et une image modifiée. Une faible MSE, signifiant peu de différences entre les images, entraîne généralement un SNR élevé, indiquant que le signal prédomine sur le bruit.

D'autre part, la MSE mesure l'erreur pixel par pixel, tandis que le SSIM (Similarité Structurelle) prend en compte la structure, le contraste et la luminance pour refléter la perception humaine de la qualité de l'image. Ainsi, une faible MSE ne garantit pas nécessairement un bon score SSIM si la structure ou le contraste de l'image est altéré pour la perception humaine.

4.1 Influence de λ

Faisons ainsi varier λ tout en fixant μ et observons les résultats.

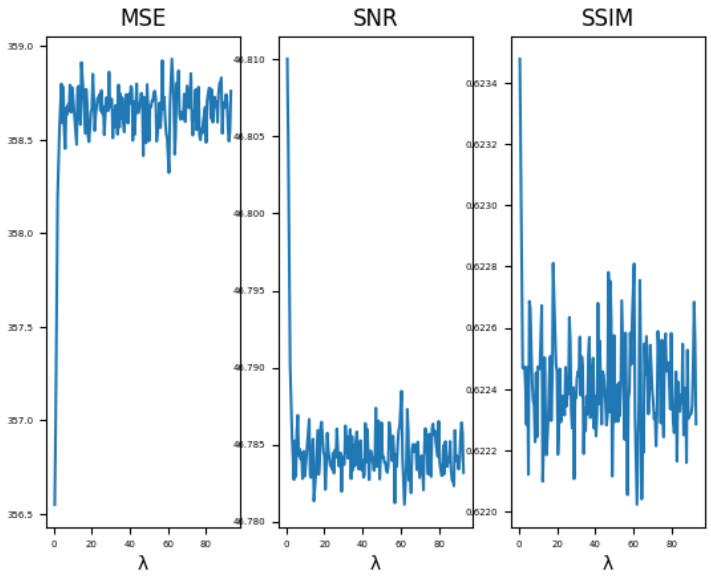

Figure 5: $\mu = 0.1$

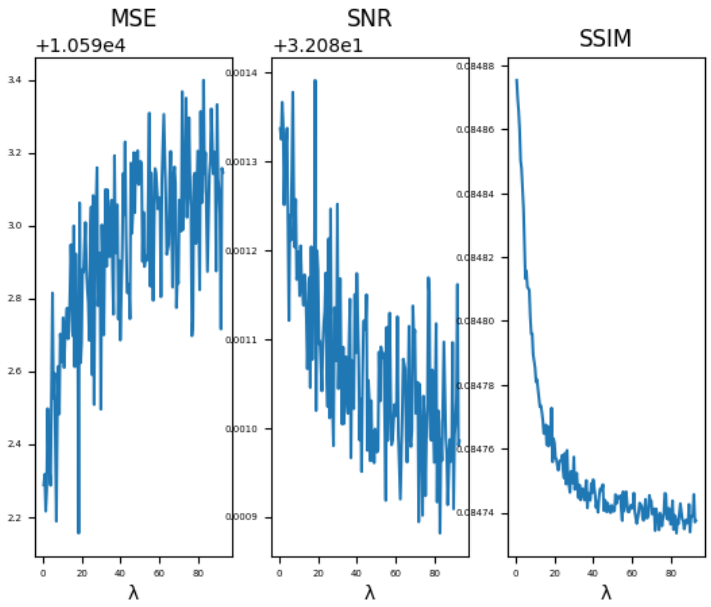

Figure 6: $\mu = 1000$

On remarque que pour des valeurs petites l'erreur est faible mais lorsque les valeurs sont grandes, l'erreur augmente. Cependant, on remarque une oscillation de l'erreur assez importante lorsque μ aug-

mente. On conclut ainsi que choisir un petit λ est plus avantageux. Mais, si le λ est suffisamment grand, le diminuer ou l'augmenter modifie l'erreur de manière pseudo-aléatoire.

4.2 Influence de μ

Maintenant, faisons varier μ tout en fixant λ et observons les résultats.

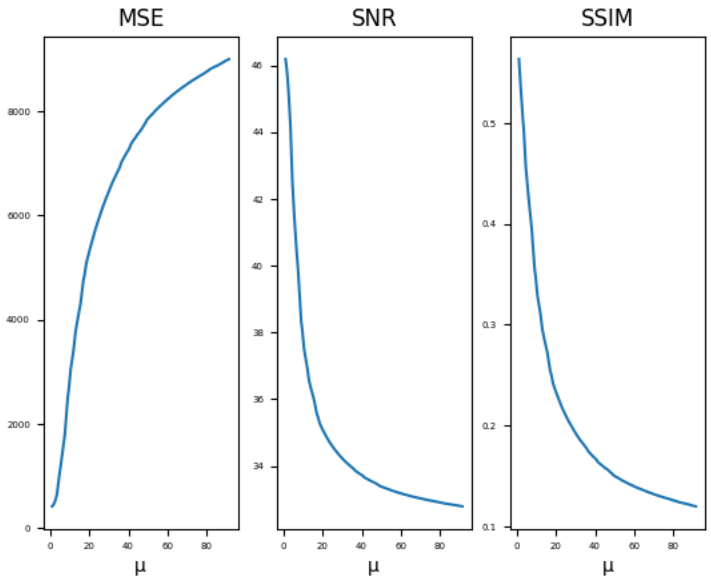

Figure 7: $\lambda = 0.1$

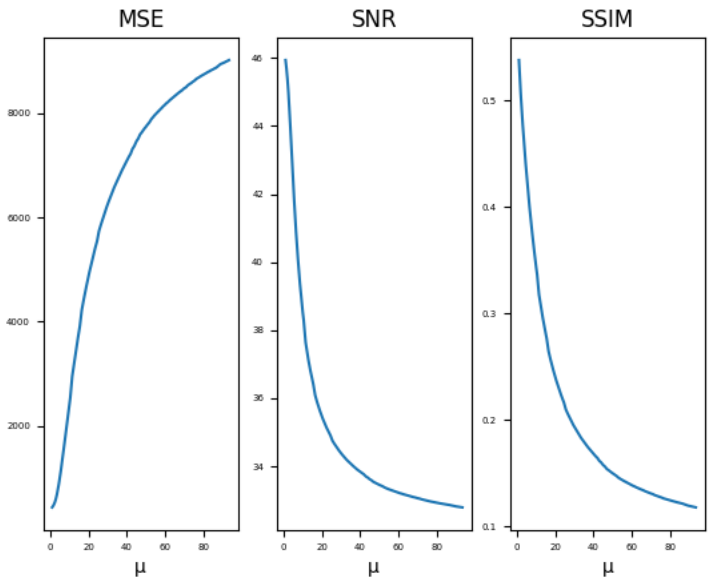

Figure 8: $\lambda = 1000$

Contrairement à précedemment, on obtient des courbes plus lisses. On observe bien que plus μ augmente, plus l'erreur augmente.

En définitive, pour obtenir une convergence optimale, il est nécessaire d'utiliser des paramètres petits pour λ et μ. C'est un résultat attentu dans la mesure où ces derniers sont des termes de régularisation et diminue l'attache à la donnée lorsqu'ils augmentent.

5 Conclusion

Au cours de notre projet sur la restauration d'images par Variation Totale, nous avons exploré en profondeur l'importance de l'optimisation dans le traitement d'images. En travaillant avec le modèle de variation totale et en ajustant les paramètres et µ, nous avons découvert l'équilibre crucial entre régularisation et fidélité aux données. L'utilisation de l'ADMM a présenté des défis intéressants, mais elle nous a aussi permis de mieux comprendre les techniques d'optimisation et leur application pratique.

6 Références

References

[1] Laure Blanc Féraud, *Approche variationnelle pour la restauration d'image*

[2] mastermas.univ-lyon1.fr, *Majeure Mathématiques, Image, Data*

[3] Khalid JALALZAI Antonin CHAMBOLLE, *Restauration d'images floutées et bruitées par une variante originale de la variation totale*