

Super-resolved image reconstruction

Yanis Gomes^{1,†}

¹Ecole Normale Supérieure de Paris-Saclay

May 1, 2024

Abstract

Le rapport étudie les techniques de super-résolution pour améliorer la résolution d'image au-delà des capacités des détecteurs en utilisant plusieurs images de basse résolution. Il aborde le problème inverse courant dans les systèmes de mesure consistant à inverser les déficiences des images de basse résolution pour améliorer les détails et la netteté. Les techniques incluent l'estimation des déplacements d'image pour les aligner et les superposer afin de créer une composition de plus haute résolution, suivie d'une déconvolution pour réduire le flou induit par le capteur. L'étude applique également les méthodes des moindres carrés et des moindres carrés régularisés pour minimiser les erreurs de reconstruction, avec des simulations MATLAB confirmant des améliorations substantielles de la qualité d'image. Ces résultats valident l'utilité de la super-résolution dans les applications nécessitant une résolution d'image améliorée à partir de données limitées.

Keywords: Super-résolution, Déconvolution, Moindres carrés, Régularisation

■ Introduction

Dans ce TP nous nous intéresserons à la reconstruction d'une image sur-résolue c'est-à-dire avec une résolution supérieur à celle du détecteur à partir de n images basse résolution. Ce problème entre dans la classe des problèmes inverses qui peuvent se poser lors de l'utilisation de systèmes de mesure réels. Le but de la sur-résolution est d'inverser le repliement présent dans les images basse résolution.

■ Estimation des décalages

L'écart entre les images est inconnu mais supposé faible, on peut donc estimer le décalage en cherchant à maximiser la ressemblance entre les images. Le critère de ressemblance utilisé est la corrélation car il est insensible aux variations d'intensité.

$$\text{cor} = \frac{(\mathbb{E}\{I_{m1} - \mathbb{E}\{I_{m1}\}\})(\mathbb{E}\{I_{m2} - \mathbb{E}\{I_{m2}\}\})}{\text{std}(I_{m1}) \text{std}(I_{m2})}$$

avec $\mathbb{E}\{x\}$ la moyenne et $\text{std}(x)$ l'écart type de x . On prend pour référence l'image suivante, la première de la série :

Nous étudierons l'image suivante dans le cadre de notre travail.



Figure 1. Image de référence

On commence par utiliser un algorithme qui calcule le maximum de la corrélation entre une image de référence et une image à laquelle on applique des décalages de façon itérative.

Algorithme

- Fixer la fenêtre d'exploration des décalages et du pas de recherche.
- Pour chaque décalage selon l'abscisse et l'ordonnée faire
 - Calculer la corrélation
 - Enregistrer les déplacements pour lesquels la corrélation est maximale.

Résultat

dx_max	dy_max	$corr_max$
0	0	0
0.0002	0.0008	9.7121
0.0004	0.0006	9.6052
0.0006	0.0008	9.6504
0	0.0006	9.7007
0	0.0004	9.6988
0	0.0008	9.7561
0.0004	0.0004	9.6025
0.0008	0.0008	9.7088
0.0006	0.0004	9.5969
0.0002	0	9.7549
0.0002	0.0006	9.6600
0.0004	0.0002	9.6516
0.0006	0.0002	9.6482
0.0008	0	9.7512

Première méthode : décalage et addition

On suppose l'échantillonneur idéale, on a donc l'équation suivante :

$$\mathbf{y}_{[1n]} = \mathbf{SD}_{[1n]} \mathbf{Rx} + \mathbf{b}$$

et l'estimateur des moindre carrés associé :

$$\hat{\mathbf{x}}_{MC} = \left(\mathbf{R}' \mathbf{D}_{[1n]}^t \mathbf{S}' \mathbf{SD} \right)^{-1} \mathbf{R}' \mathbf{D}_{[1n]}^t \mathbf{S}' \mathbf{y}_{[1n]}$$

Avec \mathbf{S} la matrice de sous-échantillonage, \mathbf{D} la matrice de ressemblance entre tous les décalages et \mathbf{R} la matrice de réplication permettant d'obtenir n images haute résolution.

Algorithme

- Générer la matrice de décalage à l'aide du code précédent
- Pour chaque image :
 - Créer une image sur-résolue à l'aide du pas de décalage utilisé précédemment.

- Affecter les valeurs des images aux bonnes places dans l'images sur-résolue en utilisant les décalages estimés.
- Générer une image de pondération qui a la même taille que l'image sur-résolue. On incrémente alors cette image de pondération à chaque fois que l'on ajoute une donnée à l'image sur-résolue
- Diviser terme à terme l'image obtenue par l'image de pondération

```

1 % Initialize the super-resolved image and
2 % the weight image
3 im_ref = data(:,:,1);
4 super_resolved_image = zeros(size(data, 1),
5 size(data, 2));
6 super_resolved_images = zeros(size(data, 1)
7 .* (1/pas), size(data, 2).* (1/pas), 15);
8 weight_image = zeros(size(data, 1).* (1/pas),
9 size(data, 2).* (1/pas));
10
11 % For each image
12 for n = 1:size(data, 3)
13     for i = 1:size(data, 1)
14         for j = 1:size(data, 2)
15             x = 1 + (i-1)/pas + tab_decalages
16             (n, 1)/pas;
17             y = 1 + (j-1)/pas + tab_decalages
18             (n, 2)/pas;
19             % Create a super-resolved image
20             % using the previously used shift step
21             super_resolved_images(x,y,n) =
22             data(i,j,n);
23             % Generate a weight image that
24             % is the same size as the super-resolved image
25             %
26             % Increment this weight image
27             % each time a data is added to the super-
28             % resolved image
29             weight_image(x,y) = weight_image
30             (x,y) + 1;
31         end
32     end
33
34 % Divide term by term the obtained image by
35 % the weight image
36 super_resolved_image = sum(super_resolved_
37 images, 3) ./ weight_image;
38
39 % Display the reference image and the super-
40 % resolved image side by side
41 figure;
42 subplot(1, 2, 1);
43 imshow(im_ref, []);
44 title('Reference Image');
45 subplot(1, 2, 2);
46 imshow(super_resolved_image, []);
47 title('Super-Resolved Image');

```

Code 1. Super-résolution x3

Résultat

Déconvolution

On peut permuter la convolution et le décalage car ce sont des opérations invariantes. Une fois que l'on a obtenu le résultat de l'approche shift and add (addition et décalage), on peut déconvoluer ce résultat par la réponse du capteur basse résolution. Notons par \mathbf{y} l'estimation obtenue dans la partie précédente, le modèle direct se formule donc de la façon suivante :

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{b}$$

avec

- \mathbf{y} sont les données du capteur (images floues)

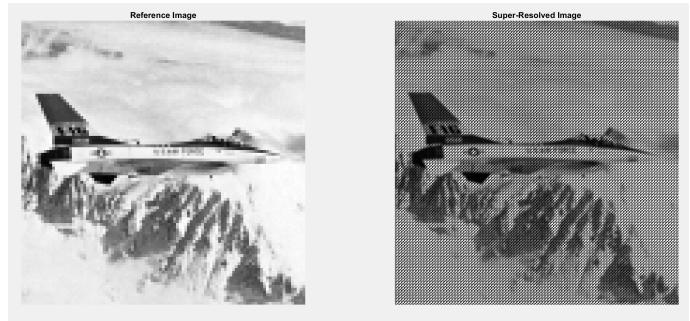


Figure 2. Image super-résolue avec un facteur 5

- \mathbf{x} l'image nette (haute résolution)
- \mathbf{C} la matrice de convolution ayant une structure de Toeplitz
- \mathbf{b} le bruit additionnel

La matrice \mathbf{C} est une matrice de convolution elle possède la structure classique suivante :

$$\mathbf{C} = \begin{bmatrix} c_p & c_{p-1} & \dots & \dots & c_1 & 0 & \dots & \dots & 0 \\ 0 & c_p & c_{p-1} & \dots & \dots & c_1 & 0 & \dots & 0 \\ 0 & 0 & c_p & c_{p-1} & \dots & \dots & c_1 & \dots & 0 \\ & & & \ddots & \ddots & \ddots & & \ddots & \\ 0 & \dots & \dots & 0 & c_p & c_{p-1} & \dots & \dots & c_1 \\ 0 & 0 & \dots & \dots & 0 & c_p & c_{p-1} & \dots & c_2 \\ 0 & 0 & 0 & \dots & \dots & 0 & c_p & \dots & c_3 \\ & & & & & & \ddots & \ddots & \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & \dots & 0 & c_p \end{bmatrix}.$$

C'est une matrice carrée, de dimension N si on dispose de N observations et qui possède une structure de Toeplitz. Le problème est donc maintenant, d'estimer le vecteur \mathbf{x} à partir du vecteur de données \mathbf{y} et de la connaissance de la matrice \mathbf{C} .

Moindres carrés

La technique la plus simple pour résoudre ce problème consiste à minimiser le critère de moindres carrés :

$$Q_{MC}(\mathbf{x}) = (\mathbf{y} - \mathbf{C}\mathbf{x})^t(\mathbf{y} - \mathbf{C}\mathbf{x}).$$

Son minimum est :

$$\hat{\mathbf{x}} = (\mathbf{C}^t \mathbf{C})^{-1} \mathbf{C}^t \mathbf{y} \quad (1)$$

l'estimée de \mathbf{x} au sens des moindres carrés. La matrice $\mathbf{C}^t \mathbf{C}$ à inverser est de taille $N \times N$ et si N est grand (comme ça peut être le cas en 2D pour une image) l'inversion directe est difficile pour des questions de coût de calcul. Il existe plusieurs méthodes rapides pour inverser cette matrice et calculer $\hat{\mathbf{x}}$.

Résolution par approximation circulante On commence tout d'abord par faire l'hypothèse que l'on peut «remplacer» les matrices apparaissant dans (1) par des matrices diagonales. Les opérations matricielles se simplifient alors grandement. De plus, la «transformation» de ces matrices en matrices diagonales est réalisée par FFT. Pour cela on tolère une approximation circulante pour la matrice de convolution. Il s'agit de remplacer \mathbf{C} par $\tilde{\mathbf{C}}$:

$$\tilde{\mathbf{C}} = \begin{bmatrix} c_p & c_{p-1} & \dots & \dots & c_1 & 0 & \dots & \dots & 0 \\ 0 & c_p & c_{p-1} & \dots & \dots & c_1 & 0 & \dots & 0 \\ 0 & 0 & c_p & c_{p-1} & \dots & \dots & c_1 & \dots & 0 \\ & & & \ddots & \ddots & & & \ddots & \\ 0 & \dots & \dots & 0 & c_p & c_{p-1} & \dots & \dots & c_1 \\ c_1 & 0 & \dots & \dots & 0 & c_p & c_{p-1} & \dots & c_2 \\ c_2 & c_1 & 0 & \dots & \dots & 0 & c_p & \dots & c_3 \\ & \ddots & \ddots & & & & \ddots & \ddots & \\ c_{p-1} & \dots & c_2 & c_1 & 0 & \dots & \dots & 0 & c_p \end{bmatrix}$$

Cette approximation consiste à ajouter des termes en bas et à gauche de la matrice \mathbf{C} de façon à la rendre circulante. A noter que cette approximation est d'autant meilleure que N et P sont grand, i.e. que la taille du signal et la longueur de la réponse impulsionnelle sont grandes.

La matrice $\tilde{\mathbf{C}}$ étant circulante, on peut la diagonaliser dans la base de Fourier :

$$\tilde{\mathbf{C}} = \mathbf{W}^\dagger \Lambda_c \mathbf{W}. \quad (2)$$

La matrice Λ_c est diagonale et ses éléments diagonaux sont les valeurs propres $\lambda_c^1, \lambda_c^2, \dots, \lambda_c^N$ de \mathbf{C} . Celles-ci s'obtiennent très facilement comme FFT de la première ligne de \mathbf{C} c'est-à-dire comme FFT de la réponse impulsionnelle calculée sur N points. Autrement dit :

$$\lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \end{bmatrix} = \mathbf{W} \begin{bmatrix} c_1 \\ \vdots \\ c_p \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Partant de la relation (1), en exploitant la relation de diagonalisation (2) et les relations sur la matrice \mathbf{W} contenues dans votre cours, on obtient :

$$\begin{aligned} \hat{\mathbf{x}} &= (\tilde{\mathbf{C}}^t \tilde{\mathbf{C}})^{-1} \tilde{\mathbf{C}}^y \mathbf{y} \\ &= (\mathbf{W}^\dagger \Lambda_c \mathbf{W} \mathbf{W}^\dagger \Lambda_c \mathbf{W})^{-1} \mathbf{W}^\dagger \Lambda_c^\dagger \mathbf{W} \mathbf{y} \\ &= \mathbf{W}^\dagger \Lambda_c^{-1} \mathbf{W} \mathbf{y} \end{aligned}$$

On en déduit alors, en multipliant à gauche par \mathbf{W} :

$$\mathbf{W} \hat{\mathbf{x}} = \Lambda_c^{-1} \mathbf{W} \mathbf{y}$$

On pose enfin $\hat{\mathbf{X}} = \mathbf{W} \hat{\mathbf{x}}$ et $\mathbf{Y} = \mathbf{W} \mathbf{y}$ les transformées de Fourier $\hat{\mathbf{x}}$ et \mathbf{y} . Notez que $\hat{\mathbf{X}}$ et \mathbf{Y} s'obtiennent eux aussi comme FFT de $\hat{\mathbf{x}}$ et de \mathbf{y} . Le problème, dans le domaine de Fourier, devient simplement :

$$\hat{\mathbf{X}} = \Lambda_c^{-1} \mathbf{Y}$$

Pour terminer, on construit le vecteur \mathbf{g}_{MC} des valeurs de la diagonale de Λ_c^{-1} :

$$g_{MC}^n = 1/\lambda_n \text{ pour } n = 1, 2, \dots, N$$

On obtient alors le vecteur $\hat{\mathbf{X}}$ comme le produit «éléments par éléments» (noté *) des deux vecteurs \mathbf{g}_{MC} et \mathbf{Y} :

$$\hat{\mathbf{X}} = \mathbf{g}_{MC} * \mathbf{Y}.$$

Remarquez que cette relation entre \mathbf{Y} (la TF de \mathbf{y}) et $\hat{\mathbf{X}}$ (la TF de $\hat{\mathbf{x}}$) est une relation de «filtrage dans le domaine de Fourier» dont la fonction de «transfert en fréquence discrète» est \mathbf{g}_{MC} .

Algorithme

1. Construire λ_c la FFT de la réponse impulsionnelle sur N points.
2. Construire le vecteur \mathbf{g}_{MC} du transfert en fréquence selon (17).
3. Construire \mathbf{Y} la FFT des données.
4. Calculer $\hat{\mathbf{X}}$ comme produit du transfert en fréquence \mathbf{g}_{MC} par \mathbf{Y} .
5. Revenir dans le domaine spatial par FFT inverse de $\hat{\mathbf{X}}$ pour obtenir $\hat{\mathbf{x}}$.

```

5     lambda = fft2(c, N(1), N(2));
6
7     gmc = lambda .^-1;
8
9     X_hat = gmc.*Y;
10    img_rec = ifft2(X_hat);
11    img_rec = reshape(img_rec, s1(1), s1(2));

```

Code 2. Moindres carrés

Le problème des NaN On constate que la matrice super résolue contient des NaN. Cela est simplement du au fait qu'il existe plus de pixels dans la matrice élémentaire de pixels de l'image sur-résolue : matrice 5x5 = 25 pixels ; que d'image différentes issue de l'image de référence. Il existe alors des pixels qui ne sont jamais comblés par une image déclées, d'où la présence de NaN. Pour résoudre ce problème on utilise simplement une super-résolution avec un pas plus faible. Un pas de 3 fonctionne vraisemblablement pour qu'il n'y ait plus de NaN dans la matrice super-résolue.

Calcul des décalages On recommence avec un pas de 3.

0	0	0
0.0003	0.0007	9.6457
0.0003	0.0007	9.5974
0.0007	0.0007	9.6185
0	0.0007	9.6969
0	0.0003	9.6960
0	0.0007	9.7246
0.0003	0.0003	9.5956
0.0007	0.0007	9.6370
0.0007	0.0003	9.5947
0.0003	0	9.7200
0.0003	0.0007	9.6229
0.0003	0	9.6230
0.0007	0	9.6242
0.0007	0	9.7126

Figure 3. Décalages et corrélations pour un pas de 3

Image super-résolue On détermine l'image super-résolue avec un pas de 3

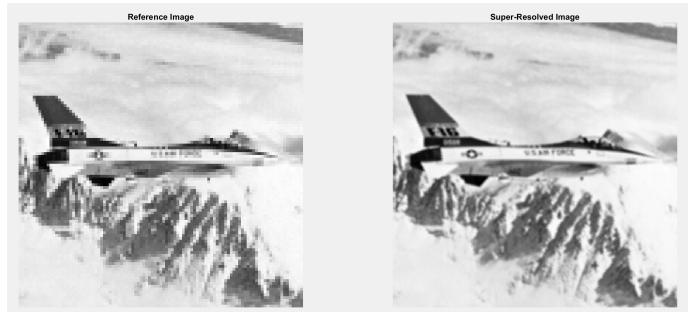


Figure 4. Image super résolue avec un pas de 3

Calcul des moindres carrés On peut alors calculer les moindres carrés car il n'y a plus de NaN dans l'image à laquelle on applique la transformée de Fourier.

```

1     y = image
2     Y = fft2(y);
3     N = size(Y);
4

```

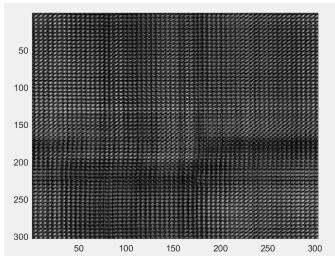


Figure 5. Image super résolue avec un pas de 3

Remarque On remarque qu'avec les moindres carrés le résultat n'est pas directement exploitable. On peut expliquer ce résultat par le fait que la transformée de Fourier du capteur est une porte. Elle donne donc un sinus cardinal auquel s'ajoute du bruit. Le bruit s'ajoute au niveau des noeuds du sinus cardinal de tel sorte que cela finit par noyer totalement le signal.

Solution On peut faire varier le facteur μ régularisant. On constate que cela permet de lisser le résultat en évitant les grands écarts entre deux échantillons successifs. Avec un μ trop faible, le rapport signal sur bruit n'est pas satisfaisant. A l'inverse quand μ est trop élevé, on coupe les haute fréquences de l'image ce qui donne un effet flou au résultat. Le facteur μ traduit le compromis. Le facteur μ permet de pénaliser les gros écarts entre les données consécutives et donc de lisser l'image estimée afin d'avoir un résultat exploitable.

Régularisation et douceur

On constate que l'image obtenue n'est pas d'autant satisfaisante. L'image obtenue reste très bruitée, voire totalement inexploitable.

Quoiqu'il en soit, pour corriger ce défaut il est possible de modifier le critère des moindres carrés afin d'assurer une certaine régularité à l'image obtenue. L'idée la plus simple consiste à pénaliser les fortes différences entre échantillons successifs du signal à restaurer, afin de d'améliorer le conditionnement de la matrice. Le critère prend alors la forme régularisée suivante :

$$\begin{aligned} Q_{\text{MCR}}(\mathbf{x}) &= (\mathbf{y} - \mathbf{Cx})^t (\mathbf{y} - \mathbf{Cx}) + \mu \sum_{n=1}^{N-1} (x_{n+1} - x_n)^2 \\ &= (\mathbf{y} - \mathbf{Cx})^t (\mathbf{y} - \mathbf{Cx}) + \mu \mathbf{x}^t \mathbf{D}^t \mathbf{D} \mathbf{x} \end{aligned}$$

avec \mathbf{D} la matrice de différences d'ordre 1, de taille $(N - 1) \times N$, définie par :

$$\mathbf{D} = \begin{bmatrix} 1 & -1 & 0 & 0 & \dots & \dots & 0 \\ 0 & 1 & -1 & 0 & \dots & \dots & 0 \\ \vdots & & \ddots & \ddots & & & \vdots \\ \vdots & & & \ddots & \ddots & & \vdots \\ 0 & \dots & \dots & 0 & 1 & -1 & 0 \\ 0 & \dots & \dots & 0 & 0 & 1 & -1 \end{bmatrix}$$

Le critère obtenu est quadratique en \mathbf{x} comme celui des moindres carrés. On connaît de plus l'expression de son minimum :

$$\hat{\mathbf{x}} = (\mathbf{C}^t \mathbf{C} + \mu \mathbf{D}^t \mathbf{D})^{-1} \mathbf{C}^t \mathbf{y}. \quad (3)$$

■ Moindres carrés

Algorithme de descente selon le gradient

Contexte

Technique de minimisation des moindres carrés :

$$Q_{\text{MC}}(\mathbf{x}) = (\mathbf{y} - \mathbf{TCx})^t (\mathbf{y} - \mathbf{TCx}). \quad (4)$$

On en déduit l'expression de l'estimée de \mathbf{x} au sens des moindres

carrés qui correspond au minimum de la fonction quadratique :

$$\mathbf{x} = (\mathbf{C}^t \mathbf{T}^t \mathbf{T} \mathbf{C})^{-1} \mathbf{T}^t \mathbf{C}^t \mathbf{y} \quad (5)$$

Remarque La matrice $\mathbf{C}^t \mathbf{T}^t \mathbf{T} \mathbf{C}$ à inverser est de taille $N \times N$ or pour N grand l'inversion directe est couteuse. On se propose d'utiliser une procédure itérative pour calculer $\hat{\mathbf{x}}$ sans inverser de matrice.

Algorithme de descente selon le gradient

Comme le critère est quadratique, il est convexe et ne possède qu'un minimum global (2). Pour trouver le minimum du critère, quelque soit le point de départ on cherche la direction de plus forte pente. Si on recommence l'opération un nombre de fois suffisant on tend vers le minimum du critère.

Calcul du gradient du critère par rapport à \mathbf{x}

$$\begin{aligned} \frac{\partial Q_{\text{MC}}(\mathbf{x})}{\partial \mathbf{x}} &= 2 \mathbf{C}^t \mathbf{T}^t \mathbf{T} \mathbf{C} \mathbf{x} - 2 \mathbf{C}^t \mathbf{T}^t \mathbf{y} \\ &= 2 \mathbf{C}^t \mathbf{T}^t (\mathbf{T} \mathbf{C} \mathbf{x} - \mathbf{y}) \end{aligned}$$

Notons par $e_k = (\mathbf{T} \mathbf{C} \mathbf{x} - \mathbf{y})$, l'erreur entre la sortie modèle et les données.

L'algorithme de descente On prend un \mathbf{x}^0 quelconque puis on applique l'itération suivante :

$$\begin{aligned} \mathbf{x}^{k+1} &= \mathbf{x}^k - \alpha \mathbf{C}^t \mathbf{T}^t (\mathbf{T} \mathbf{C} \mathbf{x}^k - \mathbf{y}). \\ \text{Si } \alpha < \frac{2}{\|\mathbf{C}^t \mathbf{T}^t \mathbf{T} \mathbf{C}\|_M}, \end{aligned}$$

L'algorithme converge alors vers le minimum du critère. On note par $\|\cdot\|_M$ la norme d'une matrice. Cette norme est égale au module au carré de la plus grande des valeurs singulières.

Pas de descente optimum Si on pose : $\frac{\partial Q_{\text{MC}}(\mathbf{x}^k)}{\partial \mathbf{x}} = \mathbf{g}_k$, on peut facilement démontrer que le scalaire α_{opt} qui permet de descendre le plus bas possible dans la direction \mathbf{g}_k est égale :

$$\alpha_{\text{opt}} = \frac{\mathbf{g}_k^t \mathbf{g}_k}{2 \mathbf{g}_k^t (\mathbf{C}^t \mathbf{T}^t \mathbf{T} \mathbf{C}) \mathbf{g}_k} \quad (6)$$

Algorithme

1. Initialiser \mathbf{x}^0 à une valeur quelconque.
2. Calculer $\mathbf{C} \mathbf{x}^k$ en convoluant \mathbf{x}^k par la réponse impulsionnelle.
3. Puis on tronque le résultat.
4. Calculer l'erreur $\mathbf{e}_k = (\mathbf{T} \mathbf{C} \mathbf{x}^k - \mathbf{y})$.
5. L'opération $\mathbf{T}^t \mathbf{e}_k$ consiste à mettre la valeur lorsqu'elle a été observée et zéro sinon.
6. Calculer $\mathbf{C}^t \mathbf{T}^t (\mathbf{T} \mathbf{C} \mathbf{x}^k - \mathbf{y})$ en convoluant le résultat de l'étape précédente par la RI retournée.
7. Multiplier le résultat de l'étape précédente par α et le soustraire à \mathbf{x}^k .
8. Retourner à l'étape 2 tant que l'algorithme n'a pas convergé, (Retourner à l'étape 2 tant que la norme du gradient est supérieure à un seuil que l'on fixera arbitrairement).

Implémentation avec un pas fixe

```

1 load Donnees1;
2 load superResx5.mat
3
4 % Initialize the image to restore
5 x = ones(size(super_resolved_image, 1), size
(super_resolved_image, 2));
6 y = super_resolved_image;
```

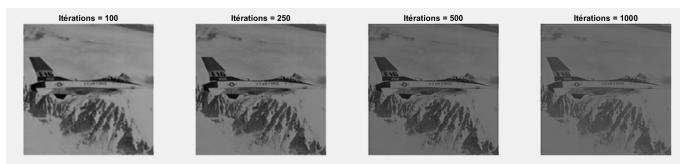
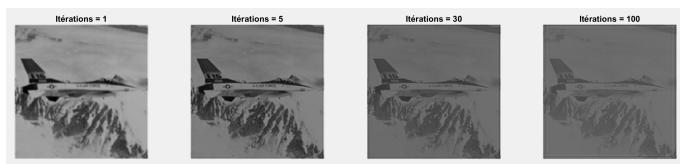
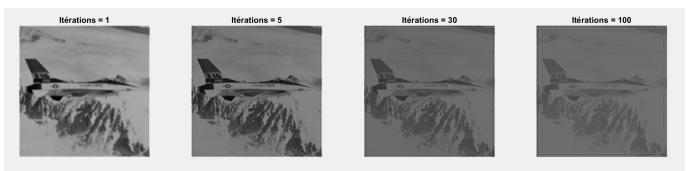
```

7
8 T = ones(size(super_resolved_image, 1), size
9 (super_resolved_image, 2));
C = ones(size(super_resolved_image, 1), size
10 (super_resolved_image, 2));
11
12 % Initialize the threshold
13 threshold = 0.5;
14
15 % Initialize the opt step
16 alpha = 0.9; % 0.015 ; 1.03
17
18 % MU to avoid edge effects
19 mu = 0.5;
20
21 % Normalisation de la reponse impulsionale
22 RI = RI/sum2(RI);
23
24 % While the norm of the gradient is greater
25 % than the threshold
26 for iter = 1:100
    % Calculer Cx^(k) en conciliuant x^(k)
    % par la RI
    H = conv2(x, RI, 'same');
    % Calcul de l'erreur
    e = H-y;
    % On tronque le resultat
    e(isnan(e)) = 0;
    % Convolution
    g = conv2(e, RI, 'same');
    % Descente de gradient avec un pas fixe
    x = x - alpha*g;
    figure(2);
    imshow(x, []);
    drawnow;
    title('Restored Image');
end

```

Code 3. Gradient à pas fixe

Observation On constate que l'algorithme de descente selon le gradient ne converge pas quand on ne normalise pas la réponse impulsionale. En évitant la divergence par la normalisation on observe grâce à **drawnow** l'évolution de l'image restaurée au fur et à mesure des itérations. Ce que l'on constate c'est que l'image initialement bruitée est déconvolue pas à pas. En revanche des effets de bords sont introduits par la convolution et la déconvolution. Ces effets de bord sont noir ou blanc ce qui a pour conséquence directe de dégrader la dynamique de l'image. Cela explique qu'on obtienne une image grisée à la fin de l'algorithme avec des effets de bords noirs et blancs. Plus les itérations progressent plus l'image est nette mais grisée.

**Figure 6.** Image restaurée avec $\alpha = 0.015$ **Figure 7.** Image restaurée avec $\alpha = 1.03$ **Figure 8.** Image restaurée avec $\alpha = 0.9$

Solution Pour éviter les effets de bords et empêcher la dégradation de la dynamique de l'image, on peut utiliser d'autres paramètres de convolution. On ajuste ensuite les dimensions pour pouvoir exécuter les opérations matricielles et obtenir l'erreur. On utilise les paramètres **textit{'valid'}** puis **textit{'full'}** pour corriger le problèmes précédent. Le coeur de la boucle for devient alors

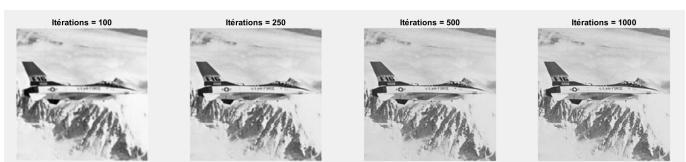
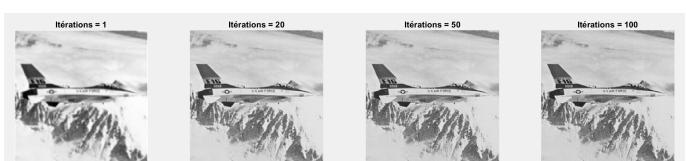
```

1 % Calculer Cx^(k) en conciliuant x^(k) par la
2 % RI
3 H = conv2(x, RI, 'valid');
4
5 % Calcul de l'erreur
6 size(H)
7 y_trunc = y(3:503, 3:503);
8 size(y_trunc)
9 e = H - y_trunc;
10
11 % On tronque le resultat
12 e(isnan(e)) = 0;
13
14 % Convolution
15 g = conv2(e, RI, 'full');
16
17 % Descente de gradient avec un pas fixe
18 x = x - alpha*g;

```

Code 4. Correction des effets de bord

Observation On constate qu'il n'y a plus d'effets de bord puis l'image en sortie de la convolution est réduite. A chaque itération les effets de bord ne s'additionnent plus et la dynamique de l'image n'est plus dégradée.

**Figure 9.** Image restaurée avec $\alpha = 0.015$ **Figure 10.** Image restaurée avec $\alpha = 1.03$ **Figure 11.** Image restaurée avec $\alpha = 0.9$

■ Régularisation et douceur

Algorithme de descente selon le gradient

Contexte

On cherche à minimiser le nouveau critère :

$$Q_{MCR}(\mathbf{x}) = (\mathbf{y} - \mathbf{T}\mathbf{C}\mathbf{x})^t(\mathbf{y} - \mathbf{T}\mathbf{C}\mathbf{x}) + \mu\mathbf{x}^t\mathbf{D}^t\mathbf{D}\mathbf{x} \quad (7)$$

Calcul du gradient pr rapport à x :

$$\begin{aligned} \frac{\partial Q_{MCR}(\mathbf{x})}{\partial \mathbf{x}} &= 2\mathbf{C}^t\mathbf{T}^t\mathbf{T}\mathbf{C}\mathbf{x} - 2\mathbf{C}^t\mathbf{T}^t\mathbf{y} + 2\mu\mathbf{D}^t\mathbf{D}\mathbf{x} \\ &= 2\mathbf{C}^t\mathbf{T}^t(\mathbf{T}\mathbf{C}\mathbf{x} - \mathbf{y}) + 2\mu\mathbf{D}^t\mathbf{D}\mathbf{x} \end{aligned}$$

L'algorithme de descente

$$\begin{aligned} \text{Si } \alpha < \frac{2}{\|\mathbf{C}^t\mathbf{T}^t\mathbf{T}\mathbf{C} + \mu\mathbf{D}^t\mathbf{D}\|_M}, \\ \mathbf{x}^{k+1} &= \mathbf{x}^k - \alpha \left(\mathbf{C}^t\mathbf{T}^t(\mathbf{T}\mathbf{C}\mathbf{x}^k - \mathbf{y}) + \mu\mathbf{D}^t\mathbf{D}\mathbf{x}^k \right). \end{aligned} \quad (8)$$

Pas de descente optimal

$$\alpha_{opt} = \frac{\mathbf{g}_k^t \mathbf{g}_k}{2\mathbf{g}_k^t (\mathbf{C}^t\mathbf{T}^t\mathbf{T}\mathbf{C} + 2 * \mu\mathbf{D}^t\mathbf{D}) \mathbf{g}_k} \quad (9)$$

Algorithme

1. Initialiser \mathbf{x}^0 à une valeur quelconque.
2. Calculer $\mathbf{C}\mathbf{x}^k$ en convoluant \mathbf{x}^k par la réponse impulsionale.
3. Calculer la différence e_k entre $\mathbf{T}\mathbf{C}\mathbf{x}^k$ et les données \mathbf{y} .
4. Calculer $\mathbf{C}^t\mathbf{T}^t(\mathbf{T}\mathbf{C}\mathbf{x}^k - \mathbf{y})$ en convoluant le résultat de l'étape précédente par la RI retournée.
5. Calculer $\mathbf{D}\mathbf{x}^k$ en convoluant \mathbf{x}^k par le filtre dérivateur $[-11]$.
6. Calculer $\mathbf{D}^t\mathbf{D}\mathbf{x}^k$ en convoluant $\mathbf{D}\mathbf{x}^k$ par le filtre dérivateur retournée $[1 - 1]$.
7. Mettre à jour \mathbf{x}^k en utilisant l'équation (15).
8. Retourner à l'étape 2 tant que l'algorithme n'a pas convergé, (Retourner à l'étape 2 tant que la norme du gradient est supérieure à un seuil que l'on fixera arbitrairement).

Implémentation

```

1 load Donnees1;
2 load superResx5.mat
3
4 % Initialize the image to restore
5 x = ones(size(super_resolved_image, 1), size
6 (super_resolved_image, 2));
7 y = super_resolved_image;
8
9 T = ones(size(super_resolved_image, 1), size
10 (super_resolved_image, 2));
11 C = ones(size(super_resolved_image, 1), size
12 (super_resolved_image, 2));
13
14 % Initialize the threshold
15 threshold = 0.5;
16
17 % Initialize the opt step
18 alpha = 0.9; % 0.015 ; 1.03
19
20 % MU to avoid edge effects
21 mu = 0.5;
22
23 % Normalisation de la reponse impulsionale
24 RI = RI/sum2(RI);
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44

```

```

% While the norm of the gradient is greater
% than the threshold
for iter = 1:100
    % Calculer Cx^(k) en convoluant x^(k)
    % par la RI
    H = conv2(x, RI, 'same');

    % Calcul de l'erreur
    e = H-y;

    % On tronque le resultat
    e(isnan(e)) = 0;

    % Convolution
    g = conv2(e, RI, 'same');

    % Descente de gradient avec un pas fixe
    x = x - alpha*g;

    figure(2);
    imshow(x, []);
    drawnow;
    title('Restored Image');
end

```

Code 5. Architecture de l'encodeur

■ Conclusion

Lors de cette deuxième séance de TP, nous avons abordé de la reconstruction d'image haute résolution à partir de plusieurs images basse résolution. Nous avons tout d'abord effectué une méthode simpliste d'addition décalage avec pondération en cas de chevauchement puis nous avons utilisé une méthode de déconvolution par approximation circulaire avec un estimateur des moindres carrés et avec les moindres carrés régularisés.

■ CONTACT

✉ yanis.gomes@ens-paris-saclay.fr
✉ Yanis Gomes